# Adaptively Secure (Threshold) Batched IBE from Static Assumptions

Sourav Das[1], Pratish Datta[2], Aditi Partap[3], and Swagata Sasmal[4]

[1]Category Labs

souravdas1547@gmail.com

[2]NTT Research

pratish.datta@ntt-research.com

[3]Stanford University

aditi712@cs.stanford.edu

[4]Indian Statistical Institute

swagata.sasmal@gmail.com

**Abstract**

Batched identity-based encryption (batched IBE) lets a key authority release one short decryption key for a set of identities and a batch label/epoch. It has been used to mitigate maximal extractable value (MEV) attacks on public blockchains. Existing constructions either use the generic-group and/or random-oracle models, or rely on $q$-type assumptions whose size grows with the batch-size bound.

In this paper, we present the first adaptively secure batched IBE scheme from static assumptions. Our scheme supports an exponential identity space and an exponential epoch space, and its ciphertexts, digests, and batch keys consist of a constant number of group elements independent of the batch size. Our scheme relies on composite-order bilinear groups, and we prove its adaptive security assuming hardness of static subgroup decision assumptions and a static pairing assumption, with a security loss polynomial in the security parameter $\lambda$ and the number of key queries. Our proof does not rely on $q$-type assumptions, random oracles, generic-group arguments, or complexity leveraging. We also present a threshold version with adaptive challenge security under static corruption, and translate our scheme to prime-order bilinear groups under the static MDDH assumption.

# Contents


**1 Introduction** **3**

**2 Technical Overview** **5**
- 2.1 Batched IBE and Adaptive Security . . . 6
- 2.2 Our Approach and the Epochless Construction . . . 6
- 2.3 Adding Epochs via the Lewko–Waters IBE Layer . . . 13
- 2.4 Extending to Subset Policies . . . 18
- 2.5 Supporting an Exponential Identity Space . . . 20
- 2.6 Extending to Threshold Batched IBE . . . 22

**3 Preliminaries** **24**
- 3.1 Asymmetric Bilinear Groups . . . 24
- 3.2 Composite-order Static Assumptions . . . 25
- 3.3 Prime-order Assumptions . . . 26
- 3.4 Subset-policy Threshold Batched IBE . . . 26

**4 A Generic Cover-Free Compiler** **30**
- 4.1 Cover-Free Families and the Compiler . . . 30
- 4.2 Correctness and Security . . . 32
- 4.3 A Cover-free Family for an Exponential Identity Space . . . 34

**5 Threshold Batched IBE in Composite-Order Groups** **35**
- 5.1 Construction . . . 36
- 5.2 Security Analysis . . . 39

**6 Batched IBE from MDDH in Prime-Order Groups** **48**
- 6.1 Construction . . . 48
- 6.2 Security Analysis . . . 50

**References** **56**

# 1 Introduction

*Batched identity-based encryption* (batched IBE) allows a key authority to release a short decryption key for any chosen set of identities in a batch. This primitive has been studied extensively in the past few years due to its application in mitigating maximal extractable value (MEV) related attacks on public blockchains. In particular, to reduce MEV, users submit encrypted transactions; the protocol orders ciphertexts and decrypts them only after it fixes the order [8, 16, 7]. After ordering, the protocol should release short decryption material that opens exactly the transactions included in the block and leaves the remaining transactions in the mempool encrypted. Standard IBE gives one short key for a batch label, but that key decrypts every ciphertext under the label, including ciphertexts not included in the block. Alternatively, issuing separate decryption material for every selected ciphertext gives selectivity, but the total key size grows with the number of selected ciphertexts.

Batched IBE captures both requirements. An encryptor labels each ciphertext with an identity $i$ and a batch label, or epoch, $\tau$. The key generator issues a secret key for a set of identities $\Gamma$ and a batch label $\tau$. Decryption succeeds if and only if the labels match and $i \in \Gamma$. We want the key for $\Gamma$ to be short, ideally independent of $|\Gamma|$. We also want adaptive security, where the adversary may choose the challenge identity and batch label after seeing polynomially many admissible secret keys.

**Existing Approaches and Their Limitations.** Agarwal, Fernando, and Pinkas [1] introduced the notion of batched IBE and presented the first pairing-based instantiation. Their construction supports non-interactive threshold key issuance and is proved secure in the generic group and random oracle models, with one batch key per epoch. TrX [12] gives epochless batched threshold encryption in the generic group and random oracle models. Policharla [20] gives a construction with constant-size ciphertexts and multiple batch keys per epoch through multi-message signatures. The number of keys per epoch is fixed at setup, and security is proved in the generic bilinear group model without random oracles. The paper also sketches non-interactive threshold key issuance. Gong et al. [14] is the first to present a pairing-based batched IBE construction in the plain model. Their construction achieves selective security and obtains constant-size ciphertexts and batch keys under a $q$-type assumption. Their construction also admits a threshold extension with silent setup. They additionally give an adaptively secure variant in the generic group and random oracle models, with a threshold extension secure against static corruptions. Stracovsky, Waters, and Wu [22] obtain the first epochless pairing-based batched IBE scheme, which achieves adaptive security in the generic group model by lifting bounded-collusion security to full collusion resistance. The work of [21] gives batched ABE and a threshold extension under the decisional bilinear Diffie–Hellman assumption in the random oracle model. Another work [17] gives verifiable outsourced batched threshold decryption with silent setup under a $q$-type assumption and extractable proofs of knowledge. Its security reduction avoids the generic group model, while its proposed proof instantiation uses random oracles.

Boneh et al. [4] give the first lattice-based instantiation of batched IBE. They present an epochless batched IBE scheme from LWE and a threshold extension in the random oracle model. Xiong et al. [25] construct epoch-based batched IBE in the plain model under the

$\ell$-succinct LWE assumption [24]. This removes the random oracle, but at the cost of a parameterized assumption. This state of affairs leaves open the following intriguing question:

*Can we build (threshold) batched IBE from static computational assumptions?*

**Our results.** In this paper, we answer the above open problem in the affirmative. We present the first adaptively secure (epoch-based) batched IBE scheme from static assumptions. We build the scheme in asymmetric composite-order bilinear groups and prove its security under two static subgroup decision assumptions [2] and a static pairing assumption. Our proof does not rely on $q$-type assumptions, random oracles, generic-group arguments, or complexity leveraging. We present our construction in five steps.

- *Epochless batched IBE.* We first present a batched IBE scheme without epochs for a polynomial-size identity universe $[n]$. The encryptor indexes ciphertexts by identities, and the key generator indexes secret keys by sets. This construction supports a polynomial number of set-key queries, provided that every queried set omits the challenge identity.
- *Epoch-based batched IBE.* We then bind keys and ciphertexts to epochs using the IBE layer of Lewko and Waters [19]. Epochs let the authority release keys for overlapping batches over time. Epochs also give the adversary more power as it can obtain keys for sets containing the challenge identity, as long as these keys are for epochs different from the challenge epoch. We prove adaptive security of our epoch-based scheme with a dual-system argument [23, 19].
- *Subset-policy batched IBE.* We next associate a ciphertext with a set of identities $X \subseteq [n]$ instead of a single identity. A key for a batch $S$ decrypts it exactly when $X \subseteq S$. This extension changes one ciphertext component, adds no group elements to ciphertexts or keys, and the security proof carries over with the same security loss.
- *Exponential identity space.* Thereafter, we extend our scheme to support an exponential-sized identity space. In particular, we encode each identity in $\{0,1\}^\lambda$ as a block of a cover-free family [10] over $[n]$, and each batch as the union of its blocks. Cover-freeness guarantees that a block lies inside the union exactly when its identity belongs to the batch. The resulting scheme inherits the security of the subset-policy scheme with no additional loss. It also preserves the ciphertext and key sizes.
- *Threshold batched IBE.* Finally, we extend our scheme to $L$ authorities and any reconstruction threshold $1 \leq \theta \leq L$. Each authority computes a partial key from the public digest and the epoch. Anyone can verify partial keys and aggregate any valid $\theta$ of them into a batch key. We prove adaptive challenge security against the static corruption of up to $\theta - 1$ authorities under the same static assumptions. Here, adaptive challenge security against static authority corruption refers to security against adversaries who fixes its authority corruption set before setup, but chooses the challenge identity, epoch, messages, and key queries adaptively.

We provide a detailed technical overview of our schemes in Section 2.

As an independent contribution, we also give a prime-order version of the epoch-based construction under the static matrix Diffie-Hellman (MDDH). We use the basis-hiding framework of Lewko and Chen–Gong–Kowalczyk–Wee [18, 6] to translate our composite order scheme to the prime order setting: we replace the composite-order subgroups by mutually orthogonal prime-order subspaces. Every computational step of the proof reduces to a one-column $\mathcal{U}_{\ell,k}$-MDDH instance in one of the source groups, and the final target randomization is a statistical change of variables, so no assumption in the target group is needed.

**Concurrent work.** In concurrent and independent work, Döttling et al. [9] construct adaptively secure epoch-based batched IBE from CDH in the standard model. Their construction combines deferred encryption via adaptive garbling with registration-based encryption and weak receiver non-committing IBE. Both their construction and ours avoid random oracles and permit at most one batch-key query at the challenge epoch. Our constructions use bilinear groups and dual-system encryption under static subgroup and pairing assumptions, with a prime-order instantiation under MDDH. Let $k$ denote the bit length of their registration hash, which is independent of the batch size. Their encryption garbles a circuit implementing registration-based encryption and performs $2k$ weak receiver non-committing IBE encryptions, storing both the garbled circuit and these ciphertexts in each batched IBE ciphertext. Their batch key contains $k$ corresponding secret keys, and decryption requires $k$ IBE decryptions, garbled-circuit evaluation, and registration-based hint generation and decryption. Our constructions avoid the computation of garbling and evaluating this circuit and the storage of its garbled representation. In particular, our composite-order scheme has only four group elements per ciphertext and only two per batch key, and decryption uses four pairings plus set-dependent public group multiplications. We additionally use Shamir secret sharing to obtain non-interactive, publicly verifiable threshold key issuance with adaptive challenge security against static authority corruptions, whereas their batched IBE construction is non-threshold. However, their construction relies only on CDH and requires no batch bound fixed at setup, while our construction supporting an exponential identity space uses public parameters that grow with a predetermined batch bound.

# 2 Technical Overview

In this section, we describe the main ideas behind our (epoch-based) batched IBE construction and its threshold extension. We first recall the batched IBE notion and the adaptive security we target (§2.1). We then build our scheme in steps. We start with an epochless construction that is secure against a single batch key (§2.2), add epochs (§2.3), and extend the result to subset policies (§2.4). We then use the subset policy scheme to encode exponentially many identities using cover-free sets and explain how to move public computation offline (§2.5). Finally, we explain how to thresholdize our scheme (§2.6).

## 2.1 Batched IBE and Adaptive Security

In a batched IBE scheme, each ciphertext is associated with an identity $\mathsf{id} \in \mathcal{I}$ and a batch label $\tau \in \mathcal{T}$, which we also call an *epoch*. A decryption key is associated with a set of identities $\Gamma \subseteq \mathcal{I}$, which we call a *batch*, and an epoch $\tau'$. A key for $(\Gamma, \tau')$ decrypts a ciphertext for $(\mathsf{id}, \tau)$ exactly when $\mathsf{id} \in \Gamma$ and $\tau' = \tau$. We next describe the syntax and the adaptive security notion informally and defer the formal definitions to §3.

