

Weighted Batch Threshold Encryption with Efficient DKG

Alexander Frolov¹, Aditi Partap², Max Resnick³, and Ertem Nusret Tas⁴

¹ University of Maryland sfrolov@umd.edu

² Stanford University aditi712@stanford.edu

³ Anza max.resnick@anza.xyz

⁴ a16z Crypto Research ntas@a16z.com

Abstract. Batch threshold encryption allows a committee to decrypt a selected batch of ciphertexts using one short pre-decryption key per party, while ciphertexts outside the batch remain private. This makes batch threshold encryption attractive for building encrypted mempools for blockchains, where validators reveal transactions selected for a block while pending transactions remain private. However, in proof-of-stake systems like Solana or Ethereum, authorization depends on each validator’s stake, which motivates a weighted version of batch threshold encryption. Naive approaches to weighting existing batch threshold encryption schemes lead to communication costs proportional to the weight assigned to each party. Existing schemes that avoid this cost either associate every ciphertext with a batch label or an encryption-time index.

We present the first weighted batch threshold encryption schemes that require neither batch/epoch labels nor encryption-time ciphertext indices, while keeping each party’s communication independent of its weight. We obtain two constructions by virtualizing prior schemes: the **BTX** construction of Agarwal et al. and the partial-fraction construction of Boneh et al. In both schemes’ decryption procedures, each party communicates a single group element for an entire batch, regardless of its assigned weight. We prove correctness, robustness, and security under weighted variants of the original schemes’ assumptions, and analyze the assumptions in generic bilinear groups. For the **BTX**-based construction, we also give a specialized distributed key generation protocol that avoids the general purpose MPC-based setup required by the previous state-of-the-art schemes. Finally, we implement our schemes to demonstrate their practical performance at parameter sizes relevant to blockchains.

Keywords: Threshold encryption · Batch encryption · Bilinear groups

Table of Contents

1	Introduction	3
2	Technical overview	5
2.1	Starting point: the BTX scheme	5
2.2	Weighted threshold decryption: the Ferveo trick	6
2.3	Combining the two schemes	7
2.4	Distributed key generation	8
2.5	Security with DKG-generated public parameters	9
3	Preliminaries	10
3.1	Basic preliminaries	10
3.2	Weighted batch threshold encryption	11
3.3	Robustness	12
3.4	Non-Interactive Zero-Knowledge Proofs (NIZKs)	13
3.5	Byzantine broadcast, the SCRAPE test, and weighted DKG	14
4	Weighted BTX	15
4.1	Security Assumption and CCA Security	18
4.2	Cost analysis for Construction 1	22
4.3	DKG	22
5	Partial-Fraction-Based Weighted BTE	37
5.1	Correctness and robustness	39
5.2	Analysis of Construction 5	40
5.3	Cost analysis for Construction 5	44
5.4	Distributed Key Generation for Construction 5	45
6	Experiments	47
6.1	Performance results for Constructions 1 and 5	47
6.2	Comparison to prior work	48
7	Additional related work	49
A	Choosing total weights for the Solana chain	52

1 Introduction

A *batch threshold encryption* (BTE) scheme [17] allows a committee of N parties to jointly decrypt a dynamically chosen batch of up to ℓ ciphertexts. During decryption, each committee member publishes a single, short *pre-decryption key* for the batch, and the pre-decryption keys from an authorized sub-committee can decrypt exactly the ciphertexts in the batch, while all other ciphertexts remain secret. With conventional threshold encryption, decrypting a batch requires each committee member to publish a decryption share *per ciphertext*, giving a total per-batch communication of $O(N\ell)$, while batch threshold encryption reduces this to $O(N)$, with one pre-decryption key per batch.

Batch threshold encryption is a natural building block for *encrypted mempools*. Users submit encrypted transactions, and once a block containing a batch of them is confirmed, a committee of validators decrypts that batch. Hiding the content of transactions until their order is fixed limits front-running and other forms of maximal extractable value (MEV) [19,21]. A related application is hiding proposals in Solana’s Multiple Concurrent Proposers (MCP) consensus design [23,29]. Properties of MCP alone do not prevent a proposer from changing its proposal after learning the content of others. Therefore, MCP additionally requires each honest participant to hide its proposal until the competing proposers cannot react with proposals that can still be included in an MCP block [23]. Encrypting proposals via a BTE scheme and decrypting them only after a block is confirmed achieves the desired hiding properties.

The first proposed BTE schemes associate every ciphertext with an epoch or batch label [17,18,4,30]. At most one pre-decryption key can be published per label without compromising security, so ciphertexts omitted from their designated batch must be re-encrypted and resubmitted. BEAT-MEV [14] removes this epoch restriction. However, its encryption algorithm requires an index from a fixed public domain, and each batch must contain at most one ciphertext per index to allow decryption. We call BEAT-MEV *index-dependent* for this reason. Avoiding index collisions requires coordination among encryptors, and adversaries may target the ciphertext indices used by a victim to keep their transactions out of a batch, thus allowing for censorship. Most recently, two schemes ([28,3] and [11]) build batch threshold encryption schemes which do not require ciphertext labels or indices. However, these schemes leave two important gaps: they support weighted decryption only through virtualization, and lack a practical distributed setup procedure for the secret keys held by the validators. In this work, we resolve both problems, which we explain in greater detail below.

Weighted decryption. Proof-of-stake blockchains like Solana or Ethereum tie authorization to stake. A set of validators is authorized when the total stake held by the validators exceeds a threshold. Any unweighted batch threshold encryption scheme can emulate a weighted access structure through *virtualization*, where a party with integer weight w_j simply runs the scheme as w_j virtual parties. Applying virtualization to the unweighted BTE schemes above gives the desired access structure, at the cost of each party publishing w_j pre-decryption keys per batch. The resulting $O(\text{wt}(T))$ communication (where $\text{wt}(T)$ is the total weight of an authorized set T), can be much larger than the number of parties and concentrates the burden on the highest-stake validators. The weighted BEAT-MEV scheme of Agarwal et al. [2], which we call BEAT^{++} , is a weighted BTE scheme with pre-decryption keys whose size is independent of per-party weights, though the scheme is still index-dependent. Thus, prior schemes in this area have weight-independent pre-decryption key sizes, or have ciphertexts with no labels/indices, but not both.

Distributed key generation. Second, prior state-of-the-art schemes for BTE without indices/labels lack specialized distributed setup procedures. While the per-party secret keys of the BTX scheme [28,3] and the partial-fraction-based scheme of [11] are highly structured, they are not of the right form for the existing Distributed Key Generation (DKG) protocols. As a result, these prior works propose using generic MPC methods for distributed setup [3,11]. In blockchain deployments where the committee of validators changes regularly (*e.g.*, Solana’s validator set is fixed only within a two-day epoch [7]), an efficient DKG is a prerequisite for practical use.

Our contributions. In this work, we construct the first weighted batch threshold encryption schemes with batch-independent, index-free ciphertexts and pre-decryption keys whose sizes are independent of the parties’ weights. We obtain two constructions by creating weighted analogs of BTX (in Section 4) and the partial-fraction-based BTE scheme of Boneh et al. (Section 5). In both constructions, a secret value is Shamir secret-shared, and each party is assigned a set of evaluation points matching its weight, with all of a party’s points tied to a single scalar secret. As a result, each party can publish a single $\mathbb{G}_1$ element as its pre-decryption key, regardless of its assigned weight. As in the original unweighted schemes, FFT-based techniques can be used to decrease the cross-ciphertext work of decrypting a batch of ℓ ciphertexts from $O(\ell^2)$ to $O(\ell \log \ell)$ group operations. Additionally, decryption committee members do not even need to compute their pre-decryption keys over identical batches, provided common ciphertexts retain consistent positions (Remark 1).

We prove the correctness and robustness of both constructions, and their static security under falsifiable security assumptions. We analyze the hardness of these assumptions in the generic bilinear group model.

Both schemes admit specialized DKG protocols. This is made possible by the same modification used to make the original BTE schemes weighted, namely tying each party’s shares to a single secret scalar. For the BTX-based construction, we give a synchronous DKG protocol with two logical rounds over a Byzantine broadcast channel that avoids a general-purpose MPC setup. We prove that the protocol lets all honest parties agree on well-formed keys. To deal with the ability of a rushing adversary to bias generated keys, we prove security of the combined DKG and BTE scheme directly, under a DKG-augmented variant of our security assumption. We further show that the partial-fraction-based BTE protocol in Section 5 admits an analogous DKG protocol (sketched in Section 5.4).

Finally, we implement both constructions in Rust over BLS12-381, together with re-implementations of three prior schemes: BEAT⁺⁺ [2], BTX [28,3], and the partial-fraction-based BTE scheme of [11] for comparison. On an Apple M4 processor, for a batch of 16 ciphertexts and a Solana-derived weight distribution with total weight $W = 1,580$, the mean total decryption times are 147.5ms for our BTX-based construction and 139.8ms for our partial fraction construction (Table 5), fitting within Solana’s 400ms block time. Assuming that the decryption committee is known in advance and certain values can be precomputed, these numbers drop to 21.0ms and 18.2ms respectively. We view these results as evidence that index-free weighted batch threshold encryption is practical at parameter sizes relevant to high-performance blockchains, and we report the full experimental comparison in Section 6.

Interesting directions for future work include designing an asynchronous variant of our DKG protocol, proving correctness and security of the conjectured DKG for the partial fraction construction, and closing the remaining constant-factor performance gap with index-dependent weighted schemes.

Paper organization. Section 2 gives a technical overview of our protocols. Section 3 provides preliminaries and definitions for weighted batch threshold encryption and the DKG model. Section 4 presents the weighted BTX construction, as well as the DKG protocol for this construction with security proofs (Section 4.3).. Section 5 presents the partial-fraction-based construction, its analysis, and its DKG. Section 6 contains our experimental evaluation and comparisons to prior schemes. Section 7 surveys additional related work. Appendix A describes how we derived the weight distributions for the experiments based on Solana’s stake distribution.

2 Technical overview

We next provide an overview of the weighted BTX construction (Section 4) and its distributed key generation protocol. We focus on this protocol in the overview since it is more elegant and easier to understand, and achieves more or less the same performance as the partial fraction-based construction in Section 5. The construction builds on the BTX batch threshold encryption scheme of Agarwal et al. and Policharla [3,28] (Section 2.1), along with the Ferveo protocol [9] for compressing the pre-decryption keys of a weighted party into a single group element (Section 2.2). We explain how the two ideas fit together in Section 2.3. Section 2.4 observes that the same modification that makes the scheme weighted also makes its setup amenable to a two step distributed key generation (DKG) protocol, in contrast to BTX’s setup, which requires multi-round MPC protocols. Finally, in Section 2.5, we provide intuition for the security of the end-to-end scheme. Throughout, ℓ denotes the batch size, N the number of parties, w_j the integer weight of party j , $W = \sum_j w_j$ the total weight, and t the corruption threshold. A set of parties is authorized if its total weight exceeds t . We write $[x]_1, [x]_2, [x]_T$ for group elements with discrete logarithm x in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and $e(\cdot, \cdot)$ for the pairing.

2.1 Starting point: the BTX scheme

Single decryptor. Consider first a single decryptor holding the secret key $\tau \in \mathbb{F}_p$. The public keys consist of the encryption key ek and the public decryption key dk :

$$\text{ek} := [\tau^{\ell+1}]_T, \quad \text{dk} := ([\tau^{\ell+1+d}]_2)_{d \in \mathcal{D}_\ell}, \quad \mathcal{D}_\ell := \{-\ell, \dots, -1, 1, \dots, \ell-1\}.$$

Here, dk contains the powers $\tau, \dots, \tau^{2\ell}$ in $\mathbb{G}_2$, with the single middle power $\tau^{\ell+1}$ omitted (*i.e.*, dk is *punctured*). A ciphertext is an ElGamal-style encryption in the target group,

$$\text{ct} = (\text{ct}[1], \text{ct}[2]) := ([r]_1, m + r \cdot \text{ek}),$$

with $\text{pad} [r\tau^{\ell+1}]_T$ (the encryptor also attaches a simulation-extractable proof of knowledge of r for CCA security). Encryption takes no batch label or index as input. A ciphertext only acquires its position when a decryptor places it in a batch. Given an ordered batch $(\text{ct}_1, \dots, \text{ct}_\ell)$ with $\text{ct}_k[1] = [r_k]_1$, the decryptor publishes the single pre-decryption key

$$\sigma := \sum_{k=1}^{\ell} \tau^k \cdot \text{ct}_k[1] = \left[\sum_{k=1}^{\ell} r_k \tau^k \right]_1 \in \mathbb{G}_1.$$

To obtain the i -th plaintext, the decryption algorithm pairs σ with the public element $[\tau^{\ell+1-i}]_2$:

$$e\left(\sigma, [\tau^{\ell+1-i}]_2\right) = \sum_{k=1}^{\ell} [r_k \tau^{\ell+1+(k-i)}]_T = r_i \cdot \mathbf{ek} + \sum_{\substack{k \in [\ell] \\ k \neq i}} e\left(\mathbf{ct}_k[1], [\tau^{\ell+1+(k-i)}]_2\right). \quad (1)$$

The diagonal term $k = i$ is exactly the pad of $\mathbf{ct}_i$. Every cross term $k \neq i$ has a nonzero offset $d = k - i \in \mathcal{D}_\ell$, so it can be computed publicly by pairing $\mathbf{ct}_k[1]$ with the element $[\tau^{\ell+1+d}]_2$ within $\mathbf{dk}$. Subtracting the cross terms exposes the pad, and $m_i = \mathbf{ct}_i[2] - r_i \cdot \mathbf{ek}$. Ciphertexts outside the batch remain secret, since pairing a fresh $[r]_1$ with any element of $\mathbf{dk}$ yields a pad shifted by a nonzero power of τ , but never the pad itself.

Threshold decryption. The batch key σ is linear in the vector $(\tau, \tau^2, \dots, \tau^\ell)$. To achieve threshold decryption among N parties, $\mathbf{BTX}$ secret shares each power τ^k separately and gives party j the shares $\langle \tau^k \rangle_j$ for all $k \in [\ell]$. Party j computes $\sigma_j := \sum_k \langle \tau^k \rangle_j \cdot \mathbf{ct}_k[1]$, and any $t + 1$ parties recover $\sigma = \sum_j \lambda_j \sigma_j$ by Lagrange interpolation in the exponent, after which decryption proceeds as above.

The threshold protocol has two drawbacks. First, every party holds ℓ field elements as its secret key, one share for each power of the same secret τ . Second, these shares are correlated in an unusual way: they are secret shares of $\tau, \tau^2, \dots, \tau^\ell$ for one common τ that should be deleted after setup. Generating them in a distributed fashion requires computing sharings of the powers $\tau^2, \dots, \tau^\ell$ from a sharing of τ , that is, secure multiplications of shared secrets, and both [3] and [28] defer the setup to a generic MPC protocol.

2.2 Weighted threshold decryption: the Ferveo trick

In a weighted setting, a natural approach is virtualization, where a party with weight w_j simply acts as w_j virtual parties in a non-weighted threshold encryption scheme. However, this requires the party to publish w_j pre-decryption keys per batch, for a total communication of $O(W)$ group elements. To resolve this problem, we next recall how Ferveo [9] achieves weighted (but not batched) threshold decryption with a single decryption share per party. Consider threshold ElGamal in the target group: a secret z is shared via a polynomial Z of degree at most t with $Z(0) = z$, the encryption key is $\mathbf{ek} = [z]_T$, and a ciphertext is $([r]_1, m + r \cdot \mathbf{ek})$. Party j owns a set Ω_j of w_j evaluation points, and the naïve weighted scheme would hand it the w_j shares $z_\omega := Z(\omega)$, $\omega \in \Omega_j$, each of which yields one decryption share $z_\omega \cdot [r]_1$ per ciphertext.

Ferveo’s observation is that a party need not hold its shares at all. Instead, party j chooses a single secret scalar $q_j \leftarrow \mathbb{F}_p^*$, and the public parameters contain the shares in *blinded* form,

$$Y_\omega := [z_\omega q_j]_2 \quad (\omega \in \Omega_j),$$

with the same blinding factor q_j for all of party j ’s points. To decrypt, party j publishes the single element $\sigma_j := q_j^{-1} \cdot [r]_1$, regardless of how many points it owns. Then, for any authorized set T and Lagrange coefficients $(\lambda_\omega)_{\omega \in \Omega_T}$,

$$\sum_{j \in T} e\left(\sigma_j, \sum_{\omega \in \Omega_j} \lambda_\omega Y_\omega\right) = \sum_{j \in T} \left[r q_j^{-1} \sum_{\omega \in \Omega_j} \lambda_\omega z_\omega q_j \right]_T = \left[r \sum_{\omega \in \Omega_T} \lambda_\omega z_\omega \right]_T = [rz]_T.$$

In other words, the pad can be “peeled off” by pairing party j ’s single share with the Lagrange combination of the public parameters.

Three aspects of Ferveo are of particular importance for our weighted batch threshold encryption scheme. First, each party's secret is a single scalar, independent of its weight. Second, its ciphertext decryption share is one group element, independent of its weight. Finally, the shares z_ω need not be known in the clear by any party after the setup. In Ferveo's DKG, a dealer with polynomial f publishes $[f(\omega)]_1$ together with $f(\omega) \cdot [q_j]_2 = [f(\omega)q_j]_2$, which anyone can check with one pairing. We return to this point in Section 2.4.

2.3 Combining the two schemes

We combine the two schemes above by augmenting the BTX construction with Ferveo-style public decryption keys:

$$\mathbf{ek} := [z]_T, \quad \mathbf{dk}_{\omega,d} := [z_\omega q_{\text{own}(\omega)}^d]_2 \quad (\omega \in \Omega_{[N]}, d \in \mathcal{D}_\ell), \quad (2)$$

where $z_\omega = Z(\omega)$ for a polynomial Z of degree at most t with $Z(0) = z$, $\text{own}(\omega)$ is the party that owns the point ω , and party j holds only $\mathbf{sk}_j := q_j$ as its secret key. Each element of $\mathbf{dk}$ is a secret share of the pad z , blinded by a nonzero power of the owner's secret key q_j . As in BTX, the middle decryption key elements with $d = 0$ ($[z_\omega]_2$ here) are deliberately missing from the public decryption key. The public key additionally contains a verification key $\mathbf{vk}_j := ([q_j^i]_2)_{i \in [\ell]}$ that will be used for verifying pre-decryption keys published by parties.

Encryption and pre-decryption. Encryption is unchanged from BTX: $\mathbf{ct} = ([r]_1, m + r \cdot \mathbf{ek})$ with pad $[rz]_T$, together with a simulation-extractable proof of knowledge of r . Ciphertexts therefore remain free of batch labels and indices. The pre-decryption key for party j for an ordered batch $(\mathbf{ct}_1, \dots, \mathbf{ct}_\ell)$ is given by

$$\mathbf{sbk}_j := \sum_{k=1}^{\ell} q_j^k \cdot \mathbf{ct}_k[1] = \left[\sum_{k=1}^{\ell} r_k q_j^k \right]_1,$$

which is exactly the BTX pre-decryption key in the case of a single decryptor, with τ replaced by party j 's secret key q_j (Section 2.1). Intuitively, since all points owned by a party are blinded by the powers of the same q_j , the party's secret key can be a single element q_j , with the pre-decryption key $\mathbf{sbk}_j$ containing powers of q_j . As in Ferveo, each party publishes one group element regardless of how many points it owns. The pre-decryption key can be publicly verified by checking $e(\mathbf{sbk}_j, [1]_2) = \sum_k e(\mathbf{ct}_k[1], \mathbf{vk}_{j,k})$.

Decryption. Let T' be an authorized set of parties whose pre-decryption keys have been verified, and let $(\lambda_{T',\omega})_{\omega \in \Omega_{T'}}$ be the Lagrange coefficients for the set $\Omega_{T'}$. Decryption first performs Ferveo's step of applying the Lagrange combination to the public decryption keys, which represent the blinded shares of the polynomial $Z(\cdot)$. For each party $j \in T'$ and each offset d , it combines the elements of the public decryption key at party j 's points into

$$K_{j,d} := \sum_{\omega \in \Omega_j} \lambda_{T',\omega} \mathbf{dk}_{\omega,d} = [\zeta_j q_j^d]_2, \quad \zeta_j := \sum_{\omega \in \Omega_j} \lambda_{T',\omega} z_\omega,$$

where ζ_j is the Lagrange-weighted sum of party j 's shares, so that $\sum_{j \in T'} \zeta_j = Z(0) = z$. To decrypt $\mathbf{ct}_i$, it then performs step (1) of BTX's decryption algorithm for each party j using the values $K_{j,-i}$

and $K_{j,k-i}$ for $k \in [\ell] \setminus \{i\}$, and adds up the results:

$$\begin{aligned}
& \sum_{j \in T'} e(\text{sbk}_j, K_{j,-i}) - \sum_{\substack{k \in [\ell] \\ k \neq i}} e\left(\text{ct}_k[1], \sum_{j \in T'} K_{j,k-i}\right) \\
&= \sum_{j \in T'} e\left(\sum_{k \in [\ell]} (q_j)^k \cdot [r_k]_1, [\zeta_j q_j^{-i}]_2\right) - \sum_{\substack{k \in [\ell] \\ k \neq i}} e\left([r_k]_1, \sum_{j \in T'} [\zeta_j q_j^{k-i}]_2\right) \\
&= \sum_{j \in T'} [\zeta_j \sum_{k \in [\ell]} r_k q_j^{k-i}]_T - \sum_{j \in T'} [\zeta_j \sum_{\substack{k \in [\ell] \\ k \neq i}} r_k q_j^{k-i}]_T \\
&= \left(\sum_{j \in T'} \zeta_j\right) [r_i]_T = r_i [z]_T = r_i \cdot \text{ek}
\end{aligned}$$

Informally, the diagonal term of $e(\text{sbk}_j, K_{j,-i})$ at $k = i$ is $[\zeta_j r_i]_T$, and every cross term $k \neq i$ has a nonzero multiplicative offset, so it can be peeled off by the public element $K_{j,k-i}$ exactly as the cross terms of BTX are canceled by the public powers. Summing the diagonal terms over T' reconstructs $r_i z$ by Lagrange interpolation as in Ferveo. Finally, $m_i = \text{ct}_i[2] - r_i \cdot \text{ek}$. Additionally, since the cross terms depend on k and i only through the offset $k - i$, the FFT acceleration techniques in BTX carry over verbatim, and decryption requires only $|T'| + 2$ pairings per ciphertext (see Section 4 for details).

Security. If the elements $[z_\omega]_2$ were public for $t + 1$ points, anyone could interpolate $[z]_2$ and pair it with $\text{ct}[1]$ to obtain the pad. However, with $d \neq 0$ and each element of the public decryption key blinded by a nonzero power of some party's q_j , pairing a fresh $[r]_1$ with any element of the public decryption key produces blinded shares $[r z_\omega q_j^d]_T$. A corrupt party can strip the blinding from the elements at its own points, but this yields at most $\text{wt}(\mathcal{C}) \leq t$ shares, which reveal nothing about z . For an honest party, q_j is unknown, and every public element involving one of its shares carries a nonzero power of q_j that no linear combination can cancel. In the generic bilinear group model, producing $[r z]_T$ requires z to lie in the linear span of the corrupt shares $(Z(\omega))_{\omega \in \Omega_C}$, which is impossible for $|\Omega_C| \leq t$ (Lemma 1).

We capture this hardness in a weighted version of the decisional power Diffie–Hellman assumption of BTX (Assumption 2). In the assumption, the adversary receives the public keys of (2), chooses a corruption set of weight at most t , receives the corrupt parties' secrets q_j , and must distinguish $[r z]_T$ from a uniformly random target group element given $[r]_1$. As in BTX, the assumption also hands the adversary the elements $[q_j^i]_1$ for honest parties j and $i \in [\ell]$, which are not part of the public parameters. The reduction uses them to answer pre-decryption queries on behalf of honest parties without knowing q_j . This gives static CCA security (Theorem 3) under a falsifiable assumption, and robustness follows from the verification of pre-decryption keys described above (Theorem 1).

2.4 Distributed key generation

Recall that the setup of BTX must produce secret sharings of $\tau, \tau^2, \dots, \tau^\ell$, nonlinear functions of the jointly held secret τ , which requires generic MPC protocols. In our protocol, the only jointly

held secret is z , and it enters every element of the public decryption key linearly. Hence, it can be handled by classic DKG protocols [27,24,16,25], where each party h acts as the dealer and secret shares a random polynomial f_h , and Z becomes the sum of all these polynomials. The nonlinear components are the powers q_j^d of the parties' secret keys, and each party can compute the powers $[q_j^d]_2$ of its own secret key and publish them in the group. Afterwards, a dealer who contributes a polynomial f_h to Z and knows $f_h(\omega)$ in the clear can compute $f_h(\omega) \cdot [q_j^d]_2$. This is the same structure that allows Ferveo's DKG to directly output the blinded shares $[f(\omega)q_j]_2$. Thus, the same modification that made the scheme weighted, namely tying all of a party's points (in the public decryption key) to one party-chosen secret key, is what enables a simple distributed setup. We now describe our protocol in more detail.

A two step protocol. The protocol runs over a Byzantine broadcast channel in a synchronous network, and in a blockchain deployment, the chain itself can serve as the broadcast channel. It has two logical steps (Constructions 2–4). In the first step, every party P_j samples its secret key $q_j \leftarrow \mathbb{F}_p^*$ and broadcasts $\overline{Q}_j := [q_j]_1$ and the powers $Q_{j,d} := [q_j^d]_2$ for $d \in \{-\ell, \dots, -1, 1, \dots, \ell\}$, together with a proof of knowledge of q_j relative to $\overline{Q}_j$. A recipient sets $Q_{j,0} := [1]_2$ and verifies the chain of pairing equations

$$e(\overline{Q}_j, Q_{j,d}) = e([1]_1, Q_{j,d+1}) \quad (d = -\ell, \dots, \ell - 1),$$

which forces every $Q_{j,d}$ to be the d -th power of the same scalar. The parties fix the set J of parties with valid transcripts and the evaluation domain Ω_J .

In the second step, every dealer P_h samples a uniformly random polynomial f_h of degree at most t and broadcasts the commitments $[f_h(0)]_1$, $[f_h(0)]_T$, and $[f_h(\omega)]_1$ for $\omega \in \Omega_J$, along with the blinded shares

$$D_{h,\omega,d} := f_h(\omega) \cdot Q_{\text{own}(\omega),d} = [f_h(\omega) q_{\text{own}(\omega)}^d]_2 \quad (\omega \in \Omega_J, d \in \mathcal{D}_\ell),$$

and a proof of knowledge of $f_h(0)$ relative to $[f_h(0)]_1$. Afterwards, a recipient runs the SCRAPE low-degree test [16] on the commitments $([f_h(x)]_1)_{x \in \{0\} \cup \Omega_J}$, which certifies without any pairings that they lie on a polynomial of degree at most t , checks that the target group commitment is consistent with $[f_h(0)]_1$, and checks that $e([f_h(\omega)]_1, Q_{\text{own}(\omega),d}) = e([1]_1, D_{h,\omega,d})$ holds for every ω and d . The parties fix the set I of dealers with valid transcripts and add up the accepted transcripts component-wise. With $Z := \sum_{h \in I} f_h$, the aggregates are $\text{ek} = [Z(0)]_T$ and $\text{dk}_{\omega,d} = [Z(\omega)q_{\text{own}(\omega)}^d]_2$, exactly the keys of (2). The second step is entirely public, and each party's secret key is selected in the first step.

The protocol requires two broadcast rounds and communicates $O(N\ell)$ group elements in the first step and $O(NW\ell)$ in the second. Its correctness properties, adapted from the weighted DKG framework of Das et al. [20], state that all honest parties agree on J , I , and the output transcript, and that the output keys have exactly the algebraic form prescribed by the setup of Construction 1 (Definition 12, Theorem 4). Agreement follows from the broadcast channel and the deterministic validation rules. Every honest party passes validation, so neither step aborts whenever the honest weight exceeds t .

2.5 Security with DKG-generated public parameters

Rushing adversaries and rogue keys. Correctness of the DKG protocol says nothing about the distribution of z . As observed by Gennaro et al. [24], a rushing adversary can wait for the honest

dealers' step 2 broadcasts and then choose its own polynomials to bias z . Moreover, a corrupt dealer could copy an honest dealer's transcript with every element negated, and thus cancel honest contributions to ensure that the final z is a known value. We prevent such rogue key attacks via a proof of knowledge of $f_h(0)$. Moreover, since every proof statement records the dealer's identity and the session, an honest proof cannot be replayed under another identity.

End-to-end security. Rather than arguing that the DKG output is close to uniform, we define security directly for the combined system (Definition 13). The adversary fixes a corrupted set of weight at most t , participates in the DKG protocol as a rushing adversary controlling the corrupt parties, and then plays the CCA security game of Section 3.2 with the keys produced by the DKG. We prove security (Theorem 6) under a DKG-augmented variant of Assumption 2, namely Assumption 5, that mirrors this interaction. The challenger plays the honest parties' first steps in the exponent, so their secret keys remain hidden, and lets the adversary choose the corrupt parties' keys after seeing the honest ones. It then publishes the second step contribution of a single designated honest dealer with a uniformly random polynomial f^* and constant term $a := f^*(0)$, together with the simulation handles $[q_j^k]_1$ for honest j , and asks the adversary to distinguish $[ra]_T$ from a uniform element given $[r]_1$.

The reduction embeds the designated dealer as one honest dealer, simulates the other honest dealers with polynomials it knows, simulates all honest proofs of knowledge, and extracts the constant terms of the corrupt dealers' accepted transcripts. It therefore knows $z_{\text{kn}} := z - a$, the sum of every contribution to z other than the designated dealer's, even though the corrupt contributions may depend arbitrarily on the honest transcripts. The final encryption key is $[a + z_{\text{kn}}]_T$, and the challenge pad under this key is $[r(a + z_{\text{kn}})]_T = Y + [rz_{\text{kn}}]_T$, where $Y = [ra]_T$ is the assumption challenge and the second term is computable from $[r]_1$ and z_{kn} . The reduction thus forms the challenge ciphertext by shifting the assumption challenge by a known quantity: if the challenge is real, the ciphertext is distributed exactly as in the real game, and if it is uniform, so is the shifted pad, no matter how the adversary chose z_{kn} . In other words, biasing the key only lets the adversary shift the challenge by an amount the reduction knows. Pre-decryption queries are answered with the simulation handles as before.

Generic hardness. It remains to justify the augmented security assumption, which we do in the generic bilinear group model (Lemma 2) by the same argument as for the basic security assumption. We use the standard generic group proof technique of treating public values as indeterminates, and reasoning about the linear spans of polynomials corresponding to these values. Together, these arguments show that the freedom of a rushing adversary in choosing its keys and polynomials gives no advantage against the encryption scheme.

3 Preliminaries

3.1 Basic preliminaries

Bilinear groups. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2, e) \leftarrow \text{GrpGen}(1^\lambda)$ be the descriptions of prime-order groups, their identity elements, and a nondegenerate bilinear pairing e . We use additive notation for group operations, so $e([a]_1, [b]_2) = [ab]_T$. All algorithms implicitly receive the group descriptions, and all scalar arithmetic is over $\mathbb{F}_p$.

Weights and sharing. Let N be the number of real parties in our scheme, and W be the total weight assigned to the N parties. Party j has positive integer weight w_j and owns a set $\Omega_j \subset \mathbb{F}_p^*$

of w_j evaluation points used for Shamir secret sharing. The sets $(\Omega_j)_{j \in [N]}$ are disjoint. For a set of parties T , set the notation

$$\Omega_T = \bigcup_{j \in T} \Omega_j, \quad \text{wt}(T) = |\Omega_T| = \sum_{j \in T} w_j, \quad W = \text{wt}([N]) = |\Omega_{[N]}|.$$

For a corruption bound $t < W$, a set T is authorized if $\text{wt}(T) > t$ and unauthorized if $\text{wt}(T) \leq t$. For authorized T , let $\lambda_{T,\omega}$ be the Lagrange coefficient at zero for each $\omega \in \Omega_T$ over interpolation set Ω_T .⁵ For every polynomial F of degree at most t ,

$$F(0) = \sum_{\omega \in \Omega_T} \lambda_{T,\omega} F(\omega).$$

The party which owns a share $\omega \in \Omega_j$ is denoted by $\text{own}(\omega) = j$.

3.2 Weighted batch threshold encryption

We now define the syntax and security properties of our scheme. The following syntax extends the batch threshold encryption definition of Boneh et al. [11] to a weighted access structure.

Definition 1 (Weighted BTE syntax). *A weighted batch threshold encryption scheme WBTE with maximum batch size ℓ , total weight W , threshold t , and message space $\mathcal{M}$ consists of the following PPT algorithms:*

- $\text{Setup}(1^\lambda, 1^\ell, 1^W, 1^t, (w_j)_{j \in [N]}) \rightarrow (\text{pp}, \text{ek}, \text{dk}, (\text{sk}_j, \text{vk}_j)_{j \in [N]})$: Given security parameter λ , maximum batch size ℓ , total weight W , threshold t and per-party weights w_j , outputs public parameters pp , a public encryption key ek , public decryption key dk , and per-party secret/verification keys $(\text{sk}_j, \text{vk}_j)_{j \in [N]}$. pp is an implicit input to the subsequent algorithms.
- $\text{Enc}(\text{ek}, m) \rightarrow \text{ct}$: encrypts m using ek .
- $\text{PreDec}(\text{sk}_j, (\text{ct}_i)_{i \in [\ell]}) \rightarrow \text{sbk}_j$ or $\perp$: Using a party's secret key sk_j , and a set of up to ℓ ciphertexts, outputs the corresponding **short** pre-decryption key sbk_j per party.
- $\text{PreDecVer}(\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j) \rightarrow 1$ or 0 : Using a party's verification key vk_j , a set of up to ℓ ciphertexts, and a claimed pre-decryption key sbk_j , outputs 1 if the pre-decryption key is the correct key corresponding to the batch of ciphertexts, and 0 otherwise.
- $\text{Dec}(\text{dk}, T, (\text{sbk}_j)_{j \in T}, (\text{ct}_i)_{i \in [\ell]}) \rightarrow (m_i)_{i \in [\ell]}$ or $\perp$: decrypts a batch of ciphertexts $(\text{ct}_i)_{i \in [\ell]}$ using a collection of pre-decryption keys $(\text{sbk}_j)_{j \in T}$ corresponding to a set of parties T with total weight greater than t . Decryption first verifies the submitted keys. It may discard invalid keys, but it can only return messages if the accepting set is authorized.

The verification keys $(\text{vk}_j)_{j \in [N]}$ are public, and we treat them as part of dk where convenient, so that Dec can verify the submitted keys.

Definition 2 (Correctness). *A weighted BTE scheme is correct if, for every authorized T , every message vector $(m_i)_{i \in [\ell]}$, honestly generated public parameters and secret keys, ciphertexts $\text{ct}_i \leftarrow \text{Enc}(\text{ek}, m_i)$ for $i \in [\ell]$, and honest pre-decryption keys $\text{sbk}_j \leftarrow \text{PreDec}(\text{sk}_j, (\text{ct}_i)_{i \in [\ell]})$ for all $j \in T$, decryption returns $(m_i)_{i \in [\ell]}$, except with negligible probability.*

⁵ I.e., $\lambda_{T,\omega}$ is the Lagrange basis polynomial $\prod_{u \in \Omega_T, u \neq \omega} \frac{X-u}{\omega-u}$ evaluated at $X = 0$.

We adapt the static batch chosen-ciphertext game of Boneh et al. [11] to this context by replacing cardinality with weight. The formulation below gives the adversary all honest pre-decryption keys, which is at least as strong as giving it the final decryption output.

Definition 3 (Static WB-IND-CCA security). *The security game, parameterized by ℓ , N , W and t , is as follows:*

1. *The challenger generates keys/parameters via Setup and gives $(\text{pp}, \text{ek}, \text{dk}, (\text{vk}_j)_{j \in [N]})$ to $\mathcal{A}$. The adversary then fixes an (unauthorized) corruption set $\mathcal{C} \subseteq [N]$ with $\text{wt}(\mathcal{C}) \leq t$ and receives $(\text{sk}_j)_{j \in \mathcal{C}}$.*
2. *$\mathcal{A}$ may submit batches of at most ℓ ciphertexts. For each batch, the challenger returns $\text{PreDec}(\text{sk}_j, \cdot)$ for every $j \notin \mathcal{C}$.*
3. *The adversary submits $m_0, m_1 \in \mathcal{M}$ of equal length. The challenger samples $b \leftarrow_{\$} \{0, 1\}$ and returns $\text{ct}^* \leftarrow_{\$} \text{Enc}(\text{ek}, m_b)$.*
4. *$\mathcal{A}$ may submit batches of at most ℓ ciphertexts which do not contain ct^* . For each batch, the challenger returns $\text{PreDec}(\text{sk}_j, \cdot)$ for every $j \notin \mathcal{C}$.*
5. *$\mathcal{A}$ outputs bit b' .*

The advantage of $\mathcal{A}$ against the weighted BTE scheme WBTE is

$$\text{Adv}_{\text{WBTE}, \mathcal{A}, \ell, N, W, t}^{\text{WB-IND-CCA}}(\lambda) = |\Pr[b' = b] - \frac{1}{2}|.$$

The scheme is secure if this advantage is negligible for every PPT adversary $\mathcal{A}$ and polynomial ℓ, N, W, t .

A weighted BTE scheme must be *efficient* in that (i) the size of the pre-decryption keys $(\text{sbk}_j)_{j \in [N]}$ is succinct in the maximum batch size ℓ and weight assigned to party j , and (ii) the size of the ciphertexts is succinct in the number of parties.

3.3 Robustness

A weighted batch threshold encryption scheme satisfies share robustness if it has accurate blaming and consistency, defined below. We adapt these properties from [12, Definitions 22.13 & 22.14].

Definition 4 (Accurate blaming). *Let*

$$\text{Blame}(\text{vk}, T, (\text{sbk}_j)_{j \in T}, (\text{ct}_i)_{i \in [\ell]}) = \{j \in T : \text{PreDecVer}(\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j) = 0\}$$

denote the set of parties whose pre-decryption keys are rejected by the verification algorithm PreDecVer. A weighted BTE scheme ensures accurate blaming if, for every honestly generated parameter tuple, every ciphertext batch $(\text{ct}_i)_{i \in [\ell]}$, every authorized submitted set $T \subseteq [N]$, every collection of shares $(\text{sbk}_j)_{j \in T}$, and every $j \in T$, membership $j \in \text{Blame}(\text{vk}, T, (\text{sbk}_j)_{j \in T}, (\text{ct}_i)_{i \in [\ell]})$ implies

$$\Pr [\text{PreDec}(\text{sk}_j, (\text{ct}_i)_{i \in [\ell]}) = \text{sbk}_j] = 0,$$

where the probability is over the internal randomness of PreDec.

Definition 5 (Consistency). *We define the following security game:*

1. *The challenger invokes $(\text{pp}, \text{ek}, \text{dk}, (\text{sk}_j, \text{vk}_j)_{j \in [N]}) \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^W, 1^t, (w_j)_{j \in [N]})$ and returns the output to Adv.*

2. Adv sends $((\text{ct}_i)_{i \in [\ell]}, S_1, \{\text{sbk}_j^{(1)}\}_{j \in S_1}, S_2, \{\text{sbk}_j^{(2)}\}_{j \in S_2})$, where $(\text{ct}_i)_{i \in [\ell]}$ is a batch of ℓ ciphertexts, $S_1, S_2 \subseteq [N]$ are (authorized) sets of parties with weight more than t , and $\{\text{sbk}_j^{(1)}\}_{j \in S_1}$ and $\{\text{sbk}_j^{(2)}\}_{j \in S_2}$ are sets of pre-decryption keys.
3. Define $\text{Dec}(\text{dk}, S_1, (\text{sbk}_j^{(1)})_{j \in S_1}, (\text{ct}_i)_{i \in [\ell]}) = (m_i^{(1)})_{i \in [\ell]}$ and $\text{Dec}(\text{dk}, S_2, (\text{sbk}_j^{(2)})_{j \in S_2}, (\text{ct}_i)_{i \in [\ell]}) = (m_i^{(2)})_{i \in [\ell]}$. Adv wins the game if $(m_i^{(1)})_{i \in [\ell]} \neq \perp$, $(m_i^{(2)})_{i \in [\ell]} \neq \perp$, and $(m_i^{(1)})_{i \in [\ell]} \neq (m_i^{(2)})_{i \in [\ell]}$.

The advantage of Adv, denoted by $\text{Adv}_{\text{WBTE}, \mathcal{A}, \ell, N, W, t}^{\text{Cons}}(\lambda)$, is the probability that it wins the game. The WBTE scheme is said to be consistent if this advantage is negligible for every PPT adversary, and polynomial ℓ, N, W, t .

3.4 Non-Interactive Zero-Knowledge Proofs (NIZKs)

Both constructions use a straightline (online) simulation-extractable NIZK Π_{DL} . For a statement $(C, E) \in \mathbb{G}_1 \times \mathbb{G}_T$, it proves the relation

$$R_{\text{DL}} = \{((C, E), r) : C = [r]_1\}.$$

It allows the extractor in our proofs to recover the scalar values needed to simulate a pre-decryption key for a given ciphertext. More generally, we treat Π_{DL} as a proof system for a *family* of relations which show knowledge of the discrete logarithm of a designated $\mathbb{G}_1$ component of the statement, while the remaining statement components are auxiliary values bound to the proof but unconstrained by the relation. Our DKG protocol (Section 4.3) uses two further relations of this form, and all properties below (in particular, simulation extractability and zero knowledge, with respect to one CRS and one simulator) are required to hold for the union of these relations. We also require Π_{DL} to be complete and zero-knowledge.

Definition 6 (Non-Interactive Zero Knowledge Proofs). Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a relation over sets $\mathcal{X}$ and $\mathcal{W}$. A NIZK scheme NIZK for $\mathcal{R}$ ⁶ is a tuple of PPT algorithms, $\text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify})$ defined as follows.

- $\text{Setup}(1^\lambda) \rightarrow \text{crs}$: given the security parameter λ , outputs a common reference string crs . The algorithms below implicitly take crs as input.
- $\text{Prove}(x, w) \rightarrow \text{pf}$: given a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$, outputs a proof pf .
- $\text{Verify}(x, \text{pf}) \rightarrow 0/1$: given statement $x \in \mathcal{X}$ and proof pf , outputs 0 or 1.

Definition 7 (Completeness). A NIZK scheme for $\mathcal{R}$ is complete if for all $(x, w) \in \mathcal{R}$,

$$\Pr[\text{Verify}(x, \pi) = 1 : \pi \leftarrow \text{Prove}(x, w)] \geq 1 - \text{negl}(\lambda).$$

Definition 8 (Zero-Knowledge [22]). A NIZK scheme for $\mathcal{R}$ is zero-knowledge if there exist PPT algorithms SimSetup and SimProve such that the following two experiments are computationally indistinguishable for every PPT distinguisher $\mathcal{A}$ making polynomially many adaptive proof queries $(x, w) \in \mathcal{R}$:

⁶ Later, we will instead have a family of indexed relations to capture public parameters with which the relation is specified.

- In the real experiment, sample $\text{crs} \leftarrow \text{Setup}(1^\lambda)$ and answer every query (x, w) with $\text{Prove}(\text{crs}, x, w)$.
- In the simulated experiment, sample $(\text{crs}, \text{td}_{\text{sim}}) \leftarrow \text{SimSetup}(1^\lambda)$ and answer every query (x, w) with $\text{SimProve}(\text{crs}, \text{td}_{\text{sim}}, x)$.

In both experiments, $\mathcal{A}$ receives crs before making its queries and may choose each query as a function of all previously received proofs.

Simulation extractability states that after seeing simulated proofs, every fresh accepting statement–proof pair has an extractable witness, except with negligible probability.

Definition 9 (Simulation extractability [15]). A NIZK scheme for $\mathcal{R}$, $\text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify})$, is said to be simulation extractable or an SE-NIZK if there exists a PPT extractor Ext and a PPT simulator $\text{Sim} = (\text{SimSetup}, \text{SimProve})$ such that for all PPT adversaries Adv , there exists a negligible function $\text{negl}(\lambda)$ for which the following holds:

$$\Pr \left[\begin{array}{l} \text{Verify}(x^*, \pi^*) = 1 \wedge \\ (x^*, \pi^*) \notin Q \wedge \\ (x^*, w^*) \notin \mathcal{R} \end{array} \middle| \begin{array}{l} (\text{crs}, \text{td}_{\text{Sim}}, \text{td}_{\text{Ext}}) \leftarrow \$ \text{SimSetup}(1^\lambda); \\ Q \leftarrow \emptyset; \\ (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sim}}}(\text{crs}); \\ w^* \leftarrow \text{Ext}(\text{crs}, \text{td}_{\text{Ext}}, x^*, \pi^*) \end{array} \right] \leq \text{negl}(\lambda).$$

Here, $\mathcal{O}_{\text{Sim}}$ denotes the simulation oracle:

$$\mathcal{O}_{\text{Sim}}(x) : \pi \leftarrow \text{SimProve}(\text{crs}, \text{td}_{\text{Sim}}, x); Q \leftarrow Q \cup \{(x, \pi)\}; \text{return } \pi$$

Note that freshness is required of the statement–proof pair: extraction is guaranteed even for a statement previously queried to the simulation oracle, as long as the accompanying proof differs from the simulated proof returned for it. We further require the simulator $(\text{SimSetup}, \text{SimProve})$ of the SE-NIZK to satisfy zero knowledge (Definition 8), with the extraction trapdoor td_{Ext} ignored by the zero-knowledge experiments; our security proofs use zero knowledge and simulation extractability with respect to this single simulator.