**Syntax.** Informally, a batched IBE scheme consists of the following algorithms.

- **Setup** outputs public parameters $\mathsf{pp}$ that specify the message space $\mathcal{M}$, the identity space $\mathcal{I}$, and the epoch space $\mathcal{T}$.
- **Key generation** outputs a master public key $\mathsf{pk}$ and a master secret key $\mathsf{msk}$.
- **Encryption** takes $\mathsf{pk}$, a message $M \in \mathcal{M}$, an identity $\mathsf{id}$, and an epoch $\tau$, and outputs a ciphertext $\mathsf{ct}$ for $(\mathsf{id}, \tau)$.
- **Digest computation** takes $\mathsf{pk}$ and a batch $\Gamma$ and outputs a short public digest $D_\Gamma$. It uses no secrets, so anyone can compute it.
- **Batch key computation** takes $\mathsf{msk}$, a digest $D_\Gamma$, and an epoch $\tau$, and outputs a batch key $\mathsf{sk}_{\Gamma,\tau}$.
- **Decryption** takes $\mathsf{pk}$, a batch key $\mathsf{sk}_{\Gamma,\tau'}$, the batch $\Gamma$, and a ciphertext $\mathsf{ct}$ for $(\mathsf{id}, \tau)$. It outputs $M$ when $\mathsf{id} \in \Gamma$ and $\tau' = \tau$, and $\perp$ otherwise.

The efficiency requirement is that the batch decryption key have size sublinear (ideally independent) in the batch size $|\Gamma|$. Our constructions achieve a constant number of group elements in the batch key and also in the ciphertexts.

**Adaptive security.** We consider the standard indistinguishability experiment for batched IBE [1, 14]. The adversary, $\mathcal{A}$, receives the public parameters and $\mathsf{pk}$, and adaptively queries batch keys for pairs $(\Gamma, \tau)$ of its choice. At some point, it outputs two messages and a challenge identity/tag pair $(\mathsf{id}^*, \tau^*)$, receives an encryption of one of the two messages under $(\mathsf{id}^*, \tau^*)$, and continues to query keys. Adaptive security requires that no efficient adversary guesses the encrypted message with probability noticeably better than one half.

Clearly, a query at epoch $\tau^*$ must specify a batch that omits $\mathsf{id}^*$. Moreover, we require $\mathcal{A}$ to make at most one query for the challenge epoch $\tau^*$. We allow $\mathcal{A}$ to make an arbitrary number of queries for non-challenge epochs with arbitrary batches, including batches that contain $\mathsf{id}^*$. The single-query restriction at $\tau^*$ matches how the primitive is used in practice. For example, in the encrypted mempool application, an epoch identifies a block height, and protocols typically decrypt only one block at each height.

## 2.2 Our Approach and the Epochless Construction

Following Gong et al. [14], we first construct an epochless scheme with a short key that decrypts ciphertexts for a set of identities, and then use an IBE layer to bind each key and

ciphertext to an epoch. Our epochless scheme can intuitively be viewed as a dual of broadcast encryption (BE), where a ciphertext is associated with a set of recipients and a key with one recipient. We start from the BE scheme of Gay et al. [13], the best known static-assumption analogue of the BGW-style BE [3]. Their scheme supports a polynomial number $n$ of users, so our epochless scheme also supports a polynomial-size identity universe. We lift this restriction later using cover-free sets (§2.5).

**Composite-order groups.** Our schemes uses asymmetric composite-order bilinear groups of order $N = p_1p_2p_3$. Each source group splits into subgroups of prime order $p_1$, $p_2$, and $p_3$, and elements from different subgroups pair to one. This property is known as orthogonality. The real scheme uses the $p_1$-subgroups, the $p_2$-subgroups appear only in the security proof, and the $p_3$-subgroups randomize the public key and the batch keys, which we need for adaptive security. Our epochless construction does not need the $p_3$-subgroups, so in this subsection we work with $N = p_1p_2$.

**The epochless setting.** A ciphertext is associated with an identity $i \in [n]$ and a key with a batch $\Gamma \subseteq [n]$, where $n = n(\lambda)$ is polynomial. The size of the key should be independent of $|\Gamma|$. For simplicity of exposition, we first consider an adversary $\mathcal{A}$ that receives at most one key, which isolates the problem of obtaining a short key for a set of identities [14].

**The construction.** Let $\mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T$ be bilinear groups of order $N = p_1p_2$ and pairing operation $e : \mathbb{G}_N \times \mathbb{H}_N \to \mathbb{G}_T$. We also use $\mathbb{G}_a, \mathbb{H}_a$ for the subgroups of order $p_a$ for $a \in \{1, 2\}$.

Let $g_1 \in \mathbb{G}_1$ and $h_1 \in \mathbb{H}_1$ be generators. Key generation samples

$$\alpha, u, \{w_i, r_i\}_{i\in[n]} \leftarrow\!\!\$\ \mathbb{Z}_N$$

and publishes the master public key:

$$\begin{aligned} &g_1, \quad g_1^u, \quad \{g_1^{w_i}\}_{i\in[n]}, \quad \Omega = e(g_1, h_1)^\alpha, \\ &\{P_j = h_1^{r_j}\}_{j\in[n]}, \quad \{P_{i,j} = h_1^{w_i r_j}\}_{i,j\in[n],\ i\neq j}. \end{aligned}$$

The last collection is the off-diagonal table with missing terms $h_1^{w_i r_i}$ for every $i$. For a batch $\Gamma$, we define the digest and batch key pair $(D_\Gamma, K_\Gamma) \in \mathbb{H}_1 \times \mathbb{H}_1$ as:

$$D_\Gamma := \prod_{j\in[n]\setminus\Gamma} P_j = h_1^{R_\Gamma} \qquad K_\Gamma := h_1^\alpha D_\Gamma^u = h_1^{\alpha+uR_\Gamma},$$

where $R_\Gamma := \sum_{j\in[n]\setminus\Gamma} r_j$.

To encrypt a message $M \in \mathbb{G}_T$ for identity $i$, sample $s \leftarrow\!\!\$\ \mathbb{Z}_N$ and output

$$\mathsf{ct}_i = (C_0, C_1, C_2) = \left(g_1^s,\ g_1^{s(u+w_i)},\ M \cdot \Omega^s\right).$$

To decrypt with $K_\Gamma$ for $i \in \Gamma$, the decryptor computes $E_{i,\Gamma} := \prod_{j\notin\Gamma} P_{i,j} = h_1^{w_i R_\Gamma}$, which uses only off-diagonal entries, and computes

$$\begin{aligned} e(C_0, K_\Gamma)\, e(C_1, D_\Gamma)^{-1}\, e(C_0, E_{i,\Gamma}) &= e(g_1, h_1)^{s(\alpha+uR_\Gamma)-s(u+w_i)R_\Gamma+sw_iR_\Gamma} \\ &= e(g_1, h_1)^{\alpha s}, \end{aligned}$$

which is sufficient to remove the mask $\Omega^s$ from $C_2$.

In terms of security, intuitively, the missing diagonal prevents the adversary $\mathcal{A}$ from decrypting an identity $i^* \notin \Gamma$. In particular, since $i^* \notin \Gamma$, the exponent $R_\Gamma$ contains $r_{i^*}$, so computing $E_{i^*,\Gamma}$ requires the missing element $h_1^{w_{i^*} r_{i^*}}$.

**Adaptive security analysis.** Our proof uses the dual-system technique of Waters and Lewko–Waters [23, 19]. In particular, in our security proof, we introduce semi-functional keys and ciphertexts that additionally carry components in the $p_2$-subgroups. Then, due to orthogonality of subgroups, a normal key decrypts a semi-functional ciphertext and a semi-functional key decrypts a normal ciphertext, whereas a semi-functional key and a semi-functional ciphertext is left with an extra term in $e(\mathbb{G}_2, \mathbb{H}_2)$ hidden from $\mathcal{A}$ which provides the additional entropy to hide the encrypted message from the view of $\mathcal{A}$. We introduce the above changes through a sequence of hybrids and argue indistinguishability between hybrids using subgroup decision assumptions or statistical arguments.

Let $g_2, h_2$ be generators of $\mathbb{G}_2, \mathbb{H}_2$. In formulas involving $\mathbb{G}_2$ or $\mathbb{H}_2$, exponents are understood modulo $p_2$. Therefore, by the Chinese Remainder Theorem, these residues are independent of the residues modulo $p_1$ that determine the real scheme. We describe the hybrid transitions next.

*Step 1: Guessing the challenge identity.* The reduction guesses $i^* \in [n]$ before interacting with $\mathcal{A}$ and aborts if the queried batch contains $i^*$ or the challenge identity differs from $i^*$. The guess is correct with probability $1/n$, so the loss is polynomial. Hereon, we condition on a correct guess.

*Step 2: Making the challenge ciphertext semi-functional.* We replace the challenge ciphertext by

$$\widetilde{\mathsf{ct}}_{i^*} = \big(g_1^s \boxed{g_2^x},\ g_1^{s(u+w_{i^*})} \boxed{g_2^{x\mu}},\ M\Omega^s\big),$$

where $x, \mu \leftarrow\!\!\$\ \mathbb{Z}_{p_2}$ and the boxes mark the semi-functional components here and subsequently. This transition uses the two-prime analogue of the subgroup decision assumption between $\mathbb{G}_1$ and $\mathbb{G}_{12}$ (Assumption 3.1). Throughout this epochless overview, we use the two-prime analogues of the stated assumptions. In particular, given a challenge $T$ from either $\mathbb{G}_1$ or $\mathbb{G}_{12}$, the reduction sets $C_0 := T$, $C_1 := T^{u+w_{i^*}}$, and $C_2 := M \cdot e(T, h_1)^\alpha$. If $T \in \mathbb{G}_1$, the ciphertext is normal. If $T \in \mathbb{G}_{12}$, the $\mathbb{G}_2$ exponent of $T$ plays the role of $x$, and the hidden residue $u + w_{i^*} \bmod p_2$ plays the role of $\mu$. The last component is correct in both cases since $h_1$ pairs trivially with the $\mathbb{G}_2$ part of $T$.

*Step 3: Making the key semi-functional.* We next add $\mathbb{H}_2$ components to the public column of the master public key corresponding to index $i^*$, i.e., we set

$$P_{i^*} = h_1^{r_{i^*}} \boxed{h_2^{r'}}, \qquad P_{a,i^*} = h_1^{w_a r_{i^*}} \boxed{h_2^{w_a r'}} \quad (a \neq i^*),$$

for $r' \leftarrow\!\!\$\ \mathbb{Z}_{p_2}$. We refer to the transformed public key as type-$i^*$ public key. Simultaneously, we change the batch key for the queried batch $\Gamma$ to its semi-functional form, i.e.,

$$\widetilde{K}_\Gamma = h_1^{\alpha + uR_\Gamma} \boxed{h_2^{ur'}}.$$

This transition is indistinguishable under the subgroup decision assumption between $\mathbb{H}_1$ and $\mathbb{H}_{12}$ (Assumption 3.2). In particular, given a challenge $T$ from either $\mathbb{H}_1$ or $\mathbb{H}_{12}$, the reduction sets

$$P_{i^*} := T, \qquad P_{a,i^*} := T^{w_a} \quad (a \neq i^*)$$

by sampling $\{w_a\}_{a\in[n]/\{i^*\}} \leftarrow\!\!\$\ \mathbb{Z}_N$. It additionally samples $\alpha, u, w_{i*}$, and $\{r_j\}_{j\neq i^*}$ uniformly from $\mathbb{Z}_N$ and generates all other public-key elements as in the construction. The $\mathbb{H}_1$ exponent of $T$ plays the role of $r_{i^*}$, so the reduction need not know this exponent to generate the column. Since $i^*$ was guessed in Step 1, the reduction can generate this public key at the beginning of the experiment, before receiving any key query or the challenge messages.

To answer the single key query for a batch $\Gamma$, recall that $i^* \notin \Gamma$ whenever the reduction does not abort. Thus $P_{i^*} = T$ is a factor of the digest $D_\Gamma$. The reduction knows $\alpha$ and $u$, so it computes the key as

$$\begin{aligned}\widetilde{K}_\Gamma &:= h_1^\alpha D_\Gamma^u \\ &= h_1^\alpha \left(\prod_{j\notin\Gamma\cup\{i^*\}} P_j\right)^u T^u \\ &= h_1^{\alpha+u\sum_{j\notin\Gamma\cup\{i^*\}} r_j} T^u.\end{aligned}$$

Every factor other than $T$ belongs to $\mathbb{H}_1$ and is generated using an exponent known to the reduction.

The reduction must also simulate the semi-functional challenge ciphertext from Step 2. For this purpose, the subgroup decision instance (Assumption 3.2) supplies an independent uniform element $X = g_1^s g_2^x \in \mathbb{G}_{12}$. The reduction sets

$$C_0 := X, \qquad C_1 := X^{u+w_{i^*}}, \qquad C_2 := M \cdot e(X, h_1)^\alpha.$$

As in Step 2, the hidden residue $u + w_{i^*} \bmod p_2$ plays the role of $\mu$, and $C_1 = g_1^{s(u+w_{i^*})} g_2^{x\mu}$. The last component equals $M\Omega^s$ because $h_1$ pairs trivially with the $\mathbb{G}_2$ component of $X$. Thus the reduction can generate the ciphertext without knowing $s$ or $x$.

If $T = h_1^{r_{i^*}} \in \mathbb{H}_1$, the public column is normal and $T^u = h_1^{ur_{i^*}}$. Substituting this into the key computation gives $\widetilde{K}_\Gamma = h_1^{\alpha+uR_\Gamma}$, so the reduction simulates the preceding hybrid with a normal key and a semi-functional ciphertext. If $T = h_1^{r_{i^*}} h_2^{r'} \in \mathbb{H}_{12}$, the public column has exactly the type-$i^*$ distribution of [13] displayed above. In this case, $T^u = h_1^{ur_{i^*}} h_2^{ur'}$, so the same key computation gives

$$\widetilde{K}_\Gamma = h_1^{\alpha+uR_\Gamma} h_2^{ur'}.$$

The challenge ciphertext is semi-functional in both cases, while the public column and the single queried key acquires their $\mathbb{H}_2$ component in the second case.

It remains to explain why the added key exponent $ur'$ is uniform independently of the public column and the ciphertext's semi-functional exponents modulo $p_2$ in the second case. By the Chinese Remainder Theorem, we may fix all residues modulo $p_1$ without affecting the independent uniform residues of $u$ and $w_{i^*}$ modulo $p_2$. We may also fix $r'$ and all residues $w_a \bmod p_2$ for $a \neq i^*$, which determine the added $\mathbb{H}_2$ components of the public column. The

diagonal entry $P_{i^*,i^*}$ is missing, so this column contains no term involving $w_{i^*} \bmod p_2$. The other entries in row $i^*$ belong to $\mathbb{H}_1$ and therefore depend only on $w_{i^*} \bmod p_1$. Likewise, the public element $g_1^u$ depends only on $u \bmod p_1$. Thus the public key places no restriction on either of these two residues modulo $p_2$.

The ciphertext's semi-functional exponent is $x\mu$, where $\mu = u + w_{i^*} \bmod p_2$. For each fixed value of $\mu$ and each choice of $u \bmod p_2$, there is exactly one value $w_{i^*} = \mu - u \bmod p_2$. Since $u$ and $w_{i^*}$ were sampled independently and uniformly modulo $p_2$, fixing $\mu$ leaves $u \bmod p_2$ uniform. If $r' \neq 0$, multiplication by $r'$ is a permutation of $\mathbb{Z}_{p_2}$, so $ur'$ is also uniform after fixing the public column, $x$ and $\mu$. This holds even when $x = 0$. The only exceptional case is $r' = 0$, which occurs with probability $1/p_2$ and contributes a negligible statistical difference to the next step. Thus, the public column receives its type-$i^*$ form and the single queried key takes its semi-functional form in the second case.

*Step 4: Relabeling the hidden key randomness.* We next replace the single queried key by

$$\widetilde{K}_\Gamma = h_1^{\alpha + uR_\Gamma} \boxed{h_2^{\alpha' + ur'}}.$$

Here $\alpha' \leftarrow\$ \mathbb{Z}_{p_2}$ is sampled at the beginning of the experiment independently of all other randomness. The master public key retains its type-$i^*$ form, and the challenge ciphertext remains semi-functional. This transition is statistical, with distance at most $1/p_2$ between the adversary's views in the two hybrids.

In particular, the new hybrid generates the same public-key elements as in Step 3 and answers the key query for $\Gamma$ by computing

$$\widetilde{K}_\Gamma := h_1^\alpha D_\Gamma^u h_2^{\alpha'}.$$

Since $i^* \notin \Gamma$, the digest contains the modified element $P_{i^*}$ from the type-$i^*$ public column and has the form $D_\Gamma = h_1^{R_\Gamma} h_2^{r'}$. Raising this digest to $u$ and multiplying by $h_1^\alpha h_2^{\alpha'}$ gives precisely the form of the key displayed above. The hybrid uses this response whether the query occurs before or after the challenge. The challenge ciphertext components remain $C_0 = g_1^s g_2^x$, $C_1 = g_1^{s(u+w_{i^*})} g_2^{x(u+w_{i^*})}$, and $C_2 = M\Omega^s$.

To compare the joint distributions of the public column, challenge ciphertext, and the queried key, first fix $r' \neq 0$ and all residues modulo $p_1$. As explained in Step 3, the public column does not involve $u \bmod p_2$ or $w_{i^*} \bmod p_2$, and the ciphertext's semi-functional components involve these residues only through their sum $\mu = u + w_{i^*} \bmod p_2$. Thus we can add the independent exponent $\alpha'$ to the key while compensating for it through a change of these two hidden residues. Starting with the randomness of the preceding hybrid, sample $\alpha' \leftarrow\$ \mathbb{Z}_{p_2}$ and make the substitutions

$$u \longmapsto u - \alpha'(r')^{-1}, \qquad w_{i^*} \longmapsto w_{i^*} + \alpha'(r')^{-1} \pmod{p_2}.$$

The inverse $(r')^{-1}$ exists because $r' \neq 0$. For each fixed $r'$ and $\alpha'$, the two substitutions are bijections on $\mathbb{Z}_{p_2}$. Consequently, the resulting residues of $u$ and $w_{i^*} \bmod p_2$ are still independent and uniform, and are independent of $\alpha'$, as required in the new hybrid.

Now for the semi-functional component of the key with this substitution, we have

$$\alpha' + \big(u - \alpha'(r')^{-1}\big)r' = ur' \pmod{p_2},$$

so the key returned by the new hybrid is exactly the key returned by the preceding hybrid. For the ciphertext, the two substitutions preserve the sum

$$\big(u - \alpha'(r')^{-1}\big) + \big(w_{i^*} + \alpha'(r')^{-1}\big) = u + w_{i^*} \pmod{p_2}.$$

Hence the semi-functional component $g_2^{x(u+w_{i^*})}$ of $C_1$ is unchanged. The component $C_0$ and the normal component of $C_1$ are unchanged as well, and $C_2 = M\Omega^s$ is the same for each choice of the challenge message $M$. This argument applies to every $x$, including $x = 0$.

Thus the views of the adversary in the two neighboring hybrids are identically distributed conditioned on $r' \neq 0$. If $r' = 0$, the original semi-functional key exponent is zero, whereas the new exponent is $\alpha'$, so we do not claim equality in this case. This event occurs with probability $1/p_2$, which bounds the statistical distance without conditioning.

*Step 5: Restoring the public key.* We next remove the $\mathbb{H}_2$ components from the public column indexed by $i^*$, restoring its elements to the normal form:

$$P_{i^*} = h_1^{r_{i^*}}, \qquad P_{a,i^*} = h_1^{w_a r_{i^*}} \quad (a \neq i^*).$$

At the same time, we remove the contribution $ur'$ from the semi-functional exponent of the single queried key, so that the key becomes

$$\widetilde{K}_\Gamma = h_1^{\alpha + uR_\Gamma} \boxed{h_2^{\alpha'}}.$$

The challenge ciphertext remains semi-functional as in the preceding hybrid. This transition is indistinguishable under the two-prime analogue of the subgroup decision assumption (Assumption 3.2), with target distributions $\mathbb{H}_1$ and $\mathbb{H}_{12}$. The reduction is very similar to the transition in Step 3 in the reverse order.

*Step 6: Hiding the message.* We finally replace the mask $\Omega^s$ in the challenge ciphertext by an independent uniform element $R \leftarrow\!\!\$\ \mathbb{G}_T$, giving the challenge ciphertext the form

$$\widetilde{\mathsf{ct}}_{i^*} = \big(g_1^s g_2^x,\ g_1^{s(u+w_{i^*})} g_2^{x\mu},\ M \cdot \boxed{R}\big).$$

The master public key remains normal, and the single queried key retains its semi-functional form from Step 5. This transition is indistinguishable under the two-prime analogue of the pairing assumption (Assumption 3.3).

In particular, the reduction receives $g_1, h_1$ and the auxiliary elements

$$X = g_1^s g_2^x, \qquad Y = h_1^\alpha h_2^{\alpha'},$$

together with a challenge $T$ that is either $e(g_1, h_1)^{\alpha s}$ or an independent uniform element of $\mathbb{G}_T$. To generate the master public key, the reduction samples $u, \{w_i, r_i\}_{i \in [n]} \leftarrow\!\!\$\ \mathbb{Z}_N$ and computes

$$\Omega := e(g_1, Y) = e(g_1, h_1^\alpha h_2^{\alpha'}) = e(g_1, h_1)^\alpha.$$

The last equality follows from orthogonality, since $g_1$ pairs trivially with the $\mathbb{H}_2$ component of $Y$. It generates the remaining public elements $g_1^u$, $\{g_1^{w_i}\}_{i\in[n]}$, $P_j = h_1^{r_j}$, and $P_{i,j} = h_1^{w_i r_j}$ for $i \neq j$ using the exponents it sampled. Every public column therefore has its normal distribution, as required after Step 5.

To answer the single admissible key query for a batch $\Gamma$, the reduction computes the normal digest $D_\Gamma = h_1^{R_\Gamma}$ and returns

$$\begin{aligned}\widetilde{K}_\Gamma &:= Y D_\Gamma^u \\ &= (h_1^\alpha h_2^{\alpha'}) h_1^{uR_\Gamma} \\ &= h_1^{\alpha + uR_\Gamma} h_2^{\alpha'}.\end{aligned}$$

The reduction knows $u$ and can compute $D_\Gamma$ from the public key, so it can form this response directly from $Y$. Since the pairing instance samples $\alpha'$ independently of the other exponents, this response has exactly the semi-functional key distribution from Step 5. The same computation answers a query before or after the challenge, without requiring the reduction to predict $\Gamma$ at setup.

When $\mathcal{A}$ submits its challenge messages $M_0, M_1$ and the already guessed identity $i^*$, the reduction forms the ciphertext for the experiment's challenge bit $b$ as

$$\widetilde{\mathsf{ct}}_{i^*} = \big(C_0 := X,\ C_1 := X^{u+w_{i^*}},\ C_2 := M_b \cdot T\big).$$

The first component is $C_0 = g_1^s g_2^x$. The second component expands to

$$C_1 = g_1^{s(u+w_{i^*})} g_2^{x(u+w_{i^*})}.$$

By the Chinese Remainder Theorem, the residues of $u$ and $w_{i^*}$ modulo $p_2$ are independent of their residues modulo $p_1$, which determine their contributions to the normal public key and key component. Their sum $\mu = u + w_{i^*} \bmod p_2$ is therefore uniform and independent of the supplied semi-functional key exponent $\alpha'$. Thus $C_0$ and $C_1$ have the required semi-functional distribution.