As noted, in our scheme, we will also require the extractor to be **straightline** (or online), which means that the extractor does not rewind the adversary.

3.5 Byzantine broadcast, the SCRAPE test, and weighted DKG

Our distributed key generation protocol (Section 4.3) assumes a synchronous network and treats Byzantine broadcast as a black-box primitive.

Definition 10 (Byzantine broadcast). A Byzantine broadcast protocol BB is executed among a set of parties $\{P_1, \dots, P_N\}$ that includes a designated broadcaster holding an initial input M . It satisfies the following properties:

- **Agreement.** If an honest party outputs M and another honest party outputs M' , then $M = M'$.
- **Validity.** If the designated broadcaster is honest and has initial input M , then every honest party outputs M .
- **Termination.** Every honest party outputs a message.

SCRAPE low-degree test. Our DKG protocol uses the low-degree test from SCRAPE [16]. Let $\mathcal{X} \subseteq \mathbb{F}_p$ be a set of m distinct evaluation points with $m \geq t + 2$, and let $(V_x)_{x \in \mathcal{X}} \in \mathbb{G}_1^m$ be a vector of group elements. The purpose of the SCRAPE test is to verify that there exists a polynomial $f(X) \in \mathbb{F}_p[X]$ of degree at most t such that $V_x = [f(x)]_1$ for all $x \in \mathcal{X}$. To verify this, SCRAPE samples a uniformly random polynomial $r(X)$ of degree at most $m - t - 2$ and defines

$$c_x := \frac{r(x)}{\prod_{y \in \mathcal{X} \setminus \{x\}} (x - y)} \quad (x \in \mathcal{X}).$$

It accepts if

$$\sum_{x \in \mathcal{X}} c_x V_x = [0]_1. \quad (3)$$

We denote the resulting procedure by $\text{SCRAPE.Test}(\mathcal{X}, t, V)$.

Definition 11 (Completeness and soundness of the SCRAPE test). *Let $X \subseteq \mathbb{F}_p$ contain m distinct points, and define*

$$\mathcal{L}_{X,t} := \left\{ (V_x)_{x \in X} \in \mathbb{G}_1^m : \begin{array}{l} \text{there exists } f \in \mathbb{F}_p[X] \text{ with } \deg(f) \leq t \\ \text{such that } V_x = [f(x)]_1 \text{ for every } x \in X \end{array} \right\}.$$

The SCRAPE low-degree test is complete if, for every $V \in \mathcal{L}_{X,t}$,

$$\Pr[\text{SCRAPE.Test}(X, t, V) = 1] = 1.$$

It is sound with error ε_{sc} if, for every $V \notin \mathcal{L}_{X,t}$,

$$\Pr[\text{SCRAPE.Test}(X, t, V) = 1] \leq \varepsilon_{\text{sc}},$$

where the probability is over the random polynomial sampled by the test. For the SCRAPE test described above, $\varepsilon_{\text{sc}} = 1/p$.

4 Weighted BTX

In Construction 1, we present a weighted batch threshold encryption scheme, WBTX, based on ideas from [9,2,3]. To support weights, **Setup** samples an encryption key, secret shares the key using a degree- t polynomial, and assigns secret shares to different parties according to their weights. **PreDec** allows one pre-decryption key to correspond to multiple evaluations of the polynomial sampled at setup. For batch size ℓ , total weight W and N parties, the scheme's keys consist of one $\mathbb{G}_T$ element for the encryption key, $(2\ell - 1)W$ $\mathbb{G}_2$ elements for the public decryption key, and ℓN $\mathbb{G}_2$ elements for the verification keys.

As in BEAT^{++} [2] and BTX [3], we can accelerate the decryption algorithm using the Fast Fourier transform (FFT). For each $i \in [\ell]$, the terms

$$\sum_{j \in T'} e \left(\text{sbk}_j, \sum_{\omega \in \Omega_j} \lambda_{T', \omega} \text{dk}_{\omega, -i} \right)$$

Setup($1^\lambda, 1^\ell, 1^W, 1^t, (w_j)_{j \in [N]} \rightarrow (\text{pp}, \text{ek}, \text{dk}, (\text{sk}_j, \text{vk}_j)_{j \in [N]}):$

- ▷ $\text{crs}_{\Pi_{\text{DL}}} \leftarrow \Pi_{\text{DL}}.\text{Setup}(1^\lambda)$.
- ▷ Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2, [1]_T, p, e) \leftarrow \text{GrpGen}(1^\lambda)$.
- ▷ Choose $\Omega_{[N]} \subseteq \mathbb{F}_p^*$ with $|\Omega_{[N]}| = W$ as an evaluation domain for secret sharing.
- ▷ Partition $\Omega_{[N]}$ into disjoint $(\Omega_j)_{j \in [N]}$ such that $|\Omega_j| = w_j$ for all $j \in [N]$.
- ▷ Let pp be $((\Omega_j)_{j \in [N]}, \text{crs}_{\Pi_{\text{DL}}}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2, [1]_T, p, e)$.
- ▷ Sample $z \leftarrow \mathbb{F}_p$ and a uniformly random polynomial $Z(X) \in \mathbb{F}_p[X]$ of degree $\leq t$ with $Z(0) = z$. Set $z_\omega := Z(\omega)$ for $\omega \in \Omega_{[N]}$ and $\text{ek} := [z]_T$.
- ▷ For all $j \in [N]$, sample $q_j \leftarrow \mathbb{F}_p^*$. Let $\text{sk}_j := q_j$ denote party j 's secret key and $\text{vk}_j := (\text{vk}_{j,i} := [(q_j)^i]_2)_{i \in [\ell]}$ be their verification key.
- ▷ Set $\text{dk}_{\omega,d} := [z_\omega q_{\text{down}(\omega)}^d]_2$ for all $d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}, \omega \in \Omega_{[N]}$.
- ▷ Output $(\text{pp}, \text{ek}, \text{dk} := (\text{dk}_{\omega,d})_{d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}, \omega \in \Omega_{[N]}}, (\text{sk}_j, \text{vk}_j)_{j \in [N]}).$

Enc(ek, m) $\rightarrow \text{ct}$:

- ▷ Given $m \in \mathbb{G}_T$, sample $r \leftarrow \mathbb{F}_p$, set $\text{ct}[1] := [r]_1$ and $\text{ct}[2] := m + r \cdot \text{ek}$.
- ▷ Obtain $\pi \leftarrow \Pi_{\text{DL}}.\text{Prove}(\text{ct}[1], \text{ct}[2]); r)$.
- ▷ Output $\text{ct} := (\text{ct}[1], \text{ct}[2], \pi)$.

PreDec($\text{sk}_j, (\text{ct}_i)_{i \in [\ell]} \rightarrow \text{sbk}_j$:

- ▷ Parse $\text{ct}_i = (\text{ct}_i[1], \text{ct}_i[2], \pi_i)$. If any $\Pi_{\text{DL}}.\text{Verify}((\text{ct}_i[1], \text{ct}_i[2]), \pi_i) = 0$, return $\perp$.
- ▷ Otherwise, return $\text{sbk}_j := \sum_{i=1}^\ell (q_j)^i \cdot \text{ct}_i[1] \in \mathbb{G}_1$.

PreDecVer($\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j \rightarrow 0/1$:

- ▷ Check $e(\text{sbk}_j, [1]_2) = \sum_{i=1}^\ell e(\text{ct}_i[1], \text{vk}_{j,i})$. If so, return 1; otherwise return 0.

Dec($\text{dk}, \{\text{sbk}_j\}_{j \in T}, (\text{ct}_i)_{i \in [\ell]} \rightarrow (m_i)_{i \in [\ell]}$:

- ▷ Verify the pre-decryption keys $\{\text{sbk}_j\}_{j \in T}$ by checking **PreDecVer**($\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j$) for every $j \in T$.
- ▷ Discard all pre-decryption keys which do not pass verification. Let T' be an authorized set of parties with accepted keys. Return $\perp$ if T' no longer has sufficient weight.
- ▷ Let $\text{dk}_{\omega,0} := 0$ for all $\omega \in \Omega_{[N]}$.
- ▷ For each $i \in [\ell]$, output

$$m_i = \text{ct}_i[2] - \sum_{j \in T'} e \left(\text{sbk}_j, \sum_{\omega \in \Omega_j} \lambda_{T', \omega} \text{dk}_{\omega, -i} \right) + \sum_{\substack{k \in [\ell] \\ k \neq i}} e \left(\text{ct}_k[1], \sum_{\omega \in \Omega_{T'}} \lambda_{T', \omega} \text{dk}_{\omega, k-i} \right)$$

Construction 1: Weighted BTX construction

can be computed using one multi-pairing of arity $|T'|$. On the other hand, for every nonzero offset $d \in \{-(\ell-1), \dots, \ell-1\}$, define

$$D_d := \sum_{\omega \in \Omega_{T'}} \lambda_{T', \omega} \mathbf{dk}_{\omega, d}, \quad D_0 := 0.$$

The remaining terms are

$$B_i := \sum_{\substack{k \in [\ell] \\ k \neq i}} e(\mathbf{ct}_k[1], D_{k-i}), \quad i \in [\ell],$$

which form a middle product between the $\mathbb{G}_1$ -valued sequence $(\mathbf{ct}_k[1])_k$ and the offset-indexed $\mathbb{G}_2$ -valued sequence $(D_d)_d$. More precisely, during decryption, we reverse the D_d -sequence and zero-pad both sequences to length $m := 2^{\lceil \log_2(2\ell) \rceil}$. A $\mathbb{G}_1$ -FFT of the ciphertext sequence and a $\mathbb{G}_2$ -FFT of the reversed sequence of public decryption key elements are then paired coordinate-wise and a truncated inverse $\mathbb{G}_T$ -FFT returns all $(B_i)_{i \in [\ell]}$ simultaneously. Consequently, decryption uses ℓ multi-pairings of arity $|T'|$ and m ordinary pairings. When ℓ is a power of two, $m = 2\ell$, giving an amortized cost of $|T'| + 2$ pairing inputs per ciphertext.

Finally, note that the submitted pre-decryption keys can also be batch verified by sampling $\delta_j \leftarrow \mathbb{F}_p$ for $j \in T$, and checking

$$e\left(\sum_{j \in T} \delta_j \mathbf{sbk}_j, [1]_2\right) = \sum_{i=1}^{\ell} e\left(\mathbf{ct}_i[1], \sum_{j \in T} \delta_j \mathbf{vk}_{j,i}\right).$$

Remark 1 (Pre-decryption keys for overlapping batches). The parties need not form their pre-decryption keys for the same batch. Let $I_j \subseteq [\ell]$ denote the indices used by party j , and write $\mathbf{ct}_{j,k}$ for the ciphertext assigned index k by that party. Fix a ciphertext $\mathbf{ct}_i$ and let T_i be a set of parties whose batches contain this ciphertext at the same index i . If T_i is authorized, then $\mathbf{ct}_i$ can be decrypted from their pre-decryption keys after verifying each $\mathbf{sbk}_j$ against its own indexed batch $(\mathbf{ct}_{j,k})_{k \in I_j}$. Indeed, in Construction 1, we can rewrite the decryption pad as a sum over $j \in T_i$ and, for the term belonging to j , subtract only the cross terms indexed by $I_j \setminus \{i\}$. The contribution of party j then reduces to

$$r_i \sum_{\omega \in \Omega_j} \lambda_{T_i, \omega} z_{\omega},$$

and summing over T_i gives $r_i z$. Consequently, if one party's batch is a subset of another's and their common ciphertexts retain the same indices, every ciphertext in the smaller batch can be decrypted whenever the parties contributing a key for it have sufficient total weight. The partial-fraction-based construction in Section 5 also supports this.

Theorem 1 (Correctness and robustness). *Construction 1 is correct and robust.*

Proof. Correctness follows by inspection. Consider the second and third terms in the decryption equation:

$$\begin{aligned}
& \sum_{j \in T'} e \left(\text{sbk}_j, \sum_{\omega \in \Omega_j} \lambda_{T', \omega} \text{dk}_{\omega, -i} \right) - \sum_{\substack{k \in [\ell] \\ k \neq i}} e \left(\text{ct}_k[1], \sum_{\omega \in \Omega_{T'}} \lambda_{T', \omega} \text{dk}_{\omega, k-i} \right) \\
&= \sum_{j \in T'} e \left(\sum_{k \in [\ell]} (q_j)^k \cdot [r_k]_1, \sum_{\omega \in \Omega_j} \lambda_{T', \omega} [z_\omega q_j^{-i}]_2 \right) \\
&\quad - \sum_{\substack{k \in [\ell] \\ k \neq i}} e \left([r_k]_1, \sum_{j \in T'} \sum_{\omega \in \Omega_j} \lambda_{T', \omega} [z_\omega (q_j)^{k-i}]_2 \right) \\
&= \left(\sum_{j \in T'} \sum_{\omega \in \Omega_j} \lambda_{T', \omega} z_\omega \right) [r_i]_T = r_i[z]_T = r_i \cdot \text{ek}
\end{aligned}$$

The sum simplifies to the ciphertext pad $r_i \cdot \text{ek}$, and $m_i = \text{ct}_i[2] - r_i \cdot \text{ek}$.

Next, we consider robustness. To show accurate blinding, observe that a correctly generated pre-decryption key always passes its verification check:

$$\begin{aligned}
e(\text{sbk}_j, [1]_2) &= e \left(\sum_{i=1}^{\ell} (q_j)^i \cdot \text{ct}_i[1], [1]_2 \right) \\
&= \sum_{i=1}^{\ell} e(\text{ct}_i[1], [(q_j)^i]_2) = \sum_{i=1}^{\ell} e(\text{ct}_i[1], \text{vk}_{j,i})
\end{aligned} \tag{4}$$

For consistency, observe that since $x \mapsto e(x, [1]_2)$ is injective on the prime-order group $\mathbb{G}_1$, any sbk_j accepted by **PreDecVer** must equal the prescribed value $\sum_{i=1}^{\ell} (q_j)^i \cdot \text{ct}_i[1]$. Writing $\text{ct}_i[1] = [r_i]_1$ for the (possibly adversarial) discrete logarithms r_i , the correctness computation above then shows that, for *every* authorized set T' of accepted keys, the decryption pad for ciphertext i equals $r_i[z]_T$, independently of T' . Hence any two authorized sets of accepted pre-decryption keys for the same batch decrypt it to the same messages.

Accurate blinding and consistency together imply the robustness of the weighted threshold construction. $\square$

4.1 Security Assumption and CCA Security

We use the following weighted analog of the decisional n -power Diffie–Hellman assumption used by BTX [3, Assumption 4.1].

Assumption 2 (Weighted Decisional d -Power Diffie–Hellman WDdP-DH). *Given parameters $(N, W, (w_j)_{j \in [N]}, (\Omega_j)_{j \in [N]}, t, \ell)$, define the two-stage experiment $\text{WDdP}_{\mathcal{B}}^b(1^\lambda)$, for $b \in \{0, 1\}$ and distinguisher $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, as follows. The challenger samples the description of the bilinear groups, $z, a_1, \dots, a_t \leftarrow_{\$} \mathbb{F}_p$, and $q_j \leftarrow_{\$} \mathbb{F}_p^*$ for every $j \in [N]$. It sets*

$$Z(X) = z + \sum_{s=1}^t a_s X^s, \quad z_\omega = Z(\omega),$$

and computes

$$\begin{aligned} \text{ek} &= [z]_T, \\ \text{vk}_{j,i} &= [q_j^i]_2 & (j \in [N], i \in [\ell]), \\ \text{dk}_{\omega,d} &= [z_\omega q_{\text{own}(\omega)}^d]_2 & (\omega \in \Omega_{[N]}, d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}), \\ H_{j,i} &= [q_j^i]_1 & (j \in [N], i \in [\ell]). \end{aligned}$$

Let pub consist of the group description together with $((\Omega_j)_{j \in [N]}, \text{ek}, (\text{vk}_{j,i})_{j,i}, (\text{dk}_{\omega,d})_{\omega,d})$. In the first stage,

$$(\mathcal{C} \subseteq [N], \text{st}) \leftarrow \mathcal{B}_1(\text{pub}, (H_{j,i})_{j,i}).$$

The experiment returns 0 if $\text{wt}(\mathcal{C}) > t$. Otherwise, the challenger samples $r, u \leftarrow_{\$} \mathbb{F}_p$, sets

$$X = [r]_1, \quad Y_0 = [rz]_T, \quad Y_1 = [u]_T,$$

and returns the output of

$$\mathcal{B}_2(\text{st}, (q_j)_{j \in \mathcal{C}}, X, Y_b).$$

Define

$$\begin{aligned} \text{Adv}_{\mathcal{B}}^{\text{WDdP-DH}}(\lambda) &= \left| \Pr[\text{WDdP}_{\mathcal{B}}^0(1^\lambda) = 1] \right. \\ &\quad \left. - \Pr[\text{WDdP}_{\mathcal{B}}^1(1^\lambda) = 1] \right|. \end{aligned}$$

The WDdP-DH assumption states that this advantage is negligible for every PPT distinguisher $\mathcal{B}$ and every polynomially bounded choice of valid parameters.

The elements $(H_{j,i})_{j,i}$ are used by the simulation and are given to the adversary, but are not public parameters of Construction 1. The assumption is falsifiable, since both distributions in the experiment are efficiently samplable.

Theorem 3 (Static WB-IND-CCA Security). *Suppose Assumption 2 holds and Π_{DL} is complete, zero knowledge, and straightline simulation-extractable for R_{DL} , with respect to the same simulator (Section 3.4). Then, Construction 1 is WB-IND-CCA secure.*

Proof. Let $\mathcal{A}$ be the WB-IND-CCA adversary (Definition 3). We consider the following games, and write S_k for the event that $\mathcal{A}$ outputs the challenge bit in Game k .

Game 0. This is the real WB-IND-CCA game. Hence

$$\left| \Pr[S_0] - \frac{1}{2} \right| = \text{Adv}_{\text{WB-TX}, \mathcal{A}}^{\text{WB-IND-CCA}}(\lambda).$$

Game 1. At setup, we replace the NIZK CRS by

$$(\text{crs}, \text{td}_{\text{sim}}, \text{td}_{\text{ext}}) \leftarrow \Pi_{\text{DL}}.\text{SimSetup}(1^\lambda),$$

and generate the proof within the challenge ciphertext using $\Pi_{\text{DL}}.\text{SimProve}$. Zero knowledge implies that there is a negligible function ϵ_{zk} such that

$$|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_{\text{zk}}.$$

Game 2. We change the pre-decryption oracle's implementation. Given a query $(\text{ct}_i)_{i \in I}$, where $\text{ct}_i = (\text{ct}_i[1], \text{ct}_i[2], \pi_i)$ and $I \subseteq [\ell]$, the oracle first performs the prescribed verification and returns $\perp$ if a proof is invalid. It then computes $r_i \leftarrow \Pi_{\text{DL}}.\text{Ext}(\text{crs}, \text{td}_{\text{ext}}, (\text{ct}_i[1], \text{ct}_i[2]), \pi_i)$ for every accepting ciphertext, and returns

$$\widehat{\text{sbk}}_j = \sum_{i \in I} r_i H_{j,i}$$

for every $j \notin \mathcal{C}$. Observe that if $\text{ct}_i[1] = [r_i]_1$ for all queried ciphertexts, then

$$\widehat{\text{sbk}}_j = \sum_{i \in I} r_i H_{j,i} = \sum_{i \in I} [r_i q_j^i]_1 = \sum_{i \in I} q_j^i \text{ct}_i[1]$$

is exactly the output of PreDec.

Before the challenge, the simulator has returned no statement–proof pair. After the challenge, the only such pair is the challenge pair $((\text{ct}^*[1], \text{ct}^*[2]), \pi^*)$. The WB-IND-CCA game rejects a query containing the exact challenge ciphertext. Hence, every pair on which the extractor is invoked is outside the simulator's output set, including the pairs which use $(\text{ct}^*[1], \text{ct}^*[2])$ as the statement along with a fresh proof. Then, by the simulation-extractability of the NIZK and a union bound over all queried ciphertexts, the probability that $\text{ct}_i[1] \neq [r_i]_1$ for an extracted $r_i \leftarrow \Pi_{\text{DL}}.\text{Ext}(\text{crs}, \text{td}_{\text{ext}}, (\text{ct}_i[1], \text{ct}_i[2]), \pi_i)$ is bounded by a negligible function ϵ_{ext} . (Formally, Games 1 and 2 proceed identically until the first extraction failure; since the challenger knows all secret keys, the SE reduction simulates the game and guesses the failing statement–proof pair, losing a multiplicative factor equal to the number of extracted proofs. Straightline extraction is what allows the extractor to be invoked online on each query without rewinding.) Therefore, it follows that

$$|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_{\text{ext}}.$$

Game 3. Replace the pad $[r^*z]_T$ in the challenge ciphertext by an independent uniformly random element of $\mathbb{G}_T$. We reduce this change to Assumption 2 with the distinguishing adversary $\mathcal{B}$. Given $(\text{pub}, (H_{j,i})_{j,i})$, the distinguisher $\mathcal{B}$ generates the simulated NIZK CRS and passes pub and the CRS to $\mathcal{A}$. It forwards the static corruption set chosen by $\mathcal{A}$ and gives the returned $(q_j)_{j \in \mathcal{C}}$ to $\mathcal{A}$. It answers pre-decryption queries, both before and after the challenge, as in Game 2. It can do so since the challenger provides the reduction with the $H_{j,i}$ terms. Upon receiving m_0, m_1 , it samples $\mu \leftarrow_{\$} \{0, 1\}$ and uses the received challenge (X, Y_b) to form

$$\text{ct}^* = (X, m_\mu + Y_b, \Pi_{\text{DL}}.\text{SimProve}(\text{crs}, \text{td}_{\text{sim}}, (X, m_\mu + Y_b))).$$

Finally, $\mathcal{B}$ returns 1 if and only if $\mathcal{A}$ outputs μ .