If $T = e(g_1, h_1)^{\alpha s}$, then $T = \Omega^s$ and $C_2 = M_b \Omega^s$. Together with the normal master public key and the semi-functional key computed above, this gives exactly the preceding hybrid. If $T$ is independent and uniform in $\mathbb{G}_T$, the same public key, key response, and first two ciphertext components are generated, while $C_2 = M_b T$ uses an independent uniform mask. This gives exactly the new hybrid.

It remains to explain why the new hybrid hides the challenge bit even when the messages are chosen adaptively. Fix the setup randomness, the adversary's randomness, and its view before the challenge, including the key response if it has already made its key query. These values determine the messages $M_0, M_1$ chosen by the adversary, while the independent mask $R$ remains uniform because it has not been used in any preceding response. We may therefore replace $C_2 = M_b R$ by a fresh uniform group element without changing its joint distribution with the setup values, the preceding view, or $C_0, C_1$. After this replacement, none of the challenge components depends on $b$. Thus the adversary's final view has the same distribution for both challenge bits.

Combining this final independence with the preceding hybrid transitions, the statistical difference of at most $1/p_2$ in Step 4, and the polynomial guessing loss from Step 1 proves one-key security. Here we have isolated the one-key case because the same argument based on the missing diagonal entry will handle the key query on the challenge epoch in the next epoch-based construction.

*Remark* 2.1 (On multi-key security). Gong et al. [14] show that their one-key epochless scheme breaks once the adversary obtains two batch keys. Our epochless scheme can, however, be shown adaptively secure against polynomially many batch-key queries. The one-key argument above extends because every admissible set omits the challenge identity $\mathsf{id}^*$, so the factor $P_{\mathsf{id}^*}$ occurs in every digest and a single type-$\mathsf{id}^*$ column shifts every key by the same hidden exponent $ur'$. As in the one-key case, this shared exponent is uniform given the view, so it can be relabeled as $\alpha' + ur'$ for a fresh $\alpha'$ shared by all keys. Finally, the type-$\mathsf{id}^*$ column is removed. When its $\mathbb{H}_2$ component disappears, the $ur'$ term disappears from every key, leaving all batch keys in the semi-functional form $h_1^{\alpha+uR_\Gamma} h_2^{\alpha'}$. The final pairing assumption then embeds this global hidden $\alpha'$ in all keys and replaces the challenge mask by a random target group element. Thus the same scheme is also multi-key secure, with a similar loss factor of $n$. However, this doesn't carry over to our epoch-based scheme discussed below. The reason is that if we allow multiple queries for the challenge epoch, the reduction will have to guess *all* the key queries out of $\{1, \ldots, Q\}$ that correspond to the challenge label, since we would need to simultaneously switch all of those queries to semi-functional. But this will lead to exponential loss in the security reduction.

## 2.3 Adding Epochs via the Lewko–Waters IBE Layer

We next add epochs to the above epochless construction by using the affine expression $\beta + \gamma\tau$ for epoch $\tau$ from the Lewko–Waters IBE scheme [19]. A ciphertext is now associated with an identity $i$ and an epoch $\tau$, and a key for $(\Gamma, \tau)$ decrypts ciphertexts at that epoch for identities in $\Gamma$. The adversary may obtain keys for batches containing the challenge identity $i^*$ at other epochs, so the one-key argument alone no longer suffices. We use the affine expression to handle these queries, while the missing diagonal entry of the public table of the master public key will handle the single possible key query at the challenge epoch as before.

We now work in composite-order bilinear groups of order $N = p_1p_2p_3$. We retain the identity space $[n]$ and take the epoch space to be $\mathcal{T} = \mathbb{Z}_N$, with epochs represented as canonical residues modulo $N$. The public table in the master public key and user keys include independent random elements of $\mathbb{H}_3$, which provides additional randomness required for proving the adaptive security with multiple keys in non-challenge epochs.

**The construction.** For clarity of presentation, here we highlight the additional factors and components introduced to support epochs to our epoch-less construction presented in §2.2 above with a light-gray background. Let $g_1 \in \mathbb{G}_1$ and $h_1 \in \mathbb{H}_1$ be generators, and sample $\alpha, u, \beta, \gamma, \{w_i, r_i\}_{i\in[n]} \leftarrow\$ \mathbb{Z}_N$. For each column $j \in [n]$ and each off-diagonal entry $(i, j)$ with

$i, j \in [n]$, $i \neq j$, sample independent uniform elements $R_{3,j}, R_{3,i,j} \leftarrow\!\$\ \mathbb{H}_3$ and set

$$P_j := R_{3,j}\, h_1^{r_j},$$
$$P_{i,j} := R_{3,i,j}\, h_1^{w_i r_j} \qquad (i \neq j).$$

The master public key is

$$\mathsf{pk} = \big(g_1, g_1^u, \{g_1^{w_i}\}_{i\in[n]}, \Omega = e(g_1,h_1)^\alpha, \{P_j\}_{j\in[n]}, \{P_{i,j}\}_{i\neq j}, g_1^\beta, g_1^\gamma, h_1^\beta, h_1^\gamma\big).$$

The additional elements $g_1^\beta, g_1^\gamma$ allow encryption to compute $g_1^{\beta+\gamma\tau}$, and $h_1^\beta, h_1^\gamma$ give the corresponding expression in the other source group. For a batch $\Gamma$, we continue to use the digest $D_\Gamma = \prod_{j\notin\Gamma} P_j$ and $R_\Gamma = \sum_{j\notin\Gamma} r_j$. The $\mathbb{H}_1$ component of $D_\Gamma$ is $h_1^{R_\Gamma}$, and its remaining component belongs to $\mathbb{H}_3$.

To encrypt $M \in \mathbb{G}_T$ for $(i,\tau)$, sample $s \leftarrow\!\$\ \mathbb{Z}_N$. The ciphertext $\mathsf{ct}_{i,\tau} = (C_0, C_1, C_2, C_3)$ has components

$$C_0 = g_1^s, \qquad C_1 = g_1^{s(u+w_i)},$$
$$C_2 = g_1^{s(\beta+\gamma\tau)}, \quad C_3 = M\Omega^s.$$

The component $C_2$ is new, while the message component of the epochless ciphertext is now indexed $C_3$ and retains the value $M\Omega^s$. To generate a key for $(\Gamma,\tau)$, sample $\rho \leftarrow\!\$\ \mathbb{Z}_N$ and independent uniform $R_3, R_3' \leftarrow\!\$\ \mathbb{H}_3$ and output $\mathsf{sk}_{\Gamma,\tau} = (K_0, K_1)$, where

$$K_0 = R_3 h_1^\rho,$$
$$K_1 = R_3'\, h_1^{\alpha+uR_\Gamma}\, h_1^{(\beta+\gamma\tau)\rho}.$$

Equivalently, the second component $K_1$ can be computed as $R_3' h_1^\alpha D_\Gamma^u (h_1^\beta (h_1^\gamma)^\tau)^\rho$. The $\mathbb{H}_3$ component of the digest is multiplied by the fresh uniform factor $R_3'$, so this computation has the same distribution as the displayed key.

To decrypt, first check that $i \in \Gamma$ and that the ciphertext and key epochs agree, and reject otherwise. For matching epochs, compute $E_{i,\Gamma} = \prod_{j\notin\Gamma} P_{i,j}$ and evaluate

$$e(C_0,K_1)e(C_1,D_\Gamma)^{-1}e(C_0,E_{i,\Gamma})\, e(C_2,K_0)^{-1}$$
$$= e(g_1,h_1)^{\alpha s + s(\beta+\gamma\tau)\rho - s(\beta+\gamma\tau)\rho} = \Omega^s.$$

The batching terms cancel as in the epochless construction. The additional pairing with $K_0$ cancels the term contributed by the epoch expression in $K_1$, and every $\mathbb{H}_3$ factor pairs trivially with the ciphertext components in $\mathbb{G}_1$. Dividing $C_3$ by the resulting mask recovers $M$.

**Adaptive security analysis.** The proof again makes the challenge ciphertext and the queried keys semi-functional before replacing the message mask by a uniform group element.

There are now two kinds of key queries to consider. For a query at an epoch different from $\tau^*$, the values of the affine expressions $\beta + \gamma\tau$ and $\beta + \gamma\tau^*$ at the queried epoch $\tau$ and at $\tau^*$ respectively, provide the independence needed for the transition. At the challenge epoch, the admissibility condition instead gives $i^* \notin \Gamma$, so we use the missing diagonal entry as in the epochless proof. Throughout the following hybrids, all residues appearing in semi-functional components are understood modulo $p_2$.

*Step 1: Guessing the challenge identity and the challenge-epoch query.* Let $Q = Q(\lambda)$ be a polynomial upper bound on the number of key queries. Before setup, the reduction guesses the challenge identity $i^* \in [n]$ and an index $\widehat{j} \in \{1, \ldots, Q, \perp\}$ of the key query which would be the allowed single key query for the challenge epoch, where $\widehat{j} = \perp$ means that no query at the challenge epoch occurs. The epoch space is exponential, so we guess the query's position rather than the value of its epoch to avoid any exponential security loss. When the guessed query arrives, the reduction answers it using the batch and epoch actually supplied by the adversary. If it arrives before the challenge, these values are recorded and checked once the challenge tag $\tau^*$ is known. The reduction aborts if the challenge identity differs from its guess $i^*$, if the guessed key query is not the unique query at $\tau^*$, or if that query's batch contains $i^*$. It also aborts if a guessed key query index never occurs, or if $\widehat{j} = \perp$ and a query at $\tau^*$ does occur. For each complete transcript of the real experiment, exactly one of the $n(Q+1)$ possible pairs of guesses is correct. Thus this step incurs a polynomial loss of $n(Q+1)$, extending Step 1 of the epochless analysis in §2.2.

*Step 2: Excluding epoch collisions modulo $p_2$.* We next restrict the game so that every query at an epoch different from $\tau^*$ satisfies $\tau_k \not\equiv \tau^* \pmod{p_2}$. The game aborts if this condition fails, checking earlier queries when the challenge arrives and later queries as they are made. For distinct epochs $\tau_k$ and $\tau^*$ that agree modulo $p_2$, the value

$$d := \gcd(\tau_k - \tau^*, N)$$

is a proper divisor of $N$ divisible by $p_2$. Since we select a composite order group whose order $N$ is hard to factor, this means that the epochs $\tau_k$ for the key queries of the adversary other than the challenge epoch $\tau^*$ will match the challenge epoch $\tau^* \bmod p_2$ only with negligible probability. Thus putting the restriction that the epochs of the key queries different from $\tau^*$ will match $\tau^* \bmod p_2$ can only change the adversary's advantage by a negligible amount. It can also be shown why finding such a pair with non-negligible probability would contradict the three-prime subgroup decision assumption (Assumption 3.2) in a similar fashion as [19].

*Step 3: Making the challenge ciphertext semi-functional.* We next replace the challenge ciphertext by $\widetilde{\mathsf{ct}}_{i^*,\tau^*} = (C_0, C_1, C_2, C_3)$ where

$$\begin{aligned}
C_0 &= g_1^s \boxed{g_2^x}, \\
C_1 &= g_1^{s(u+w_{i^*})} \boxed{g_2^{x\mu}}, \\
C_2 &= g_1^{s(\beta+\gamma\tau^*)} \boxed{g_2^{xz_c}}, \\
C_3 &= M\Omega^s,
\end{aligned}$$

where $x, \mu, z_c \leftarrow\$ \mathbb{Z}_{p_2}$. The public key and all queried keys remain normal. This transition is essentially a straight forward extension of Step 2 of the epochless analysis in §2.2, under the three-prime subgroup decision assumption between $\mathbb{G}_1$ and $\mathbb{G}_{12}$ (Assumption 3.1).

*Step 4: Making keys at other epochs semi-functional.* We next change the keys at epochs different from $\tau^*$ to semi-functional form one at a time, in the order in which they are queried. The key at the guessed query position $\widehat{j}$, if present, remains normal throughout this step. For a query $(\Gamma_k, \tau_k)$ that we have changed, the key has the form $\mathsf{sk}_{\Gamma_k,\tau_k} = (\widetilde{K}_{0,k}, \widetilde{K}_{1,k})$ where

$$\widetilde{K}_{0,k} = R_{3,0,k} h_1^{\rho_k} \boxed{h_2^{y_k}},$$
$$\widetilde{K}_{1,k} = R_{3,1,k} h_1^{\alpha + uR_{\Gamma_k} + (\beta+\gamma\tau_k)\rho_k} \boxed{h_2^{c_k}},$$

where $y_k, c_k \leftarrow\$ \mathbb{Z}_{p_2}$ are independent for each key and independent of the challenge ciphertext's semi-functional exponents. The factors $R_{3,0,k}, R_{3,1,k} \leftarrow\$ \mathbb{H}_3$ are also fresh and independent.

Fix the query index $k$ whose response is changed in the current transition. The reduction receives an instance of the subgroup decision assumption between $\mathbb{H}_{13}$ and $\mathbb{H}_{123}$ (Assumption 3.2), including $X_{12} \leftarrow\$ \mathbb{G}_{12}$, $W_{23} \leftarrow\$ \mathbb{H}_{23}$, and a target $T$ from either $\mathbb{H}_{13}$ or $\mathbb{H}_{123}$. It samples all setup exponents, generates the normal public key with fresh $\mathbb{H}_3$ masks, and uses $X_{12}$ to generate the semi-functional challenge ciphertext, similar to how it is done in Step 3 of §2.2. For the query $(\Gamma_k, \tau_k)$, it returns $\mathsf{sk}_{\Gamma_k,\tau_k} = (K_{0,k}, K_{1,k})$ where

$$K_{0,k} = R_{3,0,k} T, \qquad K_{1,k} = R_{3,1,k} h_1^{\alpha + uR_{\Gamma_k}} T^{\beta+\gamma\tau_k}.$$

Writing the $\mathbb{H}_1$ component of $T$ as $h_1^{\rho_k}$, these expressions have the required normal components. The independent masks make the two $\mathbb{H}_3$ components uniform even though both expressions use $T$.

For every query index $\ell$ changed in an earlier hybrid, the reduction samples fresh $\rho_\ell, a_\ell, b_\ell \leftarrow\$ \mathbb{Z}_N$ and returns $\mathsf{sk}_{\Gamma_\ell,\tau_\ell} = (K_{0,\ell}, K_{1,\ell})$ where

$$K_{0,\ell} = R_{3,0,\ell} h_1^{\rho_\ell} W_{23}^{a_\ell},$$
$$K_{1,\ell} = R_{3,1,\ell} h_1^{\alpha + uR_{\Gamma_\ell} + (\beta+\gamma\tau_\ell)\rho_\ell} W_{23}^{b_\ell}.$$

To verify their distribution, write $W_{23} = h_2^{\eta} h_3^{\theta}$ for the analysis. If $\eta \neq 0 \bmod p_2$, the exponents $\eta a_\ell$ and $\eta b_\ell$ are independent uniform residues modulo $p_2$, independently across all such queries. The fresh $\mathbb{H}_3$ masks absorb the $\mathbb{H}_3$ components of these powers. Thus these responses have the required semi-functional distribution and reveal nothing about $\beta, \gamma \bmod p_2$. All later queries and the challenge-epoch query, if present, are answered normally using the known setup exponents.

If $T \in \mathbb{H}_{13}$, the response at index $k$ is normal, so the reduction simulates the preceding hybrid. If $T \in \mathbb{H}_{123}$, write its $\mathbb{H}_2$ component as $h_2^{y_k}$. The added exponents in that response are

$$y_k, \qquad y_k z_k, \qquad z_k = \beta + \gamma\tau_k \bmod p_2.$$

We show that $z_k$ is uniform independently of the ciphertext's exponent $z_c = \beta + \gamma\tau^* \bmod p_2$, including when the query and challenge epochs are chosen adaptively.

Suppose first that this query occurs after the challenge. Fix the preceding view, the value $z_c$, and the queried epoch $\tau_k$. The change of variables $(\beta, \gamma) \mapsto (z_c, \gamma)$ is invertible modulo $p_2$. The public key and normal responses depend only on the residues of $\beta, \gamma$ modulo $p_1$, while earlier semi-functional responses use the independent powers $a_\ell, b_\ell$ of $W_{23}$. Consequently, after fixing $z_c$, the residue $\gamma$ mod $p_2$ remains uniform. Since Step 2 ensures $\tau_k \neq \tau^*$ mod $p_2$, the expression

$$z_k = z_c + \gamma(\tau_k - \tau^*) \bmod p_2$$

is uniform.

If the query occurs before the challenge, use the change of variables $(\beta, \gamma) \mapsto (z_k, \gamma)$ instead. At the time of the query, $z_k$ and $\gamma$ are independent uniform residues. The key query response depends only on $z_k$, and subsequent normal responses reveal nothing about $\gamma$ mod $p_2$. Thus, after fixing $z_k$ and the later chosen epoch $\tau^*$, the expression

$$z_c = z_k + \gamma(\tau^* - \tau_k) \bmod p_2$$

is uniform for the same reason. The simulation therefore works in either order without knowing a future query or challenge epoch.

Finally, for every nonzero $y_k$, multiplication by $y_k$ preserves uniformity modulo $p_2$. It follows that $c_k = y_k z_k$ is uniform independently of $y_k$, $z_c$, and the semi-functional exponents of earlier keys. The events $y_k = 0$ and $\eta = 0$ each have probability $1/p_2$. Apart from these negligible events and the events in which a supplied subgroup element is the identity, the $\mathbb{H}_{123}$ case simulates the next hybrid. Repeating this transition for every other-epoch query makes all such keys semi-functional, including keys whose batches contain $i^*$.

*Step 5: Making the challenge-epoch key semi-functional.* If there is no query at the challenge epoch, all queried keys are already semi-functional, so we omit this step and Step 6. Otherwise, write $\Gamma_*$ for the batch in this query. We change the public column indexed by $i^*$ of the master public key to

$$P_{i^*} = R_{3,i^*} h_1^{r_{i^*}} \boxed{h_2^{r'}},$$
$$P_{a,i^*} = R_{3,a,i^*} h_1^{w_a r_{i^*}} \boxed{h_2^{w_a r'}} \quad (a \neq i^*),$$

where $r' \leftarrow\!\!\$\ \mathbb{Z}_{p_2}$. At the same time, the challenge-epoch key becomes $\mathsf{sk}_{\Gamma^*,\tau^*} = (\widetilde{K}_{0,*}, \widetilde{K}_{1,*})$ where

$$\widetilde{K}_{0,*} = R_{3,0,*} h_1^{\rho_*} \boxed{h_2^{y_*}},$$
$$\widetilde{K}_{1,*} = R_{3,1,*} h_1^{\alpha + u R_{\Gamma_*} + (\beta + \gamma\tau^*)\rho_*} \boxed{h_2^{c_*}},$$

with independent uniform $y_*, c_* \leftarrow\!\!\$\ \mathbb{Z}_{p_2}$. This is the analogous change as in Step 3 of §2.2.

*Step 6: Restoring the public key.* We restore the public column to

$$P_{i^*} = R_{3,i^*} h_1^{r_{i^*}}, \qquad P_{a,i^*} = R_{3,a,i^*} h_1^{w_a r_{i^*}} \quad (a \neq i^*),$$

while retaining the independently uniform semi-functional exponents in every key. This is analogous to Step 5 of §2.2, under the subgroup decision assumption (Assumption 3.2).

*Step 7: Relabeling the hidden key randomness.* We sample one global $\alpha' \leftarrow\$ \mathbb{Z}_{p_2}$ and replace each second key component by

$$\widetilde{K}_{1,k} = R_{3,1,k} h_1^{\alpha + uR_{\Gamma_k} + (\beta+\gamma\tau_k)\rho_k} \boxed{h_2^{\alpha'+c_k}}.$$

This step is analogous to Step 4 of §2.2.

*Step 8: Hiding the challenge message.* Finally, we replace the last ciphertext component by

$$C_3 = M \cdot T, \qquad T \leftarrow\$ \mathbb{G}_T.$$

This is essentially Step 6 of §2.2, under the pairing assumption (Assumption 3.3).

Putting these hybrids together gives adaptive security with polynomial loss in $n$ and $Q$, up to the negligible exceptional events identified above.

## 2.4 Extending to Subset Policies

Our next observation is that our epoch-based construction supports a more general decryption condition. Instead of associating a ciphertext with one identity, we associate it with a nonempty set $X \subseteq [n]$. A key is associated with a set $S \subseteq [n]$ and an epoch $\tau$ as before. It decrypts a ciphertext for $(X, \tau)$ when $X \subseteq S$. Thus every identity named by the ciphertext policy must belong to the key's set. Taking $X = \{i\}$ recovers the usual Batched IBE condition. We call this extension subset-policy Batched IBE and define it formally in Definition 3.2. The underlying universe remains $[n]$. Every key set $S \subseteq [n]$ is supported, including the empty set, and there is no additional bound on $|S|$ beyond $n$.

The adaptive security definition changes accordingly. The adversary chooses a nonempty challenge set $X^*$ and an epoch $\tau^*$. It may ask for arbitrary sets at other epochs. There is still at most one query at epoch $\tau^*$, and its set $S^*$ must satisfy $X^* \not\subseteq S^*$. In particular, this set may contain some of the challenge identities, provided it omits at least one.

**The Construction.** The extension requires only a change to the identity-dependent ciphertext component. Define

$$W_X := \sum_{i \in X} w_i.$$

We replace $C_1 = g_1^{s(u+w_i)}$ by

$$C_1 = \left( g_1^u \prod_{i \in X} g_1^{w_i} \right)^s = g_1^{s(u+W_X)}.$$

The other ciphertext components and the algebraic form of the key stay the same. In particular, with $R_S = \sum_{j \notin S} r_j$, the key is

$$\mathsf{sk}_{S,\tau} = (K_0, K_1) = \left( R_3 h_1^{\rho}, R_3' h_1^{\alpha + uR_S + (\beta+\gamma\tau)\rho} \right).$$

The public products used by the decryptor become

$$D_S = \prod_{j \notin S} P_j, \qquad E_{X,S} = \prod_{i \in X} \prod_{j \notin S} P_{i,j}.$$

If $X \subseteq S$, every pair $(i, j)$ in the second product satisfies $i \neq j$. Therefore all its factors are available in the off-diagonal public table included in the master public key. Ignoring the $\mathbb{H}_3$ components, which disappear under pairing with $\mathbb{G}_1$, we have $D_S = h_1^{R_S}$ and $E_{X,S} = h_1^{W_X R_S}$. The same decryption formula therefore gives

$$\frac{e(C_0, K_1)e(C_0, E_{X,S})}{e(C_1, D_S)e(C_2, K_0)} = e(g_1, h_1)^{\alpha s}.$$

The only difference from the earlier cancellation is that $W_X$ replaces $w_i$. In particular, combining several identity exponents does not add group elements to the ciphertext or key. The master public key still has $O(n^2)$ group elements, while ciphertexts, digests, and batch keys still have four, one, and two group elements, respectively.