Now, if $b = 0$, the simulation of $\mathcal{B}$ is distributed identically to Game 2, and if $b = 1$, it is distributed identically to Game 3. Consequently,

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{B}}^{\text{WDdP-DH}}(\lambda).$$

In Game 3, the second component of the challenge ciphertext is uniform and independent of μ , so $\Pr[S_3] = 1/2$. Combining the games gives

$$\text{Adv}_{\text{WB-TX}, \mathcal{A}}^{\text{WB-IND-CCA}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{WDdP-DH}}(\lambda) + \epsilon_{\text{zk}} + \epsilon_{\text{ext}},$$

which is negligible. □

Lemma 1 (Bilinear generic group hardness). *Assumption 2 holds in the bilinear generic group model except with probability $O(Q^2(N+2)\ell/p)$, where Q denotes the number of generic group operations.*

Proof. Fix the set $\mathcal{C} \subseteq [N]$ returned in the first stage and reveal $(q_j)_{j \in \mathcal{C}}$ to the adversary. In the symbolic game, we treat

$$r, z, a_1, \dots, a_t, (q_j)_{j \notin \mathcal{C}}$$

as algebraically independent indeterminates, and the challenge exponent in the $b = 1$ branch is a fresh indeterminate. The indeterminate r occurs, with degree one, only in the exponent of $X = [r]_1$; pairing a handle $H_{j,i}$ or $[1]_1$ with public $\mathbb{G}_2$ material yields $\mathbb{G}_T$ terms without r , $\mathbb{G}_T$ elements cannot be paired again, and the $\mathbb{G}_T$ exponents 1 and z are also r -free. Hence, every $\mathbb{G}_T$ expression which contains r has an r -coefficient in the linear span of the public $\mathbb{G}_2$ exponents. Then, producing rz requires z to lie in the linear span of the Laurent polynomials

$$1, \quad q_j^i, \quad Z(\omega)q_{\text{own}(\omega)}^d \\ (\omega \in \Omega_{[N]}, j \in [N], i \in [\ell], d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}).$$

Now, consider a party $j \notin \mathcal{C}$. Since every d is nonzero, each term $Z(\omega)q_j^d$, $\omega \in \Omega_j$, contains a non-zero power of the independent indeterminate q_j . Comparing the coefficients shows that any combination of these terms which is independent of q_j contributes zero. The same observation applies to the terms q_j^i . On the other hand, the terms belonging to the corrupted parties reduce to the span of $(Z(\omega))_{\omega \in \Omega_{\mathcal{C}}}$. Thus, producing rz requires z to lie in the linear span of 1 and $(Z(\omega))_{\omega \in \Omega_{\mathcal{C}}}$, which we show to be false by contradiction.

Suppose $z = c_0 + \sum_{\omega \in \Omega_{\mathcal{C}}} c_{\omega} Z(\omega)$ for coefficients in $\mathbb{F}_p$. Comparing the coefficients of a_s , for $s = 1, \dots, t$, gives

$$\sum_{\omega \in \Omega_{\mathcal{C}}} c_{\omega} \omega^s = 0.$$

Since the points are distinct and nonzero and $|\Omega_{\mathcal{C}}| \leq t$, the corresponding Vandermonde matrix has full column rank, which implies $c_{\omega} = 0$ for $\omega \in \Omega_{\mathcal{C}}$. Comparing the coefficient of z then gives the desired contradiction.

Note that the symbolic argument holds uniformly for every valid corruption set with $\text{wt}(\mathcal{C}) \leq t$, so conditioning on the set returned in the first stage is harmless. Finally, the adversary can distinguish the symbolic and real games only if two distinct formal expressions in the symbolic game receive the same value in the real game. Let

$$D := \prod_{j \notin \mathcal{C}} q_j^{\ell} \quad \text{and} \quad M := (N - |\mathcal{C}|)\ell$$

so that multiplication by D clears every negative power of an honest party's q_j . The exponents represented in $\mathbb{G}_1$ have degree at most ℓ , while the cleared numerators of the exponents represented in $\mathbb{G}_2$ have degree at most $M + \ell$. Consequently, a pairing yields degree at most $M + 2\ell \leq (N + 2)\ell$. Hence, by the Schwartz–Zippel lemma and a union bound over at most $O(Q^2)$ pairs of expressions, after clearing the denominators of the Laurent polynomials, the adversary can distinguish the symbolic and real games with probability at most $O(Q^2(N + 2)\ell/p)$, which gives the stated bound. $\square$

Table 1: Storage and communication costs for Construction 1. N is the number of real parties, W is the total weight, $\tau = |T|$ is the number of parties in the accepted set, and ℓ is the batch size. We exclude costs related to the NIZK.

Quantity	Size
Encryption key $\mathbf{ek}$	$ \mathbb{G}_T $
Party secret key $\mathbf{sk}_j$	$ \mathbb{F}_p $
Party verification key $\mathbf{vk}_j$	$\ell \mathbb{G}_2 $
All parties' verification keys	$N\ell \mathbb{G}_2 $
Public decryption key elements $(\mathbf{dk}_{\omega,d})$	$(2\ell - 1)W \mathbb{G}_2 $
Public decryption key $\mathbf{dk}$, including all verification keys	$((2\ell - 1)W + N\ell) \mathbb{G}_2 $
Non-NIZK CRS / total public parameters $(\mathbf{ek}, \mathbf{dk})$	$((2\ell - 1)W + N\ell) \mathbb{G}_2 + \mathbb{G}_T $
Ciphertext, excluding π_{DL}	$ \mathbb{G}_1 + \mathbb{G}_T $
Padded batch of ciphertexts, excluding proofs	$\ell \mathbb{G}_1 + \ell \mathbb{G}_T $
One party's pre-decryption key $\mathbf{sbk}_j$	$ \mathbb{G}_1 $
Pre-decryptions from an accepted set T	$\tau \mathbb{G}_1 $
Decrypted message batch	$\ell \mathbb{G}_T $

Table 2: Computation costs for Construction 1. For $a \in \{1, 2, T\}$, SM_a is a scalar multiplication in $\mathbb{G}_a$ and $\text{MSM}_a(q)$ is a size q multi-scalar multiplication in $\mathbb{G}_a$. Group additions are not counted. P denotes a pairing, $\text{MP}(q)$ denotes a multi-pairing of size q , and $\text{FFT}_a(q)/\text{IFFT}_a(q)$ denote size q FFTs in $\mathbb{G}_a$. We set $m = 2^{\lceil \log_2(2\ell) \rceil}$.

Operation	Cost
Enc, one ciphertext	$1 \text{ SM}_1 + 1 \text{ SM}_T$
PreDec $_j$, one padded batch	$1 \text{ MSM}_1(\ell)$
Verify one pre-decryption σ_j	$1 \text{ MP}(\ell + 1)$
Dec, naive cross-term evaluation	$\ell \text{ MP}(\tau + \ell - 1) + \sum_{j \in T} \ell \text{ MSM}_2(w_j) + (\ell - 1) \text{ MSM}_2(W_T)$
Dec, FFT-based middle-product evaluation	$\ell \text{ MP}(\tau) + m \text{ P} + \text{FFT}_1(m) + \sum_{j \in T} \ell \text{ MSM}_2(w_j) + (\ell - 1) \text{ MSM}_2(W_T) + \text{FFT}_2(m) + \text{IFFT}_T(m)$

4.2 Cost analysis for Construction 1

In Table 1, we give the storage and communication costs for the Weighted BTX scheme. In Table 2, we give the computation costs for the Weighted BTX scheme.

4.3 DKG

Definitions. We first provide the definition and the security properties of the weighted distributed key generation protocol, adapting the definition of [20] to the public material required by the construction above. Throughout, we assume $W \geq 2t + 1$, so that the honest parties hold weight strictly greater than t .

Definition 12 (Weighted DKG for BTX). *A weighted distributed key generation protocol for Construction 1 is executed by parties $P_1, \dots, P_N$ with respective weights $(w_j)_{j \in [N]}$ and threshold t . Party P_j owns a public set $\Omega_j \subseteq \mathbb{F}_p^*$ of w_j evaluation points, where the sets $(\Omega_j)_{j \in [N]}$ are pairwise disjoint.*

At the end of the protocol, every party outputs a candidate public transcript

$$\text{trx} := (J, Z_0, (Z_\omega)_{\omega \in \Omega_J}, \text{ek}, (\text{vk}_j)_{j \in J}, \text{dk}),$$

where $J \subseteq [N]$ and the transcript contains public material only for the parties in J and the evaluation points in Ω_J . A party may instead output $\perp$ (abort); the properties below quantify over the honest parties' non- $\perp$ outputs.

The intended transcript corresponds to a polynomial $Z(X) \in \mathbb{F}_p[X]$ of degree at most t , with

$$z := Z(0), \quad z_\omega := Z(\omega) \quad (\omega \in \Omega_J),$$

and nonzero values $(q_j)_{j \in J} \in (\mathbb{F}_p^*)^{|J|}$. In particular,

$$Z_0 = [z]_1, \quad Z_\omega = [z_\omega]_1, \quad \text{ek} = [z]_T,$$

$$\text{vk}_j = ([q_j^k]_2)_{k \in [\ell]} \quad (j \in J),$$

and

$$\text{dk} := \left(\text{dk}_{\omega,d} := [z_\omega q_{\text{own}(\omega)}^d]_2 \right)_{\substack{\omega \in \Omega_J \\ d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}}}.$$

Every honest party $P_j \in J$ additionally retains $\text{sk}_j := q_j$ as its secret key.

We require the DKG protocol to satisfy the following correctness properties in the presence of an adversary that corrupts parties of total weight at most t .

- (C1) All subsets of $t+1$ share commitments appearing in an honest party's output transcript define the same unique master value. Namely, for every $S \subseteq \Omega_J$ with $|S| = t+1$, $\sum_{\omega \in S} \lambda_{S,\omega} Z_\omega = Z_0 = [z]_1$.
- (C2) All honest parties output the same encryption key $\text{ek} = [z]_T = e(Z_0, [1]_2)$, where z is the discrete logarithm of the unique master value guaranteed by C1.
- (C3) All honest parties output the same qualified set J and the same public transcript trx . Moreover, J contains every honest party and, for every $j \in J$, $k \in [\ell]$, $\omega \in \Omega_J$, and $d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}$,

$$\text{vk}_{j,k} = [q_j^k]_2 \quad \text{and} \quad \text{dk}_{\omega,d} = [z_\omega q_{\text{own}(\omega)}^d]_2,$$

for the same nonzero value q_j associated with P_j and the same degree-at-most- t polynomial $Z(X)$ guaranteed by C1.

As in [20], C1–C3 are correctness and agreement properties; they do not assert that z is uniformly distributed. Indeed, a rushing adversary can bias the distribution of z by choosing its contributions after seeing the honest parties' messages. We therefore define security of the encryption scheme directly in combination with the DKG protocol, by letting the adversary participate in key generation instead of receiving keys from a trusted setup.

Definition 13 (Static WB-IND-CCA security with DKG-generated keys). Let Π_{DKG} be a weighted DKG protocol for a weighted BTE scheme WBTE. The security game, parameterized by ℓ , N , the weights $(w_j)_{j \in [N]}$ with $W = \sum_j w_j$, and t with $W \geq 2t+1$, is as follows:

1. The challenger generates the public parameters $\mathbf{pp}$ (including group descriptions and any common reference string) and gives $\mathbf{pp}$, the weights, the evaluation domains $(\Omega_j)_{j \in [N]}$, threshold t , batch bound ℓ , and a fresh session identifier to $\mathcal{A}$. The adversary fixes a corruption set $\mathcal{C} \subseteq [N]$ with $\text{wt}(\mathcal{C}) \leq t$.
2. The challenger and $\mathcal{A}$ jointly execute Π_{DKG} : the challenger plays every honest party $j \notin \mathcal{C}$ according to the protocol, while $\mathcal{A}$ chooses all messages of the corrupted parties, and may choose each of them after seeing the honest parties' messages of the same round (a **rushing** adversary). If an honest party aborts, if the honest parties' output transcripts differ, or if some honest party outputs no secret key, the game ends and the output of the game is a uniformly random bit. Otherwise, let $\text{trx} = (J, \dots, \text{ek}, (\text{vk}_j)_{j \in J}, \text{dk})$ be the common public transcript output by the honest parties, which is given to $\mathcal{A}$, and let sk_j be the secret key retained by each honest party j .
3. $\mathcal{A}$ may submit batches of at most ℓ ciphertexts. For each batch, the challenger returns $\text{PreDec}(\text{sk}_j, \cdot)$ for every $j \notin \mathcal{C}$.
4. $\mathcal{A}$ submits $m_0, m_1 \in \mathcal{M}$ of equal length. The challenger samples $b \leftarrow \{0, 1\}$ and returns $\text{ct}^* \leftarrow \text{Enc}(\text{ek}, m_b)$.
5. $\mathcal{A}$ may submit batches of at most ℓ ciphertexts which do not contain ct^* , answered as in step 3.
6. $\mathcal{A}$ outputs bit b' , and the output of the game is 1 if and only if $b' = b$.

The advantage of $\mathcal{A}$ is

$$\text{Adv}_{\text{WBTE}, \Pi_{\text{DKG}}, \mathcal{A}, \ell, N, W, t}^{\text{WB-IND-CCA-DKG}}(\lambda) = |\Pr[\text{the game outputs } 1] - \frac{1}{2}|$$

The combined scheme $(\Pi_{\text{DKG}}, \text{WBTE})$ is secure if this advantage is negligible for every PPT adversary $\mathcal{A}$ and polynomial ℓ, N, W, t .

Compared to Definition 3, the adversary fixes its corruption set before any keys exist and obtains the corrupted parties' keys by choosing them itself during the DKG, rather than receiving honestly generated ones; this is the natural static corruption analog when keys are generated in a distributed fashion.

Protocol. We next describe the distributed key generation protocol for Construction 1 in the synchronous network model. Let $P_1, \dots, P_N$ have respective integer weights $(w_j)_{j \in [N]}$, let $W := \sum_{j \in [N]} w_j$, and suppose the adversary $\mathcal{A}$ statically corrupts parties of total weight at most t . We assume $W \geq 2t + 1$. The protocol uses the Byzantine broadcast primitive of Definition 10 and the SCRAPE low-degree test of Section 3.5. Its goal is to realize the weighted DKG functionality of Definition 12 (i.e. satisfy the correctness properties C1–C3). Recall that C1–C3 do not assert that the master secret z is uniformly distributed. Therefore, we later prove security of the encryption scheme in combination with the DKG protocol, as in Definition 13. In the description below, we treat Byzantine broadcast as a black-box primitive. In blockchain applications, the protocol can use the underlying blockchain as a secure Byzantine broadcast channel.

Informal sketch. The DKG protocol has two steps, one for generating the q_j -powers, and one for generating the sharing polynomial and the public decryption key. Specifically, in the first step, every party P_j samples its secret key $q_j \leftarrow \mathbb{F}_p^*$. It broadcasts $\bar{Q}_j := [q_j]_1$ and $Q_{j,d} := [q_j^d]_2$ for $d \in \{-\ell, \dots, -1, 1, \dots, \ell\}$. The positive powers form the party's verification key, while the remaining powers are needed to construct the public decryption key. To prove that all these elements contain

powers of the same scalar, P_j gives an extractable discrete log proof of knowledge of q_j relative to $\overline{Q}_j$. Taking $Q_{j,0} = [1]_2$, a recipient verifies the chain of pairing equations

$$e(\overline{Q}_j, Q_{j,d}) = e([1]_1, Q_{j,d+1}) \quad \text{for every } d \in \{-\ell, \dots, \ell-1\}.$$

The proof of knowledge binds $\overline{Q}_j$ to a known scalar, and the pairing chain then forces every $Q_{j,d}$ to be the corresponding power of that same scalar, including the negative powers.

In the second step, each dealer P_i samples an independent uniformly random polynomial $f_i(X) \in \mathbb{F}_p[X]$ of degree at most t . It sets $z^{(i)} := f_i(0)$ and $z_\omega^{(i)} := f_i(\omega)$ for every $\omega \in \Omega_J$, where J denotes the set of parties whose round 1 transcripts were valid, and computes

$$Z_{i,0} := [z^{(i)}]_1, \quad Z_{i,T} := [z^{(i)}]_T, \quad Z_{i,\omega} := [z_\omega^{(i)}]_1,$$

together with

$$D_{i,\omega,d} := [z_\omega^{(i)} q_{\text{own}(\omega)}^d]_2 \quad \text{for every } \omega \in \Omega_J \text{ and } d \in \{-\ell, \dots, -1, 1, \dots, \ell-1\}.$$

To prevent rogue key attacks, we require each dealer to also provide an extractable discrete log proof of knowledge of $z^{(i)}$ relative to $Z_{i,0}$. It then broadcasts the resulting transcript to all parties.

A recipient first verifies the proof of knowledge. It then checks $e(Z_{i,0}, [1]_2) = Z_{i,T}$, applies the SCRAPE low-degree test [16] to $(Z_{i,x})_{x \in \{0\} \cup \Omega_J}$, and verifies, for every relevant ω and d , that

$$e(Z_{i,\omega}, Q_{\text{own}(\omega),d}) = e([1]_1, D_{i,\omega,d}).$$

Thus, the share commitments lie on one degree $\leq t$ polynomial whose constant term is bound to the proof, the target group commitment is consistent with that constant, and every public decryption key element uses the correct share and power of q .

Finally, each party locally adds up the corresponding components of all the valid transcripts. For instance, $Z_0 = \sum_{i \in I} Z_{i,0}$, where I is the set of parties whose Round 2 transcripts were valid. For the security proofs, we assume that both proofs of knowledge admit straightline extraction.

Protocol Description. The public input consists of $\mathbf{pp}$, the weights, threshold t , batch bound ℓ , the evaluation domains, and a fresh session identifier $\mathbf{sid}$. Let

$$\mathcal{E}_\ell := \{-\ell, \dots, -1, 1, \dots, \ell\}, \quad \mathcal{D}_\ell := \{-\ell, \dots, -1, 1, \dots, \ell-1\}.$$

The protocol uses a straightline extractable proof of knowledge Π_{DL} for the following two relations, whose statements match the round 1 and round 2 transcripts below:

$$\begin{aligned} R^q &:= \{((\mathbf{sid}, j, \overline{Q}, (Q_d)_{d \in \mathcal{E}_\ell}); q) : \overline{Q} = [q]_1\}, \\ R^z &:= \{((j, \text{ctx}, Z_0, Z_T, (Z_\omega)_\omega, (D_{\omega,d})_{\omega,d}); z) : Z_0 = [z]_1\}. \end{aligned}$$

In each case, the witness is the discrete logarithm of one designated component. The remaining components are auxiliary public values bound to the proof.

Round 1 Sharing Phase for P_j :

- ▷ Sample $q_j \leftarrow \mathbb{F}_p^*$ and compute

$$\overline{Q}_j := [q_j]_1, \quad Q_{j,d} := [q_j^d]_2 \quad (d \in \mathcal{E}_\ell).$$

- ▷ Let the proof statement be

$$S_j^q := (\text{sid}, j, \overline{Q}_j, (Q_{j,d})_{d \in \mathcal{E}_\ell})$$

and compute

$$\pi_j^q \leftarrow \Pi_{\text{DL}}.\text{Prove}(S_j^q; q_j).$$

- ▷ Invoke $\text{BB}[j]$ with input $M_j^q := (S_j^q, \pi_j^q)$. (Each logical round uses its own set of N Byzantine broadcast instances, tagged by $(\text{sid}, \text{round}, j)$.)

Round 1 Validation Phase:

- ▷ Wait until all N round 1 Byzantine broadcast instances have terminated. Let $\widetilde{M}_j^q$ be the output of $\text{BB}[j]$, for all $j \in [N]$.
- ▷ For each $j \in [N]$, parse $\widetilde{M}_j^q$ as (S_j^q, π_j^q) . To verify whether $\widetilde{M}_j^q$ is valid, check that the identity recorded in S_j^q equals the index j of the broadcasting instance and that the recorded session identifier equals sid , that $\Pi_{\text{DL}}.\text{Verify}(S_j^q, \pi_j^q) = 1$, and, taking $Q_{j,0} = [1]_2$, that the following holds for every $d \in \{-\ell, \dots, \ell - 1\}$:

$$e(\overline{Q}_j, Q_{j,d}) = e([1]_1, Q_{j,d+1}). \quad (5)$$

- ▷ Set

$$J := \{j \in [N] : \widetilde{M}_j^q \text{ is valid}\}, \quad \Omega_J := \bigcup_{j \in J} \Omega_j.$$

Abort if $|\Omega_J| \leq t$.

Construction 2: Round 1 of our DKG protocol: generation of the q -powers.

Let ctx_J denote the round 2 context consisting of the original public input, J , and the list of accepted round 1 transcripts. We will bind the round 2 proof to this context and to the dealer's identity, to prevent replay across sessions and across dealers. The SCRAPE challenge is additionally bound to the dealer's complete commitment vector: deriving the challenge only from values fixed before Round 2 would let a corrupt dealer choose its commitments as a function of a known challenge and defeat the test. In the noninteractive test, the challenge polynomial is derived via a hash of the complete, fixed statement (i.e. the Fiat-Shamir transform), so all honest parties make the same accept decision; we assume the resulting noninteractive test is sound (in the random oracle model, its soundness error grows by a factor of the number of hash queries).

Round 2 Sharing Phase for P_j :

- ▷ Sample a uniformly random polynomial $f_j(X) \in \mathbb{F}_p[X]$ of degree at most t . Define

$$z^{(j)} := f_j(0) \quad \text{and} \quad z_\omega^{(j)} := f_j(\omega) \quad (\omega \in \Omega_J).$$

- ▷ Compute

$$Z_{j,0} := [z^{(j)}]_1, \quad Z_{j,T} := [z^{(j)}]_T, \quad Z_{j,\omega} := [z_\omega^{(j)}]_1 \quad (\omega \in \Omega_J).$$

For every $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$, compute

$$D_{j,\omega,d} := z_\omega^{(j)} Q_{\text{own}(\omega),d} = [z_\omega^{(j)} q_{\text{own}(\omega)}^d]_2.$$

- ▷ Define the proof statement

$$S_j^z := \left(j, \text{ctx}_J, Z_{j,0}, Z_{j,T}, (Z_{j,\omega})_{\omega \in \Omega_J}, (D_{j,\omega,d})_{\substack{\omega \in \Omega_J \\ d \in \mathcal{D}_\ell}} \right),$$

and compute $\pi_j^z \leftarrow \Pi_{\text{DL}}.\text{Prove}(S_j^z; z^{(j)})$.

- ▷ Invoke $\text{BB}[j]$ with input $M_j^z := (S_j^z, \pi_j^z)$.