**Why Correctness and Security are Preserved.** For correctness, observe that $e(C_1, D_S)$ now contains the exponent $s(u + W_X)R_S$. The term $suR_S$ cancels with the corresponding term in $e(C_0, K_1)$, while $sW_X R_S$ cancels with the contribution of $e(C_0, E_{X,S})$. The condition $X \subseteq S$ ensures that every entry needed for this product is available. When the epochs match, $e(C_2, K_0)$ also cancels the key's epoch-dependent term, leaving precisely $e(g_1, h_1)^{\alpha s}$.

For security, we need to show that including several identities in the ciphertext does not reveal the randomness used in the single-identity proof. Let $S^*$ be the set in the challenge-epoch query, or the empty set if there is no such query. Since $X^* \not\subseteq S^*$ and $X^*$ is nonempty, there is at least one identity in the challenge policy that is absent from $S^*$. We select one deterministically by setting

$$i^\dagger := \min(X^* \setminus S^*).$$

Crucially, the proof guesses this identity and as in the single-identity case above, guesses the position of the challenge-epoch key query, including the possibility that no such query occurs. As before, there are only $n(Q + 1)$ possible pairs when the adversary makes at most $Q$ key queries. Thus the guessing loss remains polynomial even though the adversary can choose among exponentially many policies.

Suppose that a challenge-epoch key is queried. We introduce a semi-functional public column in the master public key at position $i^\dagger$ and use it to change this key, as in the single-identity proof. The column contains an entry for every row except $i^\dagger$ itself. Its entries may reveal the residues $w_a \bmod p_2$ for all $a \neq i^\dagger$, so we account for this information by fixing those residues, including the ones for the other identities in $X^*$. However, the diagonal entry $P_{i^\dagger, i^\dagger}$ is absent, and the remaining public elements involving $w_{i^\dagger}$ depend only on its residue modulo $p_1$. By the Chinese Remainder Theorem, they do not constrain $w_{i^\dagger} \bmod p_2$.

To see why this remaining randomness suffices, write

$$W_- := \sum_{a \in X^* \setminus \{i^\dagger\}} w_a.$$

Recall that $x$ is the $\mathbb{G}_2$ exponent in $C_0$, $r'$ is the $\mathbb{H}_2$ exponent in the modified pubic column of the public key, and $z_c = \beta + \gamma\tau^* \bmod p_2$. For fixed $W_-$ and $z_c$, the relevant ciphertext and key exponents are given by the map over $\mathbb{Z}_{p_2}$

$$\begin{aligned}&(u, w_{i\dagger}, \bar{\rho})\\ &\quad\longmapsto \big(x(u + W_- + w_{i\dagger}), \bar{\rho}r', (u + \bar{\rho}z_c)r'\big).\end{aligned}$$

Compared with the single-identity case, the only change is the fixed additive term $xW_-$ in the ciphertext exponent. For nonzero $x$ and $r'$, the map remains invertible. Indeed, similar to the single-identity proof above, the exponent of the first challenge-epoch key component determines $\bar{\rho}$, the exponent of the second component then determines $u$, and the ciphertext exponent determines $w_{i\dagger}$. Thus uniformly sampled input residues give jointly uniform ciphertext and key exponents, except for negligible zero-exponent events. In particular, even after fixing both key exponents, the unused residue $w_{i\dagger} \bmod p_2$ still makes the ciphertext exponent uniform. The other non-challenge-epoch key queries are handled exactly the same way as the security proof of the single-identity case above since those keys are mainly converted to semi-functional form through their affine tag exponent.

## 2.5 Supporting an Exponential Identity Space

So far, our construction supports a polynomial-size identity universe $[n]$. Its public parameters contain the off-diagonal entries $P_{i,j}$ for pairs of coordinates, and therefore grow quadratically with $n$. We cannot obtain an exponential identity space simply by increasing $n$, since this would also make the public parameters exponential. Our next observation is that the subset-policy functionality gives us a way to represent exponentially many identities using a polynomial number of coordinates. We fix a polynomial upper bound $B = B(\lambda) \geq 1$ on the size of any batch, as in earlier Batched IBE schemes [1, 14], and encode each identity as a subset of a smaller universe $[n]$. Here $B$ bounds the number of identities in one batch, while $Q$ bounds the total number of adversarial key queries. The parameter $n$ will now bound the number of coordinates used by the underlying construction, rather than the number of identities supported by the resulting Batched IBE scheme.

**Representing Identities by Sets.** For each identity $\mathsf{id}$, let $C_{\mathsf{id}} \subseteq [n]$ be its nonempty associated set. To encrypt for $\mathsf{id}$, we use $C_{\mathsf{id}}$ as the subset policy. To issue a key for a batch $\Gamma$, we use the union

$$S_\Gamma := \bigcup_{\mathsf{id}\in\Gamma} C_{\mathsf{id}}.$$

If $\mathsf{id} \in \Gamma$, then $C_{\mathsf{id}} \subseteq S_\Gamma$, so the subset-policy key decrypts the ciphertext. However, this implication alone is not enough. A set representing an identity outside $\Gamma$ might also be contained in the union, allowing the batch key to decrypt an unintended ciphertext. We therefore need a set family in which no set is covered by the union of at most $B$ other sets.

This is precisely the property of a $B$-cover-free family [10]. For every identity $\mathsf{id}$ and every

batch $\Gamma$ of at most $B$ identities that does not contain $\mathsf{id}$, the family satisfies

$$C_{\mathsf{id}} \not\subseteq \bigcup_{\mathsf{id}' \in \Gamma} C_{\mathsf{id}'}.$$

Combining this property with the earlier implication gives

$$\mathsf{id} \in \Gamma \quad \Longleftrightarrow \quad C_{\mathsf{id}} \subseteq S_\Gamma.$$

Thus, the subset-policy decryption condition expresses exactly the membership condition required by ordinary Batched IBE.

Our use of cover-free families is analogous to the transformation of Gong et al. [14, Remark C.15 in the full version], with the same representation of identities by blocks and batches by their unions, and the same classical cover-free family. However, their Appendix C instantiation is adaptive in the challenge identity but selective in the challenge epoch, and relies on a $q$-type assumption. Our underlying subset-policy scheme supports adaptive choices of both under static assumptions. We also formulate the compiler for the threshold setting, where partial-key computation and aggregation remain unchanged. We show that the compiler preserves adaptive challenge security under adaptive as well as static corruption of authorities, with no additional loss.

**An Explicit Cover-Free Family.** We can obtain the required family from the evaluation graphs of low-degree polynomials [10]. Let $q$ be the smallest power of two satisfying $q \geq (\lambda+1)B$, and identify the underlying universe $[n]$, where $n = q^2$, with $\mathbb{F}_q^2$. Associate each polynomial $f \in \mathbb{F}_q[Z]_{<\lambda}$ with the set

$$C_f := \{(a, f(a)) : a \in \mathbb{F}_q\}.$$

Each set contains $q$ points, one for every evaluation point $a$. Two distinct polynomials of degree below $\lambda$ agree at at most $\lambda - 1$ points. Therefore any other set intersects $C_f$ in at most $\lambda - 1$ coordinates, and a union of at most $B$ other sets covers at most $B(\lambda-1)$ coordinates of $C_f$. Since

$$B(\lambda - 1) < q = |C_f|,$$

at least one coordinate remains uncovered. This proves the cover-free property.

There are $q^\lambda \geq 2^\lambda$ possible polynomials, while the underlying universe has size

$$n = q^2 < 4(\lambda+1)^2 B^2 = O(\lambda^2 B^2).$$

The public description remains polynomial sized since it only needs to specify the field representation and the evaluation rule, rather than listing all the sets. An identity specifies the coefficients of its polynomial, from which its set can be computed efficiently. In particular, we may support every $\lambda$-bit identity by padding it to $\lambda \log_2 q$ bits and interpreting the result as the coefficient sequence of a polynomial. This encoding is injective and requires no hashing assumption. Restricting the family to these identities preserves cover-freeness.

**Security and the Parameter Tradeoff.** The compiler encrypts under $X = C_{\mathsf{id}}$, computes the digest for $S_\Gamma$, and otherwise uses the underlying key algorithm without modification. It

also preserves the epoch. An adversary's key query $(\Gamma, \tau)$ therefore becomes the subset-policy query $(S_\Gamma, \tau)$. If a query uses the challenge epoch and omits the challenge identity $\mathsf{id}^*$, cover-freeness ensures that $C_{\mathsf{id}^*} \not\subseteq S_\Gamma$. This is exactly the restriction in the subset-policy security experiment. Queries at other epochs remain unrestricted. The reduction can answer each query immediately, even if the challenge identity and epoch have not yet been chosen. It consequently preserves adaptive security with no additional loss in advantage. As in the underlying experiment, the challenger computes each digest from the requested set, and there is at most one key query at the challenge epoch.

The resulting scheme has an exponential identity space and polynomial public parameters. Its public key contains $O(n^2) = O(\lambda^4 B^4)$ group elements, and its master secret key contains $O(n)$ scalars and a constant number of group elements. Ciphertexts, digests, and batch keys retain their four, one, and two group elements, respectively. A ciphertext carries the identity and epoch as public metadata, rather than the entire set $C_{\mathsf{id}}$.

The batch bound therefore remains a parameter of setup. A batch bound fixed at setup also appears in all earlier pairing-based Batched IBE schemes, namely, Agarwal et al. [1] and Gong et al. [14]: the public parameters of the former depend on the batch-size bound, and the $q$-type assumption underlying the plain-model construction of the latter grows with it. The two works also rest on different security foundations, since Agarwal et al. prove security in the generic group and random oracle models, whereas our construction relies only on static assumptions in the plain model. The price we pay is a public table indexed by the polynomial underlying universe, giving a public key of $O(\lambda^4 B^4)$ group elements for an exponential identity space. By comparison, the public keys of Agarwal et al. [1] and the selectively secure plain-model construction of Gong et al. [14] contain $B + 2$ and $B + 6$ group elements, respectively, excluding common group parameters, so both grow only linearly with $B$. Our static assumptions do not grow with either $B$ or $Q$, and the epoch space remains $\mathbb{Z}_N$.

**Offline preprocessing for encryption.** The exponential identity space introduces public computation that depends on the batch bound $B$, but much of this work can be performed offline. Once an identity $\mathsf{id}$ is known, anyone can compute the set $C_{\mathsf{id}}$ and the associated products of public group elements before the message or epoch is chosen. With this preprocessing, online encryption requires a constant number of group operations. For encrypted mempools, a wallet can therefore prepare fresh transaction identifiers and the public digest of many such identifiers in advance, then complete encryption when submitting a transaction.

## 2.6 Extending to Threshold Batched IBE

We finally extend the subset-policy construction to the threshold setting by distributing batch-key computation among $L$ authorities. Following the centralized key generation interface of [1], key generation distributes private states so that each authority can independently compute a partial key from the public digest and epoch. For a reconstruction threshold $1 \le \theta \le L$, anyone can combine $\theta$ valid partial keys into a batch key, while confidentiality holds against at most $\theta - 1$ statically corrupted authorities. More precisely, the adversary fixes its corruption set before setup, but may choose the challenge policy, epoch, messages,

and key queries adaptively.