Round 2 Validation Phase:

- ▷ Wait until all N round 2 Byzantine-broadcast instances have terminated. Let $\widetilde{M}_j^z$ denote the output of $\text{BB}[j]$ for all $j \in [N]$.
- ▷ Parse $\widetilde{M}_j^z$ as (S_j^z, π_j^z) . Verify whether $\widetilde{M}_j^z$ is valid via the following:
 1. The identity recorded in S_j^z equals the index j of the broadcasting instance, and S_j^z refers to the context ctx_J fixed after Round 1.
 2. $\Pi_{\text{DL}}.\text{Verify}(S_j^z, \pi_j^z) = 1$.
 3. Check that the source- and target-group commitments to the constant term are consistent: $e(Z_{j,0}, [1]_2) = Z_{j,T}$.
 4. Run the SCRAPE test on $(Z_{j,x})_{x \in \{0\} \cup \Omega_J}$ with degree bound t and a challenge polynomial derived from hashing $(\text{ctx}_J, j, (Z_{j,x})_{x \in \{0\} \cup \Omega_J})$ (*i.e.* bound to the dealer's complete commitment vector).
 5. For $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$, check $e(Z_{j,\omega}, Q_{\text{own}(\omega),d}) = e([1]_1, D_{j,\omega,d})$.
- ▷ Define the valid round 2 dealer set

$$I := \left\{ j \in [N] : \widetilde{M}_j^z \text{ is valid} \right\}.$$

Abort if $\text{wt}(I) \leq t$. (A round 2 dealer need not belong to J ; it shares over the fixed domain Ω_J using the public q -powers of the point owners.)

Construction 3: Round 2 of our DKG protocol: polynomial contributions.

Aggregation and output:

▷ Compute

$$\begin{aligned} Z_0 &:= \sum_{j \in I} Z_{j,0}, & Z_T &:= \sum_{j \in I} Z_{j,T}, \\ Z_\omega &:= \sum_{j \in I} Z_{j,\omega}, & D_{\omega,d} &:= \sum_{j \in I} D_{j,\omega,d}, \end{aligned}$$

for every $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$.

▷ Output the public transcript

$$\text{trx} := (J, Z_0, (Z_\omega)_{\omega \in \Omega_J}, \text{ek}, (\text{vk}_j)_{j \in J}, \text{dk}),$$

where

$$\text{ek} := Z_T, \quad \text{vk}_j := (Q_{j,k})_{k \in [\ell]} \quad (j \in J),$$

and

$$\text{dk} := (D_{\omega,d})_{\substack{\omega \in \Omega_J \\ d \in \mathcal{D}_\ell}}.$$

▷ Every honest party $P_j \in J$ retains $\text{sk}_j := q_j$ as its secret key.

Construction 4: Round 2 of our DKG protocol: aggregation and key derivation.

Remark 2 (Alternative setup via scalar DKG). We briefly sketch a variant of this DKG protocol with reduced communication. Assuming an existing encryption PKI and a compatible one round publicly verifiable scalar DKG, parties could obtain their scalar shares $z_\omega = Z(\omega)$ in parallel with publishing their powers of q_j in $\mathbb{G}_2$, and then publish $[z_\omega q_{\text{own}(\omega)}^d]_2$ in a second round. The scalar DKG must provide consistent commitments in $\mathbb{G}_1$ without enabling public reconstruction of $[Z(0)]_2$. Each recipient would thus expand its final shares once, giving total broadcast payload $C_{\text{scalarDKG}} + O(W\ell)$. In the case where each party has weight 1, using a scalar DKG with quadratic payload to instantiate this protocol would yield a protocol with $O(N^2 + N\ell)$ total broadcast payload, compared with $O(N^2\ell)$ for the protocol above.

Theorem 4 (Correctness of the synchronous DKG). *Suppose that $W \geq 2t + 1$, that Π_{DL} is complete, and that the noninteractive SCRAPE test (Section 4.3) is complete and sound. Suppose further that the Byzantine broadcast protocol BB satisfies Definition 10.*

Then, against every PPT adversary $\mathcal{A}$ corrupting parties of combined weight at most t , the protocol of Constructions 2, 3, and 4 terminates and satisfies C1–C3 of Definition 12, except with probability $\text{negl}(\lambda)$.

Proof. We condition on the event that no SCRAPE soundness failure occurs. By assumption, the complementary event occurs with probability at most $\text{negl}(\lambda)$. The argument below uses only the pairing checks and the SCRAPE test. It does not invoke soundness or extraction of the proofs of knowledge.

Round 1 agreement. For every $j \in [N]$, agreement and termination of the Byzantine broadcast BB ensure that all honest parties obtain the same output $\widetilde{M}_j^q$ from the instance with designated broadcaster P_j . Consequently, all honest parties obtain the same sender-indexed vector $(\widetilde{M}_j^q)_{j \in [N]}$. Because transcript validation is deterministic, all honest parties derive the same qualified set

$$J = \left\{ j \in [N] : \widetilde{M}_j^q \text{ is valid} \right\},$$

the same accepted round 1 transcripts, and the same evaluation domain

$$\Omega_J = \bigcup_{j \in J} \Omega_j.$$

If P_j is honest, validity of BB and completeness of Π_{DL} , together with correctness of the pairing checks, imply that $\widetilde{M}_j^q$ is accepted except with negligible probability. Hence every honest party belongs to J . Let $\mathcal{H} \subseteq [N]$ denote the set of honest parties. Then $|\Omega_J| = \text{wt}(J) \geq \text{wt}(\mathcal{H}) \geq W - t \geq t + 1$. Therefore, no honest party aborts after Round 1.

Well-formedness of the q_j powers. Fix any $j \in J$. Since $\mathbb{G}_1$ is a cyclic group of prime order with generator $[1]_1$, there exists a unique scalar $q_j \in \mathbb{F}_p$ such that $\overline{Q}_j = [q_j]_1$. Likewise, for every $d \in \mathcal{E}_\ell$, there exists a unique $a_{j,d} \in \mathbb{F}_p$ such that $Q_{j,d} = [a_{j,d}]_2$. Set

$$Q_{j,0} := [1]_2 \quad \text{and} \quad a_{j,0} := 1.$$

The sequence of pairing checks gives

$$e(\overline{Q}_j, Q_{j,d}) = e([1]_1, Q_{j,d+1}) \quad (-\ell \leq d \leq \ell - 1).$$

Substituting the corresponding exponents and using the non-degeneracy of the pairing yields

$$q_j a_{j,d} = a_{j,d+1} \quad (-\ell \leq d \leq \ell - 1). \quad (6)$$

In particular, taking $d = -1$ gives

$$q_j a_{j,-1} = a_{j,0} = 1,$$

so $q_j \neq 0$. Therefore, by induction,

$$Q_{j,d} = [q_j^d]_2 \quad (d \in \mathcal{E}_\ell). \quad (7)$$

Round 2 agreement. Round 2 begins only after the common set J and the accepted round 1 transcripts have been fixed. Thus, all honest parties use the same context ctx_J . For every $j \in [N]$, agreement and termination of BB ensure that all honest parties obtain the same output $\widetilde{M}_j^z$ from the instance with designated broadcaster P_j . Since validation is deterministic, all honest parties derive the same valid dealer set

$$I = \left\{ j \in [N] : \widetilde{M}_j^z \text{ is valid} \right\}.$$

If P_j is honest, validity of BB, completeness of Π_{DL} and of the SCRAPE test, and correctness of the pairing checks imply that $\widetilde{M}_j^z$ is accepted except with negligible probability. Hence, $\mathcal{H} \subseteq I$, and therefore

$$\text{wt}(I) \geq \text{wt}(\mathcal{H}) \geq W - t \geq t + 1.$$

Thus, no honest party aborts after Round 2.

Well-formedness of the round 2 contributions. Fix any $j \in I$. By soundness of the SCRAPE test, there exists a polynomial $f_j(X) \in \mathbb{F}_p[X]$ of degree at most t such that

$$Z_{j,0} = [f_j(0)]_1 \quad \text{and} \quad Z_{j,\omega} = [f_j(\omega)]_1 \quad (\omega \in \Omega_J). \quad (8)$$

Define $z^{(j)} := f_j(0)$. The constant term consistency check then gives

$$Z_{j,T} = e(Z_{j,0}, [1]_2) = [z^{(j)}]_T. \quad (9)$$

For every $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$, write $D_{j,\omega,d} = [b_{j,\omega,d}]_2$ for the unique exponent $b_{j,\omega,d} \in \mathbb{F}_p$. The pairing check $e(Z_{j,\omega}, Q_{\text{own}(\omega),d}) = e([1]_1, D_{j,\omega,d})$, together with Equations (7) and (8), implies $[f_j(\omega)q_{\text{own}(\omega)}^d]_T = [b_{j,\omega,d}]_T$. By the non-degeneracy of the pairing,

$$b_{j,\omega,d} = f_j(\omega)q_{\text{own}(\omega)}^d. \quad (10)$$

Consequently, $D_{j,\omega,d} = [f_j(\omega)q_{\text{own}(\omega)}^d]_2$.

The aggregate polynomial. Define

$$Z(X) := \sum_{j \in I} f_j(X), \quad z := Z(0) = \sum_{j \in I} z^{(j)}, \quad z_\omega := Z(\omega) \quad (\omega \in \Omega_J).$$

Since every f_j has degree at most t , $\deg(Z) \leq t$.

By componentwise aggregation,

$$Z_0 = \sum_{j \in I} Z_{j,0} = [\sum_{j \in I} f_j(0)]_1 = [z]_1, \quad (11)$$

$$Z_\omega = \sum_{j \in I} Z_{j,\omega} = [\sum_{j \in I} f_j(\omega)]_1 = [z_\omega]_1, \quad (12)$$

$$Z_T = \sum_{j \in I} Z_{j,T} = [\sum_{j \in I} z^{(j)}]_T = [z]_T, \quad (13)$$

$$D_{\omega,d} = \sum_{j \in I} D_{j,\omega,d} = \left[\left(\sum_{j \in I} f_j(\omega) \right) q_{\text{own}(\omega)}^d \right]_2 = [z_\omega q_{\text{own}(\omega)}^d]_2. \quad (14)$$

C1. Let $S \subseteq \Omega_J$ be any set of $t+1$ distinct evaluation points. Since $\deg(Z) \leq t$, Lagrange interpolation gives

$$Z(0) = \sum_{\omega \in S} \lambda_{S,\omega} Z(\omega).$$

Using Equations (11) and (12), we obtain

$$\sum_{\omega \in S} \lambda_{S,\omega} Z_\omega = [\sum_{\omega \in S} \lambda_{S,\omega} Z(\omega)]_1 = [Z(0)]_1 = Z_0 = [z]_1.$$

The result is independent of S . Moreover, $t+1$ distinct evaluations uniquely determine a polynomial of degree at most t . Therefore, all subsets of $t+1$ share commitments in the public transcript define the same unique master value. This concludes the proof of C1.

C2. All honest parties derive the same set I and aggregate the same sender-indexed round 2 transcripts. Hence, they compute the same Z_T . By Equations (11) and (13),

$$\text{ek} = Z_T = [z]_T = e([z]_1, [1]_2) = e(Z_0, [1]_2),$$

where z is the unique master value guaranteed by C1. This proves C2.

C3. Round 1 agreement and deterministic validation ensure that all honest parties derive the same qualified set J , and the argument above shows that $\mathcal{H} \subseteq J$. For every $j \in J$, the scalar q_j defined by $\bar{Q}_j = [q_j]_1$ is unique and nonzero. Equation (7) therefore implies that, for every $k \in [\ell]$, $\text{vk}_{j,k} = Q_{j,k} = [q_j^k]_2$.

Round 2 agreement and deterministic validation ensure that all honest parties derive the same set I and aggregate the same values. Thus, all honest parties compute the same

$$Z_0, \quad (Z_\omega)_{\omega \in \Omega_J}, \quad \text{ek}, \quad (\text{vk}_j)_{j \in J}, \quad \text{dk}.$$

Equation (14) further gives $\text{dk}_{\omega,d} = [z_\omega q_{\text{own}(\omega)}^d]_2$ for every $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$. Therefore, every honest party outputs the same public transcript

$$\text{trx} = (J, Z_0, (Z_\omega)_{\omega \in \Omega_J}, \text{ek}, (\text{vk}_j)_{j \in J}, \text{dk}),$$

and every component has the form required by Definition 12. Every honest party $P_j \in J$ additionally retains the value q_j that it sampled in Round 1 as its secret key sk_j . This proves C3. $\square$

4.3.1 Security Proof We now prove security of the WBTX scheme in the sense of Definition 13, i.e., when the game runs our DKG protocol with the adversary $\mathcal{A}$ to generate the keys.

Assumption 5 (DKG-augmented weighted decisional power Diffie–Hellman). *Given parameters $(N, W, (w_j)_{j \in [N]}, (\Omega_j)_{j \in [N]}, t, \ell)$, define the interactive experiment $\text{DKG-WDdP}_{\mathcal{B}}^b(1^\lambda)$ as follows for $b \in \{0, 1\}$.*

The challenger samples the description of the prime-order asymmetric bilinear groups and sends the group description, the weights, the evaluation domains $(\Omega_j)_{j \in [N]}$, the threshold, and ℓ to the stateful distinguisher $\mathcal{B}$. $\mathcal{B}$ returns a static corruption set $\mathcal{C} \subseteq [N]$. The experiment returns 0 if $\text{wt}(\mathcal{C}) > t$; otherwise, it sets $\mathcal{H} := [N] \setminus \mathcal{C}$.

For every $j \in \mathcal{H}$, the challenger samples $q_j \leftarrow_{\$} \mathbb{F}_p^*$ and gives $\mathcal{B}$ the honest round 1 algebraic transcript

$$\bar{Q}_j = [q_j]_1, \quad Q_{j,d} = [q_j^d]_2 \quad (d \in \mathcal{E}_\ell).$$

After seeing all these values, $\mathcal{B}$ outputs a set $J_{\mathcal{C}} \subseteq \mathcal{C}$ along with a scalar $q_j \in \mathbb{F}_p^*$ for every $j \in J_{\mathcal{C}}$. The challenger then computes

$$\bar{Q}_j := [q_j]_1, \quad Q_{j,d} := [q_j^d]_2 \quad (d \in \mathcal{E}_\ell)$$

for every $j \in J_{\mathcal{C}}$, and defines $J := \mathcal{H} \cup J_{\mathcal{C}}$.

Fix $j^* := \min \mathcal{H}$. The challenger samples a uniformly random polynomial $f^*(X) \in \mathbb{F}_p[X]$ of degree at most t and sets $a := f^*(0)$. It computes the algebraic round 2 contribution

$$\text{trx}^* := \left(Z_0^*, Z_T^*, (Z_\omega^*)_{\omega \in \Omega_J}, (D_{\omega,d}^*)_{\substack{\omega \in \Omega_J \\ d \in \mathcal{D}_\ell}} \right),$$

where

$$\begin{aligned} Z_0^* &= [a]_1, & Z_T^* &= [a]_T, \\ Z_\omega^* &= [f^*(\omega)]_1, & D_{\omega,d}^* &= f^*(\omega)Q_{\text{own}(\omega),d} = [f^*(\omega)q_{\text{own}(\omega)}^d]_2. \end{aligned}$$

It also computes the simulation handles

$$H_{j,k} := [q_j^k]_1 \quad (j \in \mathcal{H}, k \in [\ell]).$$

Finally, the challenger samples $r, u \leftarrow \mathbb{F}_p$ and sets

$$X := [r]_1, \quad Y_0 := [ra]_T, \quad Y_1 := [u]_T.$$

It sends

$$\left(j^*, \text{tr}x^*, (H_{j,k})_{\substack{j \in \mathcal{H}, \\ k \in [\ell]}}, X, Y_b \right)$$

to $\mathcal{B}$. The experiment returns the bit output by $\mathcal{B}$. Define

$$\text{Adv}_{\mathcal{B}}^{\text{DKG-WDdP-DH}}(\lambda) = \left| \Pr[\text{DKG-WDdP}_{\mathcal{B}}^0(1^\lambda) = 1] - \Pr[\text{DKG-WDdP}_{\mathcal{B}}^1(1^\lambda) = 1] \right|.$$

The DKG-WDdP-DH assumption states that this advantage is negligible for every PPT distinguisher $\mathcal{B}$ and every polynomially bounded choice of valid parameters.

As before, the elements $(H_{j,k})_{j \in \mathcal{H}, k \in [\ell]}$ are used by the simulation and are not part of the public DKG transcript. Although the experiment is interactive, its challenger is efficient, so the assumption remains falsifiable. We analyze its hardness in the generic bilinear group model in Lemma 2 below.

Theorem 6 (Static WB-IND-CCA security with the synchronous DKG). *Suppose Assumption 5 holds, Byzantine broadcast satisfies Definition 10, and the noninteractive SCRAPE test is sound. Suppose also that Π_{DL} is complete, zero knowledge, and straightline simulation-extractable for the relations R^q , R^z , and R_{DL} , with respect to the same simulator (Section 3.4). Then, Construction 1, with keys generated by the DKG protocol of Constructions 2–4, is statically WB-IND-CCA secure as in Definition 13.*

Proof. We follow the proof of Theorem 3, but use Assumption 5 in the last step. Let $\mathcal{A}$ be an adversary in the game of Definition 13, and write S_k for the event that Game k outputs 1.

Game 0. This is the real game in Definition 13: the adversary receives the global public parameters, fixes its corruption set $\mathcal{C}$ before the DKG begins, and then participates in DKG as a rushing adversary controlling the corrupted parties. By Theorem 4, except with negligible probability, no honest party aborts and all honest parties agree on the transcript, so the abort clause of Definition 13 does not trigger. Hence

$$\left| \Pr[S_0] - \frac{1}{2} \right| = \text{Adv}_{\text{WB-TX}, \Pi_{\text{DKG}}, \mathcal{A}}^{\text{WB-IND-CCA-DKG}}(\lambda).$$

Game 1. Replace the NIZK CRS by

$$(\text{crs}, \text{td}_{\text{sim}}, \text{td}_{\text{ext}}) \leftarrow \Pi_{\text{DL}}.\text{SimSetup}(1^\lambda).$$

Generate every proof sent on behalf of an honest party in either DKG round, as well as the proof in the challenge ciphertext, using $\Pi_{\text{DL}}.\text{SimProve}$. By the zero-knowledge property, there exists a negligible function ϵ_{zk} such that

$$|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_{\text{zk}}.$$

Game 2. For every accepted corrupt round 1 transcript, extract a scalar q_j from the NIZK proof attached to $\overline{Q}_j$. The pairing chain then implies

$$Q_{j,d} = [q_j^d]_2 \quad (d \in \mathcal{E}_\ell).$$

For every accepted corrupt round 2 transcript, extract a scalar $z^{(j)}$ from the proof attached to $Z_{j,0}$. The target-group consistency check then implies

$$Z_{j,T} = [z^{(j)}]_T.$$

Abort if an extraction fails or returns an invalid witness.

We also change the implementation of the pre-decryption oracle. Consider a query $(\text{ct}_i)_{i \in K}$, where $K \subseteq [\ell]$ and

$$\text{ct}_i = (\text{ct}_i[1], \text{ct}_i[2], \pi_i) \quad \forall i \in K.$$

The oracle first performs the prescribed verification and returns $\perp$ if any proof is invalid. For every accepting ciphertext, it extracts

$$r_i \leftarrow \Pi_{\text{DL}}.\text{Ext}(\text{crs}, \text{td}_{\text{ext}}, (\text{ct}_i[1], \text{ct}_i[2]), \pi_i)$$

and returns, for every $j \in J \setminus \mathcal{C}$:

$$\widehat{\text{sbk}}_j := \sum_{i \in K} r_i H_{j,i}, \quad H_{j,i} := [q_j^i]_1.$$

Recall that J is the set of parties whose round 1 transcripts were valid. By the validity of the Byzantine broadcast and completeness of the round 1 checks, every honest party belongs to J . Hence $J \setminus \mathcal{C} = \mathcal{H}$.

Note that every DKG statement records the identity of its broadcaster, and validation rejects a transcript whose statement records a different identity than the broadcasting instance. Then, since honest proofs are simulated only for statements recording honest identities, every accepted corrupt DKG statement–proof pair lies outside the simulator’s output set. Moreover, whenever $\text{ct}_i[1] = [r_i]_1$ for every $i \in K$, the sbk_j returned above is exactly the honest pre-decryption output, as

$$\widehat{\text{sbk}}_j = \sum_{i \in K} r_i [q_j^i]_1 = \sum_{i \in K} [r_i q_j^i]_1 = \sum_{i \in K} q_j^i \text{ct}_i[1],$$

the game rejects a query containing the exact challenge ciphertext, and every other accepting queried ciphertext yields a statement–proof pair outside the simulator’s output set (a reused challenge statement must carry a fresh proof). Therefore, by straightline simulation-extractability and a union bound over all accepted corrupt DKG proofs and all queried ciphertexts, there is a negligible function ϵ_{ext} such that

$$|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_{\text{ext}}.$$

Game 3. Replace the challenge pad by an independent uniform element of $\mathbb{G}_T$. We reduce this change to Assumption 5 using a distinguisher $\mathcal{B}$.

The distinguisher receives the group description and the remaining algebraic public parameters from the assumption challenger. It generates

$$(\text{crs}, \text{td}_{\text{sim}}, \text{td}_{\text{ext}}) \leftarrow \Pi_{\text{DL}}.\text{SimSetup}(1^\lambda)$$

and gives the resulting public parameters and the NIZK CRS to $\mathcal{A}$. When $\mathcal{A}$ outputs its static corruption set $\mathcal{C}$, the distinguisher forwards $\mathcal{C}$ to the assumption challenger. Let $\mathcal{H} := [N] \setminus \mathcal{C}$. The parameter restrictions imply that $\mathcal{H} \neq \emptyset$. Select an honest dealer deterministically as $j^* := \min \mathcal{H}$.

Simulation of Round 1. For each $j \in \mathcal{H}$, the assumption challenger supplies

$$\overline{Q}_j = [q_j]_1, \quad Q_{j,d} = [q_j^d]_2 \quad (d \in \mathcal{E}_\ell).$$

The distinguisher attaches a simulated proof to each honest round 1 transcript and delivers the resulting Byzantine broadcast outputs to $\mathcal{A}$.

After seeing the honest round 1 transcripts, $\mathcal{A}$ returns the round 1 broadcasts for the corrupt parties. The distinguisher performs the prescribed deterministic validation. For every accepted corrupt transcript, it extracts q_j and sends all these scalars to the assumption challenger. Let $J_{\mathcal{C}} := J \cap \mathcal{C}$ be the set of corrupt parties with valid transcripts. Since every honest round 1 transcript is accepted, $J = \mathcal{H} \cup J_{\mathcal{C}}$. Combining this with the fact that the extraction fails only with negligible probability by simulation-extractability of the NIZK, the assumption challenger and the simulated DKG use the same set J and the same evaluation domain Ω_J .