**Sharing the Master Secret.** Recall that, writing $T_\tau^H = h_1^\beta (h_1^\gamma)^\tau$, a nonthreshold batch key for $(S, \tau)$ has the form

$$\mathsf{sk}_{S,\tau} = (K_0, K_1) = \left(h_3^{z_0} h_1^\rho,\ h_3^{z_1} h_1^\alpha (T_\tau^H)^\rho D_S^u\right).$$

The secret contributions are determined by $\alpha$ and $u$, which occur linearly in the exponents. This allows us to share $\alpha$ and $u$ independently using Shamir sharing and reconstruct their contributions by interpolation in the exponent. We sample independent uniform polynomials $q_\alpha, q_u \in \mathbb{Z}_N[Z]$ of degree at most $\theta - 1$ with constant coefficients $\alpha$ and $u$, respectively. Authority $\ell$ receives $h_1^{\alpha_\ell}$ and $u_\ell$, where $\alpha_\ell = q_\alpha(\xi_\ell)$ and $u_\ell = q_u(\xi_\ell)$ at $\xi_\ell = \ell$. We have $L < \min\{p_1, p_2, p_3\}$, so these evaluation points and their nonzero pairwise differences are invertible modulo $N$. This is because $L$ is polynomial in $\lambda$ and all of the primes $p_1, p_2, p_3$ we select have bit-length at least $\Theta(\lambda)$. Thus Shamir interpolation is well defined even though $\mathbb{Z}_N$ is not a field.

Given $D_S$ and $\tau$, authority $\ell$ computes

$$\sigma_\ell = (K_{\ell,0}, K_{\ell,1}) = \left(h_3^{z_{\ell,0}} h_1^{\rho_\ell},\ h_3^{z_{\ell,1}} h_1^{\alpha_\ell} (T_\tau^H)^{\rho_\ell} D_S^{u_\ell}\right).$$

Each authority samples fresh randomizers $\rho_\ell, z_{\ell,0}, z_{\ell,1}$ for every partial key, as in nonthreshold key generation. The public table computation in master public key, encryption, and digest computation remain unchanged.

**Verification and Aggregation.** For any set $U$ of $\theta$ distinct authorities, let $\omega_\ell^U$ be the Lagrange coefficients for evaluation at zero. These coefficients reconstruct both $\alpha$ from the $\alpha_\ell$ and $u$ from the $u_\ell$. Applying them to the two partial-key components gives

$$\overline{\sigma} = (\overline{K}_0, \overline{K}_1) = \left(\prod_{\ell \in U} K_{\ell,0}^{\omega_\ell^U},\ \prod_{\ell \in U} K_{\ell,1}^{\omega_\ell^U}\right).$$

For honest partial keys, interpolation also combines the randomizers and subgroup masks, so the aggregate has exactly the form of a nonthreshold key and uses the same decryption algorithm. In particular, the batch key still contains only two group elements, independently of $L$ and $\theta$.

To handle malicious contributions, publish $A_\ell = e(g_1, h_1)^{\alpha_\ell}$ and $U_\ell = g_1^{u_\ell}$ for each authority. Writing $T_\tau^G = g_1^\beta (g_1^\gamma)^\tau$, a candidate partial key is accepted only if both components belong to $\mathbb{H}_N$ and

$$e(g_1, K_{\ell,1}) = A_\ell \, e(T_\tau^G, K_{\ell,0}) \, e(U_\ell, D_S).$$

Aggregation discards invalid candidates and interpolates $\theta$ accepted partial keys in the exponent, returning $\perp$ if too few are available. Interpolating the verification equations gives the pairing relation required by nonthreshold decryption, so even accepted malicious contributions cannot cause incorrect decryption. Availability separately requires at least $\theta$ responding honest authorities when the remaining authorities withhold their shares.

**Reducing to Non-threshold Security.** The main observation is that one non-threshold key suffices to simulate all honest partial keys for the same set and epoch, even given the corrupted authorities' private states. Let $C$ be the corruption set output by the adversary before setup. For this set, choose a polynomial $v_C$ of degree at most $\theta - 1$ that equals one at zero and vanishes at the corrupted evaluation points. We can express the sharing polynomials as $q_\alpha = \widetilde{q}_\alpha + \alpha v_C$ and $q_u = \widetilde{q}_u + u v_C$, where $\widetilde{q}_\alpha$ and $\widetilde{q}_u$ are independent uniform polynomials of degree at most $\theta - 1$ with zero constant coefficient. These are distributed as the original Shamir sharings, but the unknown master exponents disappear from the corrupted shares. The reduction can therefore generate the corrupted private states and all verification keys without knowing $\alpha$ or $u$.

For each query, the reduction obtains one nonthreshold key and raises its components to $v_C(\xi_\ell)$ for each honest authority $\ell$. It then supplies the known contributions from $\widetilde{q}_\alpha$ and $\widetilde{q}_u$ and independently rerandomizes each partial key. The resulting partial keys have the correct joint distribution, since fresh randomizers hide their common dependence on the nonthreshold key. Since the real honest authorities also sample fresh randomizers for every partial key, this simulation is perfect. As required by the threshold experiment, each query specifies the same set and epoch for all honest authorities. The challenge ciphertext and query authorization conditions are unchanged, so adaptive challenge security under static corruption follows from the security of the underlying nonthreshold scheme.

Finally, the same cover-free compiler gives an exponential identity space without changing partial-key computation, verification, or aggregation. Ciphertexts, digests, partial keys, and aggregate keys contain four, one, two, and two group elements, respectively, and the public parameters contain $O(\lambda^4 B^4 + L)$ group elements. Given the digest, each authority performs a constant number of group operations, while aggregation depends on $L$ and $\theta$ but not on the batch size. The complete construction appears in Section 5, and its security analysis in Section 5.2.

# 3 Preliminaries

**Notations.** We write $x \xleftarrow{\$} X$ uniform sampling from a finite set $X$, and write $y \leftarrow A(x)$ for the output of an algorithm $A$ on input $x$. We use PPT for probabilistic polynomial-time algorithms and $\lambda \in \mathbb{N}$ for the security parameter. For $n \in \mathbb{N}$, we write $[n]$ for $\{1, 2, \ldots, n\}$.

## 3.1 Asymmetric Bilinear Groups

Let $\mathcal{G}$ be a randomized polynomial-time algorithm that, on input $1^\lambda$, outputs an asymmetric bilinear group description $(N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e)$. Here $\mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T$ are groups of order $N$, and $e : \mathbb{G}_N \times \mathbb{H}_N \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map.

**Prime-order groups.** The prime-order construction of Section 6 uses asymmetric bilinear groups of order $N = p$, where $p$ is a prime with $\Theta(\lambda)$ bits.

**Composite-order groups.** The composite-order constructions use asymmetric bilinear groups of order $N = p_1 p_2 p_3$, where $p_1, p_2, p_3$ are distinct primes with $\Theta(\lambda)$ bits. We write the

source groups as products of their prime-order subgroups,

$$\mathbb{G}_N = G_1 G_2 G_3, \qquad \mathbb{H}_N = H_1 H_2 H_3,$$

where $|G_i| = |H_i| = p_i$.

Let $(N = p_1 p_2 p_3, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ be an asymmetric composite-order bilinear group. We write

$$G_i := \mathbb{G}_{p_i}, \qquad H_i := \mathbb{H}_{p_i}, \qquad T_i := e(G_i, H_i),$$

and use $G_{ij} := G_i G_j$ and $H_{ij} := H_i H_j$. Orthogonality gives $e(G_i, H_j) = 1$ for $i \neq j$.

## 3.2 Composite-order Static Assumptions

**Assumption 3.1** ($\mathsf{SD}_G$, [2])**.** Given a group generator $\mathcal{G}$, define the following distribution. Sample

$$(N = p_1 p_2 p_3, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e) \leftarrow_\$ \mathcal{G}(1^\lambda),$$
$$g_1 \leftarrow_\$ G_1, \quad X_3 \leftarrow_\$ G_3, \quad h_1 \leftarrow_\$ H_1, \quad Y_3 \leftarrow_\$ H_3,$$

and set

$$D = (N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e, g_1, X_3, h_1, Y_3).$$

Let

$$T_0 \leftarrow_\$ G_1, \qquad T_1 \leftarrow_\$ G_{12}.$$

For any PPT adversary $\mathcal{B}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{SD}_G}_{\mathcal{B}}(\lambda) := |\Pr[\mathcal{B}(D, T_0) = 1] - \Pr[\mathcal{B}(D, T_1) = 1]| \leq \mathsf{negl}(\lambda)$$

**Assumption 3.2** ($\mathsf{SD}_H$, [2])**.** Given a group generator $\mathcal{G}$, define the following distribution. Sample

$$(N = p_1 p_2 p_3, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e) \leftarrow_\$ \mathcal{G}(1^\lambda),$$
$$g_1 \leftarrow_\$ G_1, \qquad X_{12} \leftarrow_\$ G_{12}, \qquad X_3 \leftarrow_\$ G_3, \qquad Y_{23} \leftarrow_\$ G_{23},$$
$$h_1 \leftarrow_\$ H_1, \qquad h_3 \leftarrow_\$ H_3, \qquad W_{23} \leftarrow_\$ H_{23}.$$

Set

$$D = (N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e, g_1, X_{12}, X_3, Y_{23}, h_1, h_3, W_{23}).$$

Let

$$T_0 \leftarrow_\$ H_{13}, \qquad T_1 \leftarrow_\$ H_{123}.$$

For any PPT adversary $\mathcal{B}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}}(\lambda) := |\Pr[\mathcal{B}(D, T_0) = 1] - \Pr[\mathcal{B}(D, T_1) = 1]| \leq \mathsf{negl}(\lambda)$$

**Assumption 3.3** (PAIR)**.** Given a group generator $\mathcal{G}$, define the following distribution. Sample

$$(N = p_1p_2p_3, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e) \leftarrow\$ \mathcal{G}(1^\lambda),$$

$$\alpha, s \leftarrow\$ \mathbb{Z}_N, \qquad g_1 \leftarrow\$ G_1, \qquad X_2, Y_2, Z_2 \leftarrow\$ G_2, \qquad X_3 \leftarrow\$ G_3,$$

$$h_1 \leftarrow\$ H_1, \qquad h_3 \leftarrow\$ H_3, \qquad X_2' \leftarrow\$ H_2, \qquad W_{23} \leftarrow\$ H_{23}.$$

Set

$$D = \big(N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e, g_1, g_1^\alpha X_2, X_3, g_1^s Y_2, Z_2, h_1, h_3, h_1^\alpha X_2', W_{23}\big).$$

Let

$$T_0 = e(g_1, h_1)^{\alpha s}, \qquad T_1 \leftarrow\$ \mathbb{G}_T.$$

For any PPT adversary $\mathcal{B}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{PAIR}}_{\mathcal{B}}(\lambda) := |\Pr[\mathcal{B}(D, T_0) = 1] - \Pr[\mathcal{B}(D, T_1) = 1]| \le \mathsf{negl}(\lambda)$$

The assumption PAIR is the asymmetric three-prime analogue of Assumption 3 of Lewko-Waters [19].

## 3.3 Prime-order Assumptions

**Definition 3.1** (Matrix distributions [11])**.** Let $\ell > k \ge 1$. A matrix distribution $\mathcal{D}_{\ell,k}$ is a polynomial-time sampleable distribution that outputs matrices in $\mathbb{Z}_p^{\ell\times k}$ of full rank $k$. We write $\mathcal{D}_k := \mathcal{D}_{k+1,k}$, and assume that the first $k$ rows of $\mathbf{M} \leftarrow\$ \mathcal{D}_{\ell,k}$ form an invertible matrix. We write $\mathcal{U}_{\ell,k}$ for the uniform distribution over all such matrices, and $\mathcal{U}_k := \mathcal{U}_{k+1,k}$.

**Assumption 3.4** ($\mathcal{D}_{\ell,k}$-MDDH [11])**.** Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $s \in \{1, 2\}$. The $\mathcal{D}_{\ell,k}$-MDDH assumption holds in $\mathbb{G}$ (for $s = 1$) or $\mathbb{H}$ (for $s = 2$) if for every PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_{\ell,k},s,\mathcal{A}}(\lambda) := \big|\Pr[\mathcal{A}(\mathsf{pg}, [\![\mathbf{M}]\!]_s, [\![\mathbf{Mw}]\!]_s) = 1] - \Pr[\mathcal{A}(\mathsf{pg}, [\![\mathbf{M}]\!]_s, [\![\mathbf{u}]\!]_s) = 1]\big|$$

is negligible, where $\mathbf{M} \leftarrow\$ \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow\$ \mathbb{Z}_p^k$, and $\mathbf{u} \leftarrow\$ \mathbb{Z}_p^\ell$.