Simulation of Round 2. The assumption challenger supplies the algebraic round 2 contribution of P_{j^*} :

$$\left(Z_0^*, Z_T^*, (Z_\omega^*)_{\omega \in \Omega_J}, (D_{\omega,d}^*)_{\substack{\omega \in \Omega_J \\ d \in \mathcal{D}_\ell}} \right),$$

along with $(H_{j,k})_{j \in \mathcal{H}, k \in [\ell]}, X, Y_b$, where, for an implicit uniformly random degree $\leq t$ polynomial f^* with $a := f^*(0)$,

$$\begin{aligned} Z_0^* &= [a]_1, & Z_T^* &= [a]_T, \\ Z_\omega^* &= [f^*(\omega)]_1, & D_{\omega,d}^* &= f^*(\omega) Q_{\text{own}(\omega),d}. \end{aligned}$$

The distinguisher uses these values as the round 2 contribution of P_{j^*} and attaches a simulated proof.

For every other honest dealer $j \in \mathcal{H} \setminus \{j^*\}$, the distinguisher independently samples a uniformly random polynomial $f_j \in \mathbb{F}_p[X]$ of degree at most t and computes

$$\begin{aligned} Z_{j,0} &= [f_j(0)]_1, & Z_{j,T} &= [f_j(0)]_T, \\ Z_{j,\omega} &= [f_j(\omega)]_1, & D_{j,\omega,d} &= f_j(\omega) Q_{\text{own}(\omega),d} \end{aligned}$$

for every $\omega \in \Omega_J$ and $d \in \mathcal{D}_\ell$. It attaches simulated proofs and delivers all honest round 2 broadcasts to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs the round 2 transcripts for corrupt parties. Note that the adversary may choose these transcripts after seeing all honest contributions. $\mathcal{B}$ validates these broadcasts exactly

as in the real DKG and also extracts $z^{(j)}$ from every accepted corrupt transcript. Let $I_C := I \cap \mathcal{C}$ be the resulting set of accepted corrupt dealers, where I is the set of all parties with valid transcripts, and define

$$z_{\text{kn}} := \sum_{j \in \mathcal{H} \setminus \{j^*\}} f_j(0) + \sum_{j \in I_C} z^{(j)}.$$

Every honest dealer belongs to I . Moreover, target-group consistency gives $Z_{j,T} = [z^{(j)}]_T$ for every $j \in I_C$. Consequently, the aggregate encryption key satisfies

$$\text{ek} = Z_T = Z_T^* + [z_{\text{kn}}]_T = [a + z_{\text{kn}}]_T.$$

The distinguisher similarly obtains every other aggregate DKG value by adding party j^* 's group elements (provided by the assumption) to the honest transcripts it generated and the corrupt transcripts it received. Thus, $\mathcal{A}$ receives a complete DKG transcript, and the output is distributed exactly as in Game 2. Note that this remains true even though the corrupt contributions, and hence z_{kn} , may be arbitrary functions of the honest parties' round 2 transcripts.

Simulation handles and pre-decryption queries. Once the DKG execution is complete, $\mathcal{B}$ uses the $\{H_{j,i}\}_{j \in \mathcal{H}, i \in [\ell]}$ and the NIZK's trapdoor for extraction to answer all pre-decryption queries exactly as in Game 2.

Challenge ciphertext. When $\mathcal{A}$ submits challenge messages m_0, m_1 , the distinguisher samples $\mu \leftarrow \{0, 1\}$ and computes

$$\Delta_{\text{kn}} := e(z_{\text{kn}}X, [1]_2) = [rz_{\text{kn}}]_T$$

where $X = [r]_1$. It then sets

$$\tilde{Y}_b := Y_b + \Delta_{\text{kn}}.$$

It returns the challenge ciphertext

$$\text{ct}^* := \left(X, m_\mu + \tilde{Y}_b, \Pi_{\text{DL}}.\text{SimProve}(\text{crs}, \text{td}_{\text{sim}}, (X, m_\mu + \tilde{Y}_b)) \right).$$

The distinguisher continues to answer pre-decryption queries as in Game 2 and finally returns 1 if and only if $\mathcal{A}$ outputs μ . (If an extraction inherited from Game 2 fails, or if an honest party aborts in the simulated DKG, $\mathcal{B}$ outputs a uniform bit, matching the abort clause of Definition 13.)

If the assumption bit is $b = 0$, then

$$\begin{aligned} \tilde{Y}_0 &= [ra]_T + [rz_{\text{kn}}]_T \\ &= [r(a + z_{\text{kn}})]_T, \end{aligned}$$

which is precisely the encryption pad under the final DKG key $\text{ek} = [a + z_{\text{kn}}]_T$. Thus, the simulation is Game 2.

If the assumption bit is $b = 1$, then Y_1 is uniform in $\mathbb{G}_T$. Translation by Δ_{kn} preserves uniformity, even conditioned on the complete DKG transcript and even when z_{kn} depends on the dealer's transcript. Thus, the simulation is Game 3. It follows that

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{B}}^{\text{DKG-WDdP-DH}}(\lambda).$$

In Game 3, the challenge ciphertext is independent of μ , and therefore $\Pr[S_3] = 1/2$. This implies

$$\text{Adv}_{\text{WB-TX}, \Pi_{\text{DKG}}, \mathcal{A}}^{\text{WB-IND-CCA-DKG}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{DKG-WDdP-DH}}(\lambda) + \epsilon_{\text{zk}} + \epsilon_{\text{ext}},$$

which is negligible. $\square$

Lemma 2 (Bilinear generic group hardness). *Assumption 5 holds in the bilinear generic group model except with probability $O(Q^2(N+2)\ell/p)$, where Q denotes the number of generic group operations.*

Proof. Fix the corruption set $\mathcal{C}$ returned by $\mathcal{B}$ and the scalars $(q_j)_{j \in J_{\mathcal{C}}}$ it outputs after round 1; the latter are field elements known to the adversary. The symbolic argument below holds uniformly for every valid choice of $\mathcal{C}$ and of these scalars.

Let $f^*(X) = a + \sum_{s=1}^t a_s X^s$. In the symbolic game, we treat

$$r, a, a_1, \dots, a_t, (q_j)_{j \in \mathcal{H}}$$

as algebraically independent indeterminates. The exponents of the $\mathbb{G}_1$ encodings available to the adversary lie in the linear span of the Laurent polynomials

$$1, r, a, f^*(\omega), q_j^k \quad (\omega \in \Omega_J, j \in \mathcal{H}, k \in [\ell]),$$

and the exponents of the $\mathbb{G}_2$ encodings lie in the span of

$$1, q_j^d, f^*(\omega)q_{\text{own}(\omega)}^{d'} \quad (j \in \mathcal{H}, d \in \mathcal{E}_{\ell}, \omega \in \Omega_J, d' \in \mathcal{D}_{\ell}),$$

where $q_{\text{own}(\omega)}$ is a known constant whenever $\text{own}(\omega) \in J_{\mathcal{C}}$. The exponents of the $\mathbb{G}_T$ encodings are 1, a , and the challenge exponent. The indeterminate r occurs only in the challenge element $X = [r]_1$. Hence, every $\mathbb{G}_T$ expression which contains r has an r -coefficient in the linear span of the $\mathbb{G}_2$ exponents above, and producing ra requires a to lie in that span.

Now, consider a party $j \in \mathcal{H}$. Note that each term q_j^d and $f^*(\omega)q_j^{d'}$ with $\text{own}(\omega) = j$ carries a nonzero power of the independent indeterminate q_j , while a carries none. Comparing the coefficients shows that any combination of these terms which is independent of q_j contributes zero. On the other hand, the terms whose evaluation points are owned by the parties in $J_{\mathcal{C}}$ reduce to the span of $(f^*(\omega))_{\omega \in \Omega_{J_{\mathcal{C}}}}$. Thus, producing ra requires a to lie in the linear span of 1 and $(f^*(\omega))_{\omega \in \Omega_{J_{\mathcal{C}}}}$. Since $|\Omega_{J_{\mathcal{C}}}| = \text{wt}(J_{\mathcal{C}}) \leq \text{wt}(\mathcal{C}) \leq t$ and the evaluation points are distinct and nonzero, the Vandermonde argument in the proof of Lemma 1, applied to the coefficients $a_1, \dots, a_t$ of f^* , shows that this is impossible.

Finally, the adversary can distinguish the symbolic and real games only if two distinct formal expressions in the symbolic game receive the same value in the real game. Let

$$D := \prod_{j \in \mathcal{H}} q_j^{\ell} \quad \text{and} \quad M := |\mathcal{H}| \ell,$$

so that multiplication by D clears every negative power of an honest party's q_j . The exponents represented in $\mathbb{G}_1$ have degree at most ℓ , while the cleared numerators of the exponents represented in $\mathbb{G}_2$ have degree at most $M + \ell$. Consequently, a pairing yields degree at most $M + 2\ell \leq (N + 2)\ell$. Hence, by the Schwartz–Zippel lemma and a union bound over at most $O(Q^2)$ pairs of expressions, after clearing the denominators of the Laurent polynomials, the adversary can distinguish the symbolic and real games with probability at most $O(Q^2(N + 2)\ell/p)$, which gives the stated bound. $\square$

5 Partial-Fraction-Based Weighted BTE

In this section, we present our second scheme, WPFE (specified in Construction 5): a weighted batch threshold encryption scheme based on ideas from [11,9,2]. For $a \in \mathbb{F}_p$, define $p_a(X) := 1/(X + a)$. We fix distinct public $\alpha_1, \dots, \alpha_\ell \in \mathbb{F}_p \setminus \{0, -1\}$. Decryption uses the partial fraction identity

$$g_{j,k}g_{j,i} = \frac{g_{j,i} - g_{j,k}}{\alpha_k - \alpha_i} \quad (k \neq i). \quad (15)$$

We can again accelerate the decryption algorithm using multi-pairings and the Fast Fourier Transform (FFT). The first terms in $\sum_{j \in T'} L_{j,i}$ can be evaluated as one multi-pairing of arity $|T'|$ for each $i \in [\ell]$. For the remaining terms,

$$\sum_{j \in T'} \sum_{\substack{k \in [\ell] \\ k \neq i}} e \left(\frac{\text{ct}_k[1]}{\alpha_k - \alpha_i}, \sum_{w \in \Omega_j} \lambda_{T',w} \mathbf{dk}_{w,k}[1] \right) = \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{E_k}{\alpha_k - \alpha_i},$$

where

$$E_k := e \left(\text{ct}_k[1], \sum_{w \in \Omega_{T'}} \lambda_{T',w} \mathbf{dk}_{w,k}[1] \right).$$

Each E_k is independent of i and is therefore computed once and reused for every output ciphertext. This requires ℓ ordinary pairings. Moreover, the two remaining pairings with left input $\text{ct}_i[1]$ can be combined as

$$(\alpha_i + 1)e(\text{ct}_i[1], D_i) - e(\text{ct}_i[1], \mathbf{dk}[0]) = e(\text{ct}_i[1], (\alpha_i + 1)D_i - \mathbf{dk}[0]),$$

where $D_i := \sum_{w \in \Omega_{T'}} \lambda_{T',w} \mathbf{dk}_{w,i}[2]$. Consequently, decryption requires 2ℓ pairings in addition to the ℓ multi-pairings of size $|T'|$, implying $|T'| + 2$ pairing inputs per ciphertext.

The remaining quadratic work in decryption consists of Cauchy sums of the form

$$\mathcal{C}_i(X) := \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{X_k}{\alpha_k - \alpha_i},$$

where $(X_k)_k$ is either $(\text{ct}_k[1])_k \subseteq \mathbb{G}_1$ or the cached sequence $(E_k)_k \subseteq \mathbb{G}_T$. Choosing the public indices as $\alpha_i = \gamma \nu^{i-1}$, where ν is a primitive ℓ -th root of unity (assuming $\ell \mid p-1$) and γ is chosen so that every $\alpha_i \notin \{0, -1\}$, we can write $\mathcal{C}_i(X)$ as

$$\mathcal{C}_i(X) = \gamma^{-1} \nu^{1-i} \sum_{d \neq 0} \frac{X_{i+d}}{\nu^d - 1},$$

with indices interpreted modulo ℓ . Thus, the values $(\mathcal{C}_i(X))_{i \in [\ell]}$ form a cyclic convolution and can be computed using an FFT in $O(\ell \log \ell)$ group operations. To see the computation explicitly, define the scalar kernel $h_0 := 0$ and $h_d := (\nu^d - 1)^{-1}$ for $d \neq 0$. Up to the public factor $\gamma^{-1} \nu^{1-i}$, the vector $(\mathcal{C}_i(X))_{i \in [\ell]}$ is the cyclic convolution of $(X_i)_i$ with $(h_d)_d$ after reversing the kernel. It is therefore obtained by taking a group FFT of $(X_i)_i$, multiplying its coordinates by the ordinary field FFT of the fixed kernel, and applying an inverse group FFT. The procedure is run once with

Setup($1^\lambda, 1^\ell, (\Omega_j)_{j \in [N]}, 1^t, 1^W$) $\rightarrow$ (ek, sk, vk, dk):

- ▷ $\text{crs}_{\Pi_{\text{DL}}} \leftarrow \Pi_{\text{DL}}.\text{Setup}(1^\lambda)$.
- ▷ Sample $z \leftarrow \mathbb{F}_p$ and a uniformly random polynomial $Z(X) \in \mathbb{F}_p[X]$ of degree at most t such that $Z(0) = z$. Set $z_w := Z(w)$ for $w \in \Omega_{[N]}$.
- ▷ Fix distinct public indices $\alpha_1, \dots, \alpha_\ell \in \mathbb{F}_p \setminus \{0, -1\}$. These values are included in the public parameters. Sample $\beta_{-1}, \beta_0 \leftarrow \mathbb{F}_p$ subject to $\beta_0 \neq 0$ and $\beta_{-1} + \beta_0 \neq 0$. For every $j \in [N]$, sample a hidden shift ρ_j such that every $\rho_j + \alpha_i$ is nonzero and the values $\rho_j + \alpha_i$, over $(j, i) \in [N] \times [\ell]$, are distinct. Define

$$g_{j,i} := \frac{1}{\rho_j + \alpha_i}, \quad u_{j,i} := g_{j,i}^2 + \frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i}$$

for every $j \in [N]$ and $i \in [\ell]$.

- ▷ Let $\text{ek} := [z\beta_0]_T$, and let $\text{sk}_j := \rho_j$ and $\text{vk}_j := (\text{vk}_{j,i} := [g_{j,i}]_2)_{i \in [\ell]}$ denote party j 's secret and verification keys.
- ▷ Define $\text{dk}[0] := [z(\beta_{-1} + \beta_0)]_2$, and, for every $w \in \Omega_{[N]}$ and $i \in [\ell]$, define

$$\text{dk}_{w,i}[1] := [z_w g_{\text{own}(w),i}]_2, \quad \text{dk}_{w,i}[2] := [z_w u_{\text{own}(w),i}]_2.$$

- ▷ **Output**

$$\left(\text{ek}, (\text{sk}_j)_{j \in [N]}, (\text{vk}_j)_{j \in [N]}, \text{dk} := \left(\text{dk}[0], (\text{dk}_{w,i}[1], \text{dk}_{w,i}[2])_{w \in \Omega_{[N]}, i \in [\ell]} \right) \right).$$

Enc(ek, m) $\rightarrow$ ct:

- ▷ Given $m \in \mathbb{G}_T$, sample $r \leftarrow \mathbb{F}_p$, and set $\text{ct}[1] := [r]_1$ and $\text{ct}[2] := m + r \cdot \text{ek}$.
- ▷ Obtain $\pi \leftarrow \Pi_{\text{DL}}.\text{Prove}((\text{ct}[1], \text{ct}[2]); r)$ and output $\text{ct} := (\text{ct}[1], \text{ct}[2], \pi)$.

PreDec($\text{sk}_j, (\text{ct}_i)_{i \in [\ell]}$) $\rightarrow$ sbk_j :

- ▷ Parse $\text{ct}_i = (\text{ct}_i[1], \text{ct}_i[2], \pi_i)$ for every $i \in [\ell]$. If $\Pi_{\text{DL}}.\text{Verify}((\text{ct}_i[1], \text{ct}_i[2]), \pi_i) = 0$ for any $i \in [\ell]$, return $\perp$.
- ▷ Otherwise, return $\text{sbk}_j := \sum_{i=1}^{\ell} \frac{1}{\rho_j + \alpha_i} \text{ct}_i[1] \in \mathbb{G}_1$.

PreDecVer($\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j$) $\rightarrow$ 0/1:

- ▷ Check $e(\text{sbk}_j, [1]_2) = \sum_{i=1}^{\ell} e(\text{ct}_i[1], \text{vk}_{j,i})$; if so, return 1, else return 0.

Dec($\text{dk}, \{\text{sbk}_j\}_{j \in T}, (\text{ct}_i)_{i \in [\ell]}$) $\rightarrow$ $(m_i)_{i \in [\ell]}$:

- ▷ Verify each submitted pre-decryption key by checking **PreDecVer**($\text{vk}_j, (\text{ct}_i)_{i \in [\ell]}, \text{sbk}_j$) for every $j \in T$. Discard the keys which do not pass verification. Let T' be an authorized set of accepted pre-decryption keys, and return $\perp$ if no such set remains.
- ▷ For every $j \in T'$ and $i \in [\ell]$, compute

$$L_{j,i} := e \left(\text{sbk}_j - \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{\text{ct}_k[1]}{\alpha_k - \alpha_i}, \sum_{w \in \Omega_j} \lambda_{T',w} \text{dk}_{w,i}[1] \right) \\ + \sum_{\substack{k \in [\ell] \\ k \neq i}} e \left(\frac{\text{ct}_k[1]}{\alpha_k - \alpha_i}, \sum_{w \in \Omega_j} \lambda_{T',w} \text{dk}_{w,k}[1] \right).$$

- ▷ For every $i \in [\ell]$, output $m_i := \text{ct}_i[2] - R_i$, where

$$R_i := \alpha_i \left((\alpha_i + 1) \left(e \left(\text{ct}_i[1], \sum_{w \in \Omega_{T'}} \lambda_{T',w} \text{dk}_{w,i}[2] \right) - \sum_{j \in T'} L_{j,i} \right) - e(\text{ct}_i[1], \text{dk}[0]) \right).$$

Construction 5: Partial-fraction-based weighted BTE construction

$X_i = \text{ct}_i[1] \in \mathbb{G}_1$ and once with $X_i = E_i \in \mathbb{G}_T$. This reduces the non-pairing work of decryption from $O(\ell^2)$ to $O(\ell \log \ell)$ without changing the pairing complexity.

As discussed in Remark 1, this construction also supports decryption when ciphertext batches are subsets of each other. We can rewrite $L_{j,i}$ by replacing $[\ell]$ in its two inner sums with I_j and using $\text{ct}_{j,k}[1]$. Equation (15) cancels the additional ciphertexts in party j 's batch, leaving the same $g_{j,i}^2 r_i$ term as before; the remainder of the correctness proof is unchanged.

Finally, we note that the pre-decryption keys submitted by a set of parties T can also be batch verified by sampling $\delta_j \leftarrow \mathbb{F}_p$ for $j \in T$, and checking

$$e\left(\sum_{j \in T} \delta_j \text{sbk}_j, [1]_2\right) = \sum_{i=1}^{\ell} e\left(\text{ct}_i[1], \sum_{j \in T} \delta_j \text{vk}_{j,i}\right).$$

5.1 Correctness and robustness

Theorem 7 (Correctness and robustness). *Construction 5 is correct and satisfies robustness.*

Proof. Fix an authorized set T' of accepted pre-decryption keys and an index $i \in [\ell]$. Write $\text{ct}_k[1] = [r_k]_1$ and set

$$z_j := \sum_{w \in \Omega_j} \lambda_{T',w} z_w \quad (j \in T')$$

Since $\Omega_{T'}$ contains more than t evaluation points and Z has degree at most t , Lagrange interpolation gives

$$\sum_{j \in T'} z_j = \sum_{w \in \Omega_{T'}} \lambda_{T',w} z_w = z. \quad (16)$$

Now, $L_{j,i}$ is given by

$$\begin{aligned} & e\left(\text{sbk}_j - \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{\text{ct}_k[1]}{\alpha_k - \alpha_i}, \sum_{w \in \Omega_j} \lambda_{T',w} \text{dk}_{w,i}[1]\right) \\ & + \sum_{\substack{k \in [\ell] \\ k \neq i}} e\left(\frac{\text{ct}_k[1]}{\alpha_k - \alpha_i}, \sum_{w \in \Omega_j} \lambda_{T',w} \text{dk}_{w,k}[1]\right) \\ & = \left[z_j g_{j,i} \left(\sum_{k=1}^{\ell} r_k g_{j,k} - \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{r_k}{\alpha_k - \alpha_i} \right) + z_j \sum_{\substack{k \in [\ell] \\ k \neq i}} \frac{r_k g_{j,k}}{\alpha_k - \alpha_i} \right]_T \end{aligned}$$

Observe that the coefficient of $z_j r_k$ for $k \neq i$ is

$$g_{j,i} g_{j,k} - \frac{g_{j,i}}{\alpha_k - \alpha_i} + \frac{g_{j,k}}{\alpha_k - \alpha_i} = 0$$

by the partial fraction identity at Equation (15). Consequently,

$$L_{j,i} = [r_i z_j g_{j,i}^2]_T = \sum_{w \in \Omega_j} [r_i \lambda_{T',w} z_w g_{j,i}^2]_T. \quad (17)$$

Finally, using the definition of $\mathbf{dk}_{w,i}[2]$, partitioning the points in $\Omega_{T'}$ by owner, and using Equation (16), we obtain

$$e\left(\mathbf{ct}_i[1], \sum_{w \in \Omega_{T'}} \lambda_{T',w} \mathbf{dk}_{w,i}[2]\right) = \left[r_i z \left(\frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i}\right)\right]_T + \sum_{j \in T'} [r_i z_j g_{j,i}^2]_T.$$

Therefore, subtracting $\sum_{j \in T'} L_{j,i}$ from the term above, we derive the pad

$$\alpha_i \left((\alpha_i + 1) \left[r_i z \left(\frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i} \right) \right]_T - [r_i z(\beta_{-1} + \beta_0)]_T \right) = [r_i z \beta_0]_T,$$

which proves correctness.

It remains to prove robustness. To show accurate blaming, observe that a correctly generated pre-decryption key always passes its verification check:

$$e(\mathbf{sbk}_j, [1]_2) = e\left(\sum_{i=1}^{\ell} g_{j,i} \mathbf{ct}_i[1], [1]_2\right) = \sum_{i=1}^{\ell} e(\mathbf{ct}_i[1], [g_{j,i}]_2) = \sum_{i=1}^{\ell} e(\mathbf{ct}_i[1], \mathbf{vk}_{j,i}) \quad (18)$$

For consistency, observe that since $x \mapsto e(x, [1]_2)$ is injective on the prime-order group $\mathbb{G}_1$, any $\mathbf{sbk}_j$ accepted by the verification check must equal the prescribed value $\sum_{i=1}^{\ell} g_{j,i} \cdot \mathbf{ct}_i[1]$. Writing $\mathbf{ct}_i[1] = [r_i]_1$ for the (possibly adversarial) discrete logarithms r_i , the correctness computation above then shows that, for *every* authorized set T' of accepted keys, $R_i = [r_i z \beta_0]_T$, independently of T' . Hence any two authorized sets of accepted pre-decryption keys for the same batch decrypt it to the same messages. Accurate blaming and consistency together imply the robustness of Construction 5. $\square$

5.2 Analysis of Construction 5

We use the following weighted analog of the q -QuadRankDef assumption [11, Assumption 2] underlying the original partial-fraction batch threshold encryption scheme of [11].

Assumption 8 (Weighted Quadratic Rank Deficiency, WQuadRankDef). *Given parameters $(N, W, (w_j)_{j \in [N]}, (\Omega_j)_{j \in [N]}, t, \ell, (\alpha_i)_{i \in [\ell]})$, define the two-stage experiment $\text{WQRD}_{\mathcal{B}}^b(1^\lambda)$ for $b \in \{0, 1\}$ and distinguisher $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, as follows. The challenger samples the bilinear groups, $z, a_1, \dots, a_t, \beta_{-1}, \beta_0 \leftarrow_{\$} \mathbb{F}_p$, subject to $\beta_0 \neq 0$ and $\beta_{-1} + \beta_0 \neq 0$, and sets*

$$Z(X) = z + \sum_{s=1}^t a_s X^s, \quad z_w = Z(w).$$

For each $j \in [N]$, it samples $\rho_j \leftarrow_{\$} \mathbb{F}_p$ subject to the conditions in Construction 5, and defines

$$g_{j,i} = \frac{1}{\rho_j + \alpha_i}, \quad u_{j,i} = g_{j,i}^2 + \frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i}.$$

It computes

$$\begin{aligned} \text{ek} &= [z\beta_0]_T, \\ \text{vk}_{j,i} &= [g_{j,i}]_2, \\ \text{dk}[0] &= [z(\beta_{-1} + \beta_0)]_2, \\ \text{dk}_{w,i}[1] &= [z_w g_{\text{own}(w),i}]_2, \\ \text{dk}_{w,i}[2] &= [z_w u_{\text{own}(w),i}]_2, \\ H_{j,i} &= [g_{j,i}]_1. \end{aligned}$$

Let pub contain $(\Omega_j)_{j \in [N]}$, ek , all verification keys, and all public decryption key elements. In the first stage,

$$(\mathcal{C}, \text{st}) \leftarrow \mathcal{B}_1(\text{pub}, (H_{j,i})_{j \in [N], i \in [\ell]}).$$

The experiment returns 0 if $\text{wt}(\mathcal{C}) > t$. Otherwise, it samples $r, u \leftarrow \$ \mathbb{F}_p$, sets

$$X = [r]_1, \quad Y_0 = [rz\beta_0]_T, \quad Y_1 = [u]_T,$$

sets $\text{sk}_j = \rho_j$ for every $j \in [N]$, and returns the output of

$$\mathcal{B}_2(\text{st}, (\text{sk}_j)_{j \in \mathcal{C}}, X, Y_b).$$

The advantage is

$$\text{Adv}_{\mathcal{B}}^{\text{WQuadRankDef}}(\lambda) = |\Pr[\text{WQRD}_{\mathcal{B}}^0(1^\lambda) = 1] - \Pr[\text{WQRD}_{\mathcal{B}}^1(1^\lambda) = 1]|.$$

The WQuadRankDef assumption states that this advantage is negligible for every PPT distinguisher and every polynomially bounded choice of valid parameters.

As before, the elements $(H_{j,i})_{j \in [N], i \in [\ell]}$ are used by the simulation and are given to the distinguishing adversary, but are not public parameters of Construction 5. The assumption is falsifiable, as both distributions in the experiment are efficiently samplable.

Theorem 9 (Static WB-IND-CCA Security). *Suppose Assumption 8 holds and Π_{DL} is complete, zero knowledge, and straightline simulation-extractable for R_{DL} , with respect to the same simulator (Section 3.4). Then, Construction 5 is WB-IND-CCA secure.*

Proof. Let $\mathcal{A}$ be the WB-IND-CCA adversary (Definition 3). We consider the following games, and write S_k for the event that $\mathcal{A}$ outputs the challenge bit in Game k .

Game 0. This is the real WB-IND-CCA game. Hence

$$\left| \Pr[S_0] - \frac{1}{2} \right| = \text{Adv}_{\text{WPFE}, \mathcal{A}}^{\text{WB-IND-CCA}}(\lambda).$$

Game 1. At setup, we replace the NIZK CRS by

$$(\text{crs}, \text{td}_{\text{sim}}, \text{td}_{\text{ext}}) \leftarrow \Pi_{\text{DL}}.\text{SimSetup}(1^\lambda),$$

and generate the proof within the challenge ciphertext using $\Pi_{\text{DL}}.\text{SimProve}$. Zero knowledge implies that there is a negligible function ϵ_{zk} such that

$$|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_{\text{zk}}.$$

Game 2. We change the implementation of the pre-decryption oracle. Given a query $(\text{ct}_i)_{i \in I}$, where $\text{ct}_i = (\text{ct}_i[1], \text{ct}_i[2], \pi_i)$ and $I \subseteq [\ell]$, the oracle first performs the prescribed verification and returns $\perp$ if a proof is invalid. It then computes $r_i \leftarrow \Pi_{\text{DL}}.\text{Ext}(\text{crs}, \text{td}_{\text{ext}}, (\text{ct}_i[1], \text{ct}_i[2]), \pi_i)$ for every accepting ciphertext, and returns

$$\widehat{\text{sbk}}_j = \sum_{i \in I} r_i H_{j,i}$$

for every $j \notin \mathcal{C}$. Observe that if $\text{ct}_i[1] = [r_i]_1$ for all queried ciphertexts, then $\widehat{\text{sbk}}_j = \sum_{i \in I} g_{j,i} \text{ct}_i[1]$ is exactly the prescribed output of PreDec .

Before the challenge, the simulator has returned no statement–proof pair. After the challenge, the only such pair is the challenge pair $((\text{ct}^*[1], \text{ct}^*[2]), \pi^*)$. The WB-IND-CCA game rejects a query containing the exact challenge ciphertext. Hence, every pair on which the extractor is invoked is outside the simulator’s output set, including the pairs which use $(\text{ct}^*[1], \text{ct}^*[2])$ as the statement along with a fresh proof. Then, by the simulation-extractability of the NIZK and a union bound over all queried ciphertexts, the probability that $\text{ct}_i[1] \neq [r_i]_1$ for an extracted $r_i \leftarrow \Pi_{\text{DL}}.\text{Ext}(\text{crs}, \text{td}_{\text{ext}}, (\text{ct}_i[1], \text{ct}_i[2]), \pi_i)$ is bounded by a negligible function ϵ_{ext} . Therefore, it follows that

$$|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_{\text{ext}}.$$

Game 3. In Game 3, we replace the challenge pad $[r^* z \beta_0]_T$ by an independent uniformly random element. A distinguisher $\mathcal{B}$ for Assumption 8 simulates Game 2 using the handles $(H_{j,i})_{j \in [N], i \in [\ell]}$. Given $(\text{pub}, (H_{j,i})_{j,i})$, it generates the simulated NIZK CRS and passes pub and the CRS to $\mathcal{A}$; it answers pre-decryption queries, both before and after the challenge, as in Game 2. It forwards the corrupted set $\mathcal{C}$ chosen by $\mathcal{A}$ and gives the returned secret keys $(\rho_j)_{j \in \mathcal{C}}$ to $\mathcal{A}$. On challenge messages m_0, m_1 , it samples $\mu \leftarrow \$ \{0, 1\}$, and using its assumption challenge (X, Y_b) , returns

$$\text{ct}^* = (X, m_\mu + Y_b, \Pi_{\text{DL}}.\text{SimProve}(\text{crs}, \text{td}_{\text{sim}}, (X, m_\mu + Y_b))).$$

It outputs 1 if and only if $\mathcal{A}$ outputs μ . Now, if $b = 0$, the simulation of $\mathcal{B}$ is distributed identically to Game 2, and if $b = 1$, it is distributed identically to Game 3. Consequently,

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{B}}^{\text{WQuadRankDef}}(\lambda).$$

In Game 3, the second component of the challenge ciphertext is uniform and independent of μ , so $\Pr[S_3] = 1/2$. Combining the games gives

$$\text{Adv}_{\text{WPFE}, \mathcal{A}}^{\text{WB-IND-CCA}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{WQuadRankDef}}(\lambda) + \epsilon_{\text{zk}} + \epsilon_{\text{ext}},$$

which is negligible. $\square$

Lemma 3 (Bilinear generic group hardness). *Assumption 8 holds in the generic bilinear group model, except with probability $O(Q^2(3N\ell+3)/p)$, where Q denotes the total number of generic group handles occurring in the experiment.*

Proof. Fix the set $\mathcal{C} \subseteq [N]$ returned in the first stage and reveal $(\rho_j)_{j \in \mathcal{C}}$ to the adversary. In the symbolic game, we treat

$$r, z, a_1, \dots, a_t, \beta_{-1}, \beta_0, (\rho_j)_{j \notin \mathcal{C}}$$

as algebraically independent indeterminates, and the challenge exponent in the $b = 1$ branch is a fresh indeterminate. Any formal $\mathbb{G}_T$ expression that contains r has the form $rL + L'$, where L lies

in the linear span of the public $\mathbb{G}_2$ exponents, and L' does not contain r : indeed, r occurs, with degree one, only in the exponent of $X = [r]_1$, pairing a handle $H_{j,i}$ or $[1]_1$ with public $\mathbb{G}_2$ material yields $\mathbb{G}_T$ terms without r , and $\mathbb{G}_T$ elements cannot be paired again. It is therefore enough to prove that $z\beta_0$ is not in the linear span of the rational functions

$$1, \quad g_{j,i} := \frac{1}{\rho_j + \alpha_i}, \quad z(\beta_{-1} + \beta_0), \\ Z(w)g_{\text{own}(w),i}, \\ Z(w) \left(g_{\text{own}(w),i}^2 + \frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i} \right),$$

over $j \in [N]$, $i \in [\ell]$, and $w \in \Omega_{[N]}$.

Consider a party $j \notin \mathcal{C}$ and fix $i \in [\ell]$. The contribution of all terms $\mathbf{dk}_{w,i}[2]$ with $w \in \Omega_j$ has the form

$$\left(\sum_{w \in \Omega_j} c_{w,i} Z(w) \right) \left(g_{j,i}^2 + \frac{\beta_{-1}}{\alpha_i + 1} + \frac{\beta_0}{\alpha_i} \right)$$

for coefficients $c_{w,i} \in \mathbb{F}_p$. Expanding around the pole $\rho_j = -\alpha_i$, the coefficient of the order two pole term $g_{j,i}^2$ is

$$\sum_{w \in \Omega_j} c_{w,i} Z(w),$$

since all other span elements have at most a simple pole there. However, the target $z\beta_0$ has no pole in ρ_j , and the terms associated with indices $i' \neq i$ have poles $\rho_j = -\alpha_{i'}$ at different locations. The displayed coefficient must therefore vanish identically; since $u_{j,i}$ is the same for every $w \in \Omega_j$, the entire contribution of the $\mathbf{dk}_{w,i}[2]$ terms for $\text{own}(w) \notin \mathcal{C}$, including its β_{-1} and β_0 parts, is then zero.

After eliminating these terms, the remaining expressions having a pole at $\rho_j = -\alpha_i$ are of the form

$$g_{j,i} \left(c_i + \sum_{w \in \Omega_j} b_{w,i} Z(w) \right),$$

where $c_i, b_{w,i} \in \mathbb{F}_p$. Its coefficient must also vanish identically, making the contribution from linear combinations of the $\mathbf{dk}_{w,i}[1]$ and $[g_{j,i}]_2$ terms for $\text{own}(w) \notin \mathcal{C}$ zero. Consequently, terms belonging to uncorrupted parties cannot contribute to a pole-free expression for $z\beta_0$.

For every corrupted party $j \in \mathcal{C}$, the revealed secret key ρ_j determines all

$$g_{j,i} = (\rho_j + \alpha_i)^{-1}.$$

We may therefore grant the adversary the larger span

$$1, \quad Z(w), \quad Z(w)\beta_{-1}, \quad Z(w)\beta_0 \quad (w \in \Omega_{\mathcal{C}}),$$

in addition to $z(\beta_{-1} + \beta_0)$. For contradiction, suppose $z\beta_0$ belongs to this enlarged span. Then, there exist coefficients $c_0, c_1, (e_w)_w, (b_w)_w, (d_w)_w \in \mathbb{F}_p$ such that

$$z\beta_0 = c_0 + c_1 z(\beta_{-1} + \beta_0) \\ + \sum_{w \in \Omega_{\mathcal{C}}} (e_w Z(w) + b_w Z(w)\beta_{-1} + d_w Z(w)\beta_0).$$

Comparing the coefficients of β_{-1} and β_0 gives

$$0 = c_1 z + \sum_{w \in \Omega_{\mathcal{C}}} b_w Z(w), \quad z = c_1 z + \sum_{w \in \Omega_{\mathcal{C}}} d_w Z(w). \quad (19)$$

If $c_1 \neq 0$, the first equality expresses z as a linear combination of the evaluations $(Z(w))_{w \in \Omega_{\mathcal{C}}}$. If $c_1 = 0$, the second equality does so. Now, suppose that

$$z = \sum_{w \in \Omega_{\mathcal{C}}} \mu_w Z(w)$$

for coefficients $\mu_w \in \mathbb{F}_p$. Since

$$Z(w) = z + \sum_{s=1}^t a_s w^s,$$

comparison of the coefficients of a_s , for $s \in [t]$, gives

$$\sum_{w \in \Omega_{\mathcal{C}}} \mu_w w^s = 0.$$

The points in $\Omega_{\mathcal{C}}$ are distinct and nonzero, and $|\Omega_{\mathcal{C}}| = \text{wt}(\mathcal{C}) \leq t$. Therefore, the corresponding Vandermonde matrix has full column rank, and $\mu_w = 0$ for every $w \in \Omega_{\mathcal{C}}$. Comparing the coefficient of z then gives the desired contradiction. Hence, it follows that $z\beta_0$ is outside the formal $\mathbb{G}_2$ span.

Finally, the adversary can distinguish the symbolic and real games only if two distinct formal expressions in the symbolic game receive the same value in the real game. Let

$$D := \prod_{\substack{j \notin \mathcal{C} \\ i \in [\ell]}} (\rho_j + \alpha_i) \quad \text{and} \quad M := (N - |\mathcal{C}|)\ell.$$

The denominators of the exponents represented in $\mathbb{G}_1$ and $\mathbb{G}_2$ divide D and D^2 , respectively, since the public encodings contain both $g_{j,i}$ and $g_{j,i}^2$. A pairing therefore produces a rational function whose denominator divides D^3 and whose numerator has degree at most $3M + 3 \leq 3(N\ell + 1)$. Hence, by the Schwartz–Zippel lemma and a union bound over at most $O(Q^2)$ pairs of expressions, after clearing the denominators of the rational functions, the adversary can distinguish the symbolic and real games with probability at most $O(Q^2(3N\ell + 3)/p)$, which gives the stated bound. $\square$

5.3 Cost analysis for Construction 5

Table 3 gives the storage and communication costs for the partial-fraction-based weighted BTE scheme, and Table 4 gives its online computation costs. The secret key is the single scalar ρ_j . A party can derive $g_{j,i} = (\rho_j + \alpha_i)^{-1}$ once and cache these coefficients. We list the cost of deriving this cache separately from the subsequent pre-decryption MSM. The decryption rows give the opening cost after the set T has been accepted, so the cost of either individual or randomized batch verification must be added as appropriate. The “Randomized batch verification” row gives costs when all shares are valid, and excluding the cost of finding which shares are invalid. For the two rows corresponding to opening costs, we fold the outer α_i factor into $H_i := \alpha_i(\alpha_i + 1)D_i - \alpha_i \mathbf{dk}[0] \in \mathbb{G}_2$ before pairing, where $D_i := \sum_{w \in \Omega_T} \lambda_{T,w} \mathbf{dk}_{w,i}[2]$. The remaining target group sum then requires one scalar multiplication per ciphertext.

Table 3: Storage and communication costs for Construction 5. Here N is the number of real parties, $W = \sum_{j=1}^N w_j$ is the total weight, $\tau = |T|$ is the number of parties in the accepted set, and ℓ is the padded batch size. We exclude the NIZK, the bilinear group description and fixed generators, and the description of the access structure $(\Omega_j)_{j \in [N]}$; the public indices $(\alpha_i)_{i \in [\ell]}$ are counted explicitly.

Storage costs	
Quantity	Size
Encryption key $\mathbf{ek}$	$ \mathbb{G}_T $
Party secret key $\mathbf{sk}_j$	$ \mathbb{F}_p $
Optional cached coefficients $(g_{j,i})_{i \in [\ell]}$, in addition to $\mathbf{sk}_j$	$\ell \mathbb{F}_p $
Party verification key $\mathbf{vk}_j$	$\ell \mathbb{G}_2 $
All parties' verification keys	$N\ell \mathbb{G}_2 $
Public decryption key $\mathbf{dk} = (\mathbf{dk}[0], (\mathbf{dk}_{w,i}[1], \mathbf{dk}_{w,i}[2])_{w,i})$	$(2W\ell + 1) \mathbb{G}_2 $
Public decryption key including all verification keys	$((2W + N)\ell + 1) \mathbb{G}_2 $
Public indices $(\alpha_i)_{i \in [\ell]}$	$\ell \mathbb{F}_p $
Counted non-NIZK public material	$\ell \mathbb{F}_p + ((2W + N)\ell + 1) \mathbb{G}_2 + \mathbb{G}_T $
Ciphertext, excluding π_{DL}	$ \mathbb{G}_1 + \mathbb{G}_T $
Padded batch of ciphertexts, excluding proofs	$\ell \mathbb{G}_1 + \ell \mathbb{G}_T $
One party's pre-decryption key $\mathbf{sbk}_j$	$ \mathbb{G}_1 $
Pre-decryptions from an accepted set T	$\tau \mathbb{G}_1 $
Decrypted message batch	$\ell \mathbb{G}_T $

5.4 Distributed Key Generation for Construction 5

The setup of Construction 5 has the same shape as that of Construction 1: a secret shared master polynomial Z , and a single secret key ρ_j per party from which all of the public decryption keys owned by the party are derived. These elements of the public decryption key are again shares $Z(w)$ blinded with powers of the party's secret keys. This is what makes the scheme amenable to a DKG protocol analogous to the one in Section 4.3. Relative to the original partial fraction scheme of [11], the only structural change is that the global values $p_{-1}(x)$ and $p_0(x)$ are replaced by independently sampled scalars β_{-1} and β_0 , which can be generated additively by the parties. The public indices $\alpha_1, \dots, \alpha_\ell$ are fixed in advance and are part of the public DKG context. We sketch the protocol below in the same synchronous model as Section 4.3, over the Byzantine-broadcast primitive of Definition 10, with every statement bound to the session identifier and the sender's identity, and with straightline-extractable proofs of knowledge.

The protocol has two steps. In the first step, every party P_j samples its secret key $\rho_j \leftarrow \$ \mathbb{F}_p$ together with additive contributions $\beta_{-1}^{(j)}, \beta_0^{(j)} \leftarrow \$ \mathbb{F}_p$ to the global scalars. It derives $g_{j,i} := (\rho_j + \alpha_i)^{-1}$ for every $i \in [\ell]$ and broadcasts

$$[\rho_j]_1, \quad [\beta_{-1}^{(j)}]_2, [\beta_0^{(j)}]_2, \quad ([g_{j,i}]_1, [g_{j,i}]_2, [g_{j,i}^2]_2)_{i \in [\ell]},$$

together with proofs of knowledge of ρ_j , $\beta_{-1}^{(j)}$, and $\beta_0^{(j)}$. The $\mathbb{G}_2$ encodings of the $g_{j,i}$ form the party's verification key, while the remaining elements are needed to construct the public decryption key. As in Section 4.3, the proof of knowledge binds $[\rho_j]_1$ to a known scalar, and a short chain of pairing equations forces the other encodings to be the correct functions of that scalar: a recipient checks $e([\rho_j]_1 + \alpha_i[1]_1, [g_{j,i}]_2) = [1]_T$ (so that $g_{j,i}$ is indeed the inverse of $\rho_j + \alpha_i$), that the $\mathbb{G}_1$ and $\mathbb{G}_2$ encodings of $g_{j,i}$ agree, and that $[g_{j,i}^2]_2$ is the square of $g_{j,i}$. It also rejects transcripts whose values $\rho_j + \alpha_i$ collide, which enforces the distinctness condition of Setup. Once all Step

Table 4: Online computation costs for Construction 5. For $a \in \{1, 2, T\}$, SM_a is a scalar multiplication in $\mathbb{G}_a$ and $\text{MSM}_a(q)$ is a size- q multi-scalar multiplication. P denotes a pairing, $\text{MP}(q)$ a size- q multi-pairing, and $\text{CauchyFFT}_a(q)$ the collective FFT evaluation of all q Cauchy sums in $\mathbb{G}_a$. Finally, $\text{BatchInv}_{\mathbb{F}_p}(q)$ denotes a batch inversion of q field elements, using one field inversion and $O(q)$ field operations. We omit group additions, other field arithmetic, Lagrange coefficient generation, setup, and NIZK costs. Here $\tau = |T|$ and $W_T = \sum_{j \in T} w_j$.

Computational costs	
Operation	Cost
Enc, one ciphertext	$1 \text{SM}_1 + 1 \text{SM}_T$
PreDec _{j} coefficient derivation from $\text{sk}_j = \rho_j$	$1 \text{BatchInv}_{\mathbb{F}_p}(\ell)$
PreDec _{j} , one padded batch, with cached coefficients	$1 \text{MSM}_1(\ell)$
Verify one submitted pre-decryption key sbk_j	$1 \text{MP}(\ell + 1)$
Verify τ pre-decryption keys individually	$\tau \text{MP}(\ell + 1)$
Randomized batch verification of τ pre-decryption keys	$1 \text{MSM}_1(\tau) + \ell \text{MSM}_2(\tau) + 1 \text{MP}(\ell + 1)$
Dec opening, direct Cauchy-sum evaluation	$2\ell \text{P} + \ell \text{MP}(\tau) + \ell \text{MSM}_1(\ell - 1) + \sum_{j \in T} \ell \text{MSM}_2(w_j) + \ell \text{MSM}_2(W_T) + \ell \text{MSM}_2(2) + \ell \text{MSM}_T(\ell - 1) + \ell \text{SM}_T$
Dec opening, FFT-accelerated Cauchy sums	$2\ell \text{P} + \ell \text{MP}(\tau) + \text{CauchyFFT}_1(\ell) + \sum_{j \in T} \ell \text{MSM}_2(w_j) + \ell \text{MSM}_2(W_T) + \ell \text{MSM}_2(2) + \text{CauchyFFT}_T(\ell) + \ell \text{SM}_T$

1 broadcasts have terminated, the parties fix the set J of parties with valid transcripts and the domain Ω_J , aggregate $[\beta_a]_2 := \sum_{j \in J} [\beta_a^{(j)}]_2$ for $a \in \{-1, 0\}$, and derive the public encodings $[u_{j,i}]_2 = [g_{j,i}^2]_2 + [\beta_{-1}]_2 / (\alpha_i + 1) + [\beta_0]_2 / \alpha_i$ by linear operations. The conditions $\beta_0 \neq 0$ and $\beta_{-1} + \beta_0 \neq 0$ required by Setup are checked on the aggregated encodings, and the step is restarted in the unlikely event that either fails.

In the second step, each dealer P_h samples a uniformly random polynomial $Z^{(h)}$ of degree at most t and broadcasts $\mathbb{G}_1$ commitments to its coefficients, a proof of knowledge of the coefficients, and, for every $w \in \Omega_J$ and $i \in [\ell]$, the elements

$$Z^{(h)}(w) [g_{\text{own}(w),i}]_2, \quad Z^{(h)}(w) [u_{\text{own}(w),i}]_2, \quad Z^{(h)}(0) [(\beta_{-1} + \beta_0)]_2.$$

A recipient derives the evaluation commitments $[Z^{(h)}(w)]_1$ from the coefficient commitments by linear operations and verifies each broadcast element by pairing $[Z^{(h)}(w)]_1$ with the Step 1 encoding of $g_{\text{own}(w),i}$, $u_{\text{own}(w),i}$, or $\beta_{-1} + \beta_0$. Because the dealer commits to coefficients rather than to evaluations, the degree bound holds by construction and no low-degree test is needed, in contrast to Section 4.3; the tradeoff is that each verifier performs $O(Wt)$ group operations per dealer to derive the evaluation commitments. Finally, each party fixes the set I of dealers with valid transcripts and adds up the corresponding components of all valid transcripts, exactly as in Section 4.3. With $Z := \sum_{h \in I} Z^{(h)}$, the aggregated elements are precisely $\text{dk}_{w,i}[1]$, $\text{dk}_{w,i}[2]$, and $\text{dk}[0]$ of Setup, the

encryption key is obtained as $\text{ek} = e([Z(0)]_1, [\beta_0]_2)$, the verification keys are the Step 1 encodings $[g_{j,i}]_2$, and party P_j retains $\text{sk}_j = \rho_j$ as its secret key. Both steps require the accepted set to have weight greater than t , which guarantees that it contains an honest party. The first step communicates $O(N\ell)$ group elements and the second $O(N(W\ell + t))$ group elements when all parties act as dealers.

We defer the proof of correctness and security of this DKG protocol to future work. We expect a correctness proof to follow the same steps as the proof of Theorem 4 using only the pairing checks. As in Section 4.3, a rushing adversary may bias the distributions of β_{-1} , β_0 , and Z , so a security proof for the DKG-generated keys would need a transcript-augmented variant of Assumption 8, analogous to Assumption 5.

6 Experiments

We next report on our Rust implementation of the schemes from Section 4 and Section 5, using the `blst` library’s [31] implementation of fields, curves and pairings. Our experiments were run on an Apple MacBook Pro with an M4 processor and 24 GB of RAM. We also re-implemented BEAT⁺⁺ [2], BTX [28,3], and the partial-fraction-based BTE scheme of [11] for comparison using the same implementations of fields, curves, and pairings and a similar level of optimization. Code for our experiments is available in [5] text.

We benchmark the schemes with an authorization threshold of $2/3$. We chose the distribution of weights to approximate Solana’s share distribution [6]. We used the algorithm used by Aptos [8] to assign weights to parties given a stake distribution. For our experiments, we chose total weights of 764, 1,580, 3,060, and 6,486 corresponding to different qualities of approximation. These are the lowest total weight counts for an approximation of Solana’s stake distribution around the time of writing such that no decryption committee with total stake fraction less than $2/3 - \epsilon$ is authorized, and every committee with total stake fraction more than $2/3 + \epsilon$ is authorized, for $\epsilon = 1/8, 1/16, 1/32$, and $1/64$ respectively. We describe the parameter choices and the algorithm from [8] in greater depth in Appendix A.

We target Solana’s block times of 400 ms [7] as an upper bound for decryption time (threshold decryption is on the “hot path” of consensus). Based on personal communications with Solana developers, 12 cores of a powerful server is a reasonable allocation of resources to this task, and a batch size of 16 is a reasonable choice for the number of concurrent proposals [29,1].

6.1 Performance results for Constructions 1 and 5

In Table 5, we provide performance numbers for our two weighted BTE schemes. We label Construction 1 as WBTX and Construction 5 (Section 5) as WPFE. In Table 6, we show how the performance of the two schemes varies across different batch sizes. As can be seen, both schemes achieve comparable performance, with a total decryption time of 147.5ms and 139.8ms for $W = 1580, \ell = 16$. Furthermore, the bulk of decryption work in both schemes is related to computing terms which depend on a specific decryption committee, (e.g., the sums $\sum_{\omega \in \Omega_j} \lambda_{T',\omega} \text{dk}_{\omega,-i}$ and $\sum_{\omega \in \Omega_{T'}} \lambda_{T',\omega} \text{dk}_{\omega,k-i}$ in Construction 1), but not the values of ciphertexts in a batch. While knowing a decryption committee in advance is a very strong assumption in a blockchain context, if these sums can be precomputed,⁷ the time to decrypt a batch drops to 21.0ms and 18.2ms with precomputed per-committee values.

⁷ We envision that validators could cache these sums for a few commonly used committees (likely validators who are geographically close to them) and use this as an additional optimization to decryption.

W	PreDec (ms/item)		Validate sbk's (ms/item)		Dec Precomp (ms)		Dec Open (ms)		Dec Total (ms)	
	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE
764	0.068	0.063	0.189	0.513	75.9	72.3	18.7	24.7	96.6	98.8
1,580	0.055	0.058	0.168	0.166	124.3	119.5	21.0	18.2	147.5	139.8
3,060	0.055	0.052	0.164	0.160	205.8	205.4	22.2	18.6	230.2	226.0
6,486	0.056	0.057	0.161	0.163	381.8	364.8	21.5	18.4	405.6	385.5

Table 5: Mean running times of decryption phases of WBTX and WPFE while varying total weight. Batch size ℓ is fixed at 16 for these experiments. Dec Precomp and Dec Open correspond to the ciphertext-independent and ciphertext-dependent parts of Dec respectively.

ℓ	PreDec (ms/item)		Validate sbk's (ms/item)		Dec Precomp (ms)		Dec Open (ms)		Dec Total (ms)	
	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE
16	0.055	0.058	0.168	0.166	124.3	119.5	21.0	18.2	147.5	139.8
32	0.081	0.083	0.302	0.305	234.9	226.2	45.1	61.2	283.1	290.5
64	0.129	0.126	0.563	0.551	482.1	436.6	95.4	72.7	582.4	514.1
128	0.210	0.211	1.049	1.055	876.5	861.3	114.1	153.1	998.6	1022.4
256	0.346	0.352	2.032	2.129	1774.3	1781.6	231.1	231.8	2018.5	2026.7

Table 6: Mean running time of decryption phases of WBTX and WPFE while varying batch size. The total weight is fixed at 1,580 for these experiments. Dec Precomp and Dec Open correspond to the ciphertext-independent and ciphertext-dependent parts of Dec respectively.

We give the other costs of the schemes in Table 7, showing the encryption time and the sizes of the public key material.

6.2 Comparison to prior work

We report the performance of our re-implementations of three prior schemes from the literature in Table 8. We re-implement the index-dependent weighted batch threshold encryption scheme of [2], labeled BEAT++. We label the scheme of Agarwal et al. and Policharla [3,28] as BTX and the scheme of [11] as PFE.⁸

Implementations for BEAT++ and BTX were not available publicly at the time of writing. While our reimplementation of BTX is slower than the quoted results of [3], their scheme uses AVX-512 optimized implementations of primitives. We re-implemented both schemes using the same collection of primitives and level of optimization as our schemes. While there were published implementations of [28] and of PFE [11], our re-implementations are faster, which makes the comparisons more fair. An independent contribution of our work is the fastest open-source implementation of BTX [3,28].

For BEAT++, we ran our implementation with the same per-party weights and batch sizes as our scheme. As can be seen, the total decryption running time of BEAT++ was about 1.3–1.6 $\times$ faster than our scheme at these batch sizes due to its simpler (and index-dependent) algebraic structure. For BTX and PFE, we ran the schemes with parameters that one would use with “naive

⁸ Meaning Partial Fraction Encryption.

W	ℓ	Enc (ms/item)		Public Decryption Key Size (kB)		Verification Key Size (kB)		Total Key Size (kB)	
		WBTX	WPFE	WBTX	WPFE	WBTX	WPFE	WBTX	WPFE
764	16	0.080	0.081	2220	2292	351	351	2571	2643
1,580	16	0.061	0.066	4592	4740	629	629	5220	5369
3,060	16	0.067	0.102	8893	9180	897	897	9790	10077
6,486	16	0.064	0.066	18850	19458	1032	1032	19882	20490
1,580	32	0.062	0.060	9332	9480	1257	1257	10589	10737
1,580	64	0.053	0.051	18812	18960	2514	2514	21326	21474
1,580	128	0.051	0.049	37772	37920	5028	5028	42800	42948
1,580	256	0.049	0.053	75692	75840	10056	10056	85748	85896

Table 7: Encryption cost and public key sizes of WBTX and WPFE.

W	Enc (ms/item)			PreDec (ms/item)			Dec (ms)		
	PFE	BTX	BEAT++	PFE	BTX	BEAT++	PFE	BTX	BEAT++
764	0.079	0.427	0.102	0.011	0.011	0.021	14.8	15.0	62.3
1,580	0.077	0.431	0.080	0.011	0.011	0.018	27.2	26.8	102.7
3,060	0.072	0.450	0.085	0.011	0.011	0.018	70.4	70.2	162.1
6,486	0.081	0.429	0.083	0.011	0.011	0.017	274.4	270.6	268.9

Table 8: Mean running times of encryption and decryption phases of the reimplemented prior schemes at batch size $\ell = 16$.

virtualization”, meaning that for a party assigned weight w_j , it would simply be given w_j independent keys in a non-weighted batch threshold encryption scheme. These schemes’ total decryption running time was 1.4–6.7 $\times$ faster than our scheme. This is explained by their simpler algebraic structure. However, as mentioned previously, these schemes only support DKGs via generic MPCs and do not have pre-decryption keys which are succinct in the amount of weight assigned to an individual party. For instance, the parties with the most stake at the largest setting of $W = 6,486$, with individual weights derived from Solana’s stake distribution, would have individual weights in the range of 100–200. This would impact the “hot-path” communication for threshold decryption. To conclude, the prior schemes have faster threshold decryption, though this is to be expected because they have fewer features and simpler structures that require less algebraic work.

7 Additional related work

We discuss the main lines of prior research relevant to this work.

Encrypted mempools/MEV. Public blockchain mempools reveal pending transactions before their ordering is finalized, enabling front-running and other forms of maximal extractable value (MEV) [19]. Encrypted mempools seek to mitigate these attacks by hiding transaction contents until their order has been fixed. This setting motivates threshold encryption schemes that remain efficient for large batches of transactions and whose access structures reflect each validator’s stake [21].

Batch- and epoch-labelled BTE. Choudhuri et al. [17] introduced the idea of batch threshold encryption (BTE), which lets each party in a threshold encryption scheme publish a short pre-decryption key to decrypt an entire selected batch of ciphertexts while ciphertexts outside that

batch remain private. Subsequent works improved on this approach by supporting reusable setups or building related batched identity- and tag-based schemes [18,4,30]. These constructions still associate every ciphertext with some sort of batch-specific information, so a ciphertext omitted from its intended batch may become “stranded”.

Epochless, index-dependent BTE. BEAT-MEV [14] removes batch labels and allows ciphertexts to be reused across batches. However, its encryption requires an index from a fixed public domain, and a decryptable batch must contain at most one ciphertext for each index. Dealing with index collisions requires additional communication or leads to issues with censorship resistance. Subsequent works improve on this approach [13,2]. Boneh, Laufer, and Tas [10] construct epochless batch IBE and threshold batch IBE schemes from lattices, though with much larger parameters than the pairing-based schemes.

Weighted threshold cryptography. Proof-of-stake systems motivate cryptographic schemes with weighted access structures, since the influence of a validator is determined by its stake. Ferveo [9] supports (non-batched) weighted threshold decryption while compressing the virtual decryption shares for one ciphertext into a single group element per party. Agarwal et al. [2] (BEAT⁺⁺) give the first weighted BTE scheme where a party’s pre-decryption communication is independent of its assigned weight, and reduce decryption from quadratic to quasilinear complexity, though its encryption is still index-dependent.

Epochless, index-free BTE. Recent works construct pairing-based batch threshold encryption schemes without indices or epoch-specific labels. Boneh et al. [11] adapt partial-fraction techniques [26] to construct an epochless and index-free BTE scheme with similar properties. Similarly, Agarwal et al. [3] (BTX) and, concurrently, Policharla [28] give the same BTE construction with short ciphertexts and quasilinear decryption. We use these schemes as the starting points for Construction 5 and Construction 1 (Section 5) respectively. We extend them to support weighted access structures and to admit simple DKG protocols.

Distributed key generation. Das et al. [20] construct DKG protocols for weighted threshold cryptosystems in the context of weighted verifiable unpredictable functions. We adapt their correctness framework (properties C1–C3) to the key structure of our constructions. Our protocols most directly build on Ferveo’s DKG protocol [9].

Acknowledgements Aditi Partap and Alexander Frolov were interns at a16z Crypto Research at the time of writing. We thank Guru Vamsi Policharla for his helpful feedback on a draft of this work.

References

1. 0xIchigo: Constellation: A proposal for mcp on solana. Helius Blog (Apr 2026), <https://www.helius.dev/blog/constellation>, published April 7, 2026
2. Agarwal, A., Babel, K., Das, S., Gilkalaye, B.P., Mondal, A., Pinkas, B., Rindal, P., Yadav, A.: Weighted batched threshold encryption with applications to mempool privacy. In: SP. pp. 2462–2481. IEEE (2026)
3. Agarwal, A., Das, S., Gilkalaye, B.P., Rindal, P., Shoup, V.: BTX: Simple and efficient batch threshold encryption. Cryptology ePrint Archive, Paper 2026/754 (2026), <https://eprint.iacr.org/2026/754>
4. Agarwal, A., Fernando, R., Pinkas, B.: Efficiently-thresholdizable batched identity based encryption, with applications. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part III. LNCS, vol. 16002, pp. 69–100. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01881-6_3
5. Alexander Frolov: Weighted BTE. https://github.com/sashafrlov/weighted_bte (2026)

6. Anza: Scheduler War. <https://schedulerwar.vercel.app/> (2026), accessed: 2026-08-06
7. Anza Technologies: solana-sdk: clock/src/lib.rs. Software repository (2026), <https://github.com/anza-xyz/solana-sdk/blob/d045b94c18f8cae8bc30eec310984030ead8d4f4/clock/src/lib.rs#L79>, commit d045b94, accessed 2026-08-02
8. Aptos Foundation: real_dkg/rounding/mod.rs. https://github.com/aptos-labs/aptos-core/blob/main/types/src/dkg/real_dkg/rounding/mod.rs
9. Bebel, J., Ojha, D.: Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Report 2022/898 (2022), <https://eprint.iacr.org/2022/898>
10. Boneh, D., Laufer, E., Tas, E.N.: Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Report 2025/1254 (2025), <https://eprint.iacr.org/2025/1254>
11. Boneh, D., Nema, R., Roy, A., Tas, E.N.: Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674 (2026), <https://eprint.iacr.org/2026/674>
12. Boneh, D., Shoup, V.: A graduate course in applied cryptography. <https://toc.cryptobook.us/book.pdf> (2023)
13. Bormet, J., Choudhuri, A.R., Faust, S., Garg, S., Othman, H., Policharla, G.V., Qu, Z., Wang, M.: BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Report 2025/1419 (2025), <https://eprint.iacr.org/2025/1419>
14. Bormet, J., Faust, S., Othman, H., Qu, Z.: BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. In: Bauer, L., Pellegrino, G. (eds.) USENIX Security 2025. pp. 3457–3476. USENIX Association (Aug 2025), <https://www.usenix.org/conference/usenixsecurity25/presentation/bormet>
15. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: Definitions and practical constructions. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 9453, pp. 262–288. Springer (2015)
16. Cascudo, I., David, B.: Scrape: Scalable randomness attested by public entities (06 2017). https://doi.org/10.1007/978-3-319-61204-1_27
17. Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.V.: Mempool privacy via batched threshold encryption: Attacks and defenses. In: Balzarotti, D., Xu, W. (eds.) USENIX Security 2024. USENIX Association (Aug 2024), <https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri>
18. Choudhuri, A.R., Garg, S., Policharla, G.V., Wang, M.: Practical mempool privacy via one-time setup batched threshold encryption. In: Bauer, L., Pellegrino, G. (eds.) USENIX Security 2025. pp. 3477–3495. USENIX Association (Aug 2025), <https://www.usenix.org/conference/usenixsecurity25/presentation/choudhuri>
19. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE Symposium on Security and Privacy. pp. 910–927. IEEE Computer Society Press (May 2020). <https://doi.org/10.1109/SP40000.2020.00040>
20. Das, S., Pinkas, B., Tomescu, A., Xiang, Z.: Distributed randomness using weighted VUFs. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VII. LNCS, vol. 15607, pp. 314–344. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91098-2_12
21. Dziembowski, S., Faust, S., Luhn, J.: Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Report 2024/1981 (2024), <https://eprint.iacr.org/2024/1981>
22. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS. pp. 308–317. IEEE Computer Society Press (Oct 1990). <https://doi.org/10.1109/FSCS.1990.89549>
23. Garimidi, P., Neu, J., Resnick, M.: Multiple concurrent proposers: Why and how (2025), <https://arxiv.org/abs/2509.23984>
24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT’99. LNCS, vol. 1592, pp. 295–310. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_21
25. Gurkan, K., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Aggregatable distributed key generation. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 147–176. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_6
26. Jutla, C.S., Nema, R., Roy, A.: Partial fraction techniques for cryptography. In: Daemen, J., Thomé, E. (eds.) EUROCRYPT 2026, Part V. LNCS, vol. 16545, pp. 423–453. Springer, Cham (May 2026). https://doi.org/10.1007/978-3-032-25330-9_15
27. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO’91. LNCS, vol. 576, pp. 129–140. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9

28. Policharla, G.V.: A simple batched threshold encryption scheme. Cryptology ePrint Archive, Paper 2026/760 (2026), <https://eprint.iacr.org/2026/760>
29. Solana core developers: Personal communication. Private communication with the authors (2026)
30. Suegami, S., Ashizawa, S., Shibano, K.: Constant-cost batched partial decryption in threshold encryption. Cryptology ePrint Archive, Paper 2024/762 (2024), <https://eprint.iacr.org/2024/762>
31. Supranational: blst: Multilingual BLS12-381 signature library. Software repository (2020), <https://github.com/supranational/blst>

A Choosing total weights for the Solana chain

For our experiments, we computed total weight and per-party weights based on a recent snapshot of the stake distribution among Solana’s validators obtained from [6]. At the time of writing, there were 862 validator accounts with active stake, totaling about 478.4 million SOL.

Aptos implements a weighted threshold cryptographic scheme based on [20]. They implement an algorithm for computing a total weight and assigning integer weights to parties based on stake percentages in [8]. We use Aptos’ algorithm to compute the per-party weights approximating Solana’s share distribution, choosing it because this is an algorithm with real-world usage. However, there are multiple, equally valid approaches.

Assigning integer weights that approximate each party’s stake necessarily leads to some approximation error from using integer share counts. Slightly reinterpreted, Aptos’ algorithm takes a target threshold T and an acceptable deviation ε . The algorithm computes a minimum share count such that no decryption committee with true stake fraction less than $T - \varepsilon$ is authorized, and such that any committee with true stake fraction more than $T + \varepsilon$ is authorized. The algorithm then rounds each party’s fraction of the stake to determine the number of shares to assign to each party. The Aptos code interprets the algorithm differently, instead treating $T - \epsilon$ and $T + \epsilon$ as direct parameters of the protocol (setting them to $1/2$ and $2/3$).

In Table 9, for $T = 2/3$ and varying ε , we show W , the minimal total weight required to achieve error ε , as well as some statistics for each resulting assignment of weights. Specifically, we report the actual threshold q , the minimum weight required to decrypt ciphertexts. We also specify the number of validators assigned nonzero weights for each value of ε and the percentage of stake assigned to validators who receive zero weight. We also measure the quality of approximations. Letting $\hat{p}_j = w_j/W$ be validator j ’s approximate stake fraction and p_j be their true stake fraction, we report statistics on each party’s absolute approximation error $|\hat{p}_j - p_j|$, as well as their relative error $|\hat{p}_j - p_j|/p_j$. Validators receiving zero weight are included in these statistics.

Table 9: Aptos-style share approximations to the Solana active-stake distribution for target threshold $T = 2/3$ and varying allowed deviations ε .

Metric	Allowed deviation ε				
	1/8	1/16	1/32	1/64	1/128
Computed total weight W	764	1,580	3,060	6,486	12,460
Minimum reconstruction weight q	512	1,060	2,044	4,327	8,308
$(T + \varepsilon)$ reconstruction boundary (%)	75.658188	72.884133	69.791029	68.227607	67.447787
Validators with nonzero weights	234	419	598	688	729
Stake (%) held by validators with zero weight	13.3965	5.5137	1.4216	0.3656	0.1176
Mean absolute error (pp)	0.032983	0.015630	0.007260	0.003622	0.001814
Median absolute error (pp)	0.023706	0.016225	0.007337	0.003650	0.001768
Maximum absolute error (pp)	0.452110	0.078842	0.019015	0.007689	0.004087
Median relative error (%)	100.00	100.00	25.98	13.14	5.90

We choose the weight distributions with $\varepsilon = 1/8, 1/16, 1/32, 1/64$ for our experiments. The resulting total share counts are comparable to those used by [20]’s experiments. Furthermore, at $\varepsilon = 1/32, 1/64$, the high-end reconstruction boundary comes within a few percentage points of the true threshold, and the various error statistics drop to a few basis points. Our reimplementations of Aptos’ share assignment rule and generated allocations are available in the `scripts` directory at [5] text.