The $\mathcal{U}_k$-MDDH assumption is the weakest of these: it is implied by $\mathcal{D}_{\ell,k}$-MDDH for every matrix distribution $\mathcal{D}_{\ell,k}$, and in particular by the $k$-Lin assumption, see also [13, Lemma 2.5 in the full version].

## 3.4 Subset-policy Threshold Batched IBE

We define threshold subset-policy batched IBE following the threshold batched IBE syntax of [1]. A ciphertext is associated with a nonempty *policy* $X \subseteq [n]$ and a batch label $\tau$, and a key is associated with a set $S \subseteq [n]$ and a batch label.

**Definition 3.2** (Threshold Subset-Policy Batched IBE)**.** A threshold subset-policy batched IBE scheme $\Theta$ consists of the following efficient algorithms:

- $\mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta) \to \mathsf{pp}$: on input the security parameter $1^\lambda$, the universe size $n$, the number of authorities $L$, and the threshold $\theta$, the randomized setup algorithm outputs public parameters $\mathsf{pp}$.

- $\mathsf{KeyGen}(\mathsf{pp}) \to (\{\mathsf{pk}_\ell, \mathsf{sk}_\ell\}_{\ell\in[L]}, \mathsf{mpk})$: on input $\mathsf{pp}$, the randomized key-generation algorithm outputs per-authority public keys $\{\mathsf{pk}_\ell\}_{\ell\in[L]}$ and partial master secret keys $\{\mathsf{sk}_\ell\}_{\ell\in[L]}$, together with a master public key $\mathsf{mpk}$. We assume all subsequent algorithms take $\mathsf{pp}$ as input.

- $\mathsf{Encrypt}(\mathsf{mpk}, M, X, \tau) \to \mathsf{ct}$: on input the master public key $\mathsf{mpk}$, a message $M \in \mathcal{M}$, a policy $\emptyset \neq X \subseteq [n]$, and a batch label $\tau \in \mathcal{T}$, the randomized encryption algorithm outputs a ciphertext $\mathsf{ct}$.

- $\mathsf{Digest}(\mathsf{mpk}, S) \to \mathsf{dig}_S$: on input $\mathsf{mpk}$ and a set $S \subseteq [n]$, the deterministic digest algorithm outputs a public digest $\mathsf{dig}_S$.

- $\mathsf{CompKeyShare}(\mathsf{sk}_\ell, \mathsf{dig}_S, \tau) \to \sigma_\ell$: on input the partial master secret key $\mathsf{sk}_\ell$ of authority $\ell \in [L]$, a digest $\mathsf{dig}_S$, and a batch label $\tau$, the randomized share-computation algorithm outputs a key share $\sigma_\ell$.

- $\mathsf{VerifyKeyShare}(\mathsf{pk}, \mathsf{dig}_S, \tau, \sigma) \to 1/0$ : on input a public key $\mathsf{pk}$, a digest $\mathsf{dig}_S$, a batch label $\tau$ and a key share $\sigma$, the deterministic algorithm outputs 1 for a valid key share, otherwise outputs 0.

- $\mathsf{CompKeyAggregate}(\mathsf{mpk}, \{\mathsf{pk}_\ell\}_\ell, \boldsymbol{\sigma}, \mathsf{dig}_S, \tau) \to \bar{\sigma}/\perp$: on input $\mathsf{mpk}$, the per-authority public keys, a vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_L)$ of key shares, a digest $\mathsf{dig}_S$, and a batch label $\tau$, the deterministic aggregation algorithm outputs an aggregate key $\bar{\sigma}$.

- $\mathsf{Decrypt}(\mathsf{mpk}, \bar{\sigma}, \mathsf{dig}_S, S, (X, \tau), \mathsf{ct}) \to M/\perp$: on input $\mathsf{mpk}$, an aggregate key $\bar{\sigma}$, a digest $\mathsf{dig}_S$, the corresponding set $S$, a policy–label pair $(X, \tau)$, and a ciphertext $\mathsf{ct}$, the deterministic decryption algorithm outputs a message $M \in \mathcal{M}$ or $\perp$.

**Completeness.** A threshold subset-policy batched IBE scheme $\Theta$ is said to satisfy *completeness* if for every $\lambda, n, L, \theta \in \mathbb{N}$ , every $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta)$, every $(\{\mathsf{pk}_\ell, \mathsf{sk}_\ell\}_{\ell\in[L]}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, every $M \in \mathcal{M}$, every $\tau \in \mathcal{T}$, and every $\emptyset \neq X \subseteq S \subseteq [n]$, the following holds, where $\mathsf{dig}_S := \mathsf{Digest}(\mathsf{mpk}, S)$:

$$\Pr\left[\mathsf{VerifyKeyShare}(\mathsf{pk}_\ell, \mathsf{dig}_S, \tau, \sigma_\ell) \to 1 : \ \sigma_\ell \leftarrow \mathsf{CompKeyShare}(\mathsf{sk}_\ell, \mathsf{dig}_S, \tau) \ \right] = 1$$

**Correctness.** A threshold subset-policy batched IBE scheme $\Theta$ is said to satisfy *correctness* if for every $\lambda, n, L, \theta \in \mathbb{N}$ , every $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta)$, every $(\{\mathsf{pk}_\ell, \mathsf{sk}_\ell\}_{\ell\in[L]}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, every $M \in \mathcal{M}$, every $\tau \in \mathcal{T}$, and every $\emptyset \neq X \subseteq S \subseteq [n]$, the following holds, where $\mathsf{dig}_S := \mathsf{Digest}(\mathsf{mpk}, S)$:

For every authorized set $V \subseteq [L]$ with $|V| \geq \theta$, we have $\Pr[M' = M] = 1$ in the experiment

$$\left[\begin{array}{l} \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, M, X, \tau) \\ \sigma_\ell \leftarrow \mathsf{CompKeyShare}(\mathsf{sk}_\ell, \mathsf{dig}_S, \tau) \text{ for } \ell \in U \\ \sigma_\ell := \perp \text{ for } \ell \notin U \\ \bar{\sigma} \leftarrow \mathsf{CompKeyAggregate}(\mathsf{mpk}, \{\mathsf{pk}_\ell\}_\ell, \boldsymbol{\sigma}, \mathsf{dig}_S, \tau) \\ M' \leftarrow \mathsf{Decrypt}(\mathsf{mpk}, \bar{\sigma}, \mathsf{dig}_S, S, (X, \tau), \mathsf{ct}) \end{array}\right],$$

where $\boldsymbol{\sigma} := (\sigma_1, \ldots, \sigma_L)$ and $\bar{\sigma}$ is the aggregate key.

**Non-triviality.** A threshold subset-policy batched IBE scheme $\Theta$ is *non-trivial* if the size of the aggregate or batch decryption key $\bar{\sigma}$ doesn't grow linearly in $|S|$.

**Security.** We consider two corruption models, $\chi \in \{\mathsf{ad}, \mathsf{st}\}$. In both, the adversary chooses the challenge policy, the challenge batch label, the challenge messages, and its key-share queries adaptively. For $\chi = \mathsf{ad}$, it may also corrupt authorities adaptively, and for $\chi = \mathsf{st}$, it fixes the corrupted set before setup, as in the static-corruption model of [1].

**Definition 3.3** (Adaptive Security)**.** We define the adaptive security experiment $\mathsf{Exp}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta, b)$ as follows:

1. Challenger computes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta)$.
2. If $\chi = \mathsf{st}$, then $\mathcal{A}(\mathsf{pp})$ outputs a set $S_{\mathsf{corr}} \subseteq [L]$ with $|corr| \leq \theta - 1$. If $\chi = \mathsf{ad}$, the challenger sets $S_{\mathsf{corr}} := \emptyset$.
3. The challenger computes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta)$ and $(\{\mathsf{pk}_\ell, \mathsf{sk}_\ell\}_{\ell \in [L]}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and gives $\mathsf{pp}$, $\mathsf{mpk}$, $\{\mathsf{pk}_\ell\}_{\ell \in [L]}$, and $\{\mathsf{sk}_\ell\}_{\ell \in S_{\mathsf{corr}}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ makes polynomially many queries of the following two types, adaptively.
   - If $\chi = \mathsf{ad}$, $\mathcal{A}$ can make corruption queries. On input $\ell \in [L]$ with $|S_{\mathsf{corr}} \cup \{\ell\}| \leq \theta - 1$, the challenger returns $\mathsf{sk}_\ell$ and sets $S_{\mathsf{corr}} := S_{\mathsf{corr}} \cup \{\ell\}$.
   - *Key queries.* On input $(S, \tau)$, with $S \subseteq [n]$ and $\tau \in \mathcal{T}$, the challenger computes $\mathsf{dig}_S \leftarrow \mathsf{Digest}(\mathsf{mpk}, S)$ and returns
   $$\left(\sigma_\ell \leftarrow \mathsf{CompKeyShare}(\mathsf{sk}_\ell, \mathsf{dig}_S, \tau)\right)_{\ell \in [L] \setminus S_{\mathsf{corr}}},$$
5. $\mathcal{A}$ outputs two messages $M_0, M_1 \in \mathcal{M}$, a policy $\emptyset \neq X^* \subseteq [n]$, and a batch label $\tau^* \in \mathcal{T}$. The challenger computes $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, M_b, X^*, \tau^*)$ and gives $\mathsf{ct}^*$ to $\mathcal{A}$.
6. $\mathcal{A}$ can make (admissible) post challenge queries which are answered as in Step 3.
7. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. The experiment outputs $b = b'$.

**Admissibility.** $\mathcal{A}$ is admissible if the following conditions are satisfied across all pre-challenge and post-challenge queries:

(a) $\mathcal{A}$ makes at most one key query with label equal to the challenge label $\tau^*$;

(b) If $\mathcal{A}$ makes a key query $(S, \tau^*)$, then it must be the case that $X^* \not\subseteq S$.

Key queries with batch labels $\tau \neq \tau^*$ are unrestricted. In particular, a batch label may be queried several times and $S$ may contain $X^*$. The advantage of $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta) := \Big| \Pr\big[\mathsf{Exp}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta, 0) = 1\big] - \Pr\big[\mathsf{Exp}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta, 1) = 1\big]\Big|.$$

In both static as well as adaptive corruption models, the challenge and the key-share queries are adaptive. We say that the threshold batched IBE scheme $\Theta$ is *fully adaptively secure* if $\mathsf{Adv}^{\mathsf{ThSUB},\mathsf{ad}}_{\Theta,\mathcal{A}}$ is negligible for every admissible PPT adversary $\mathcal{A}$, and that $\Theta$ has *adaptive security under static corruption* if the same holds for $\mathsf{Adv}^{\mathsf{ThSUB},\mathsf{st}}_{\Theta,\mathcal{A}}$.

As in [1], a key query returns the shares of honest authorities for one common pair $(S, \tau)$. In particular, a single query cannot ask different honest authorities for different sets under the same batch label.

*Remark* 3.1 (Subset-policy batched IBE). For the single authority case of $L = \theta = 1$, the two corruption models coincide and we can omit $\chi$. There is a single master secret key $\mathsf{msk}$ and $(S, \tau)$ specific keys are computed as in $\mathsf{CompKeyShare}$. Setting $\mathsf{ComputeKey} := \mathsf{CompKeyShare}$ we obtain the synatx for a subset-policy batched IBE as follows:

$$\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Digest}, \mathsf{ComputeKey}, \mathsf{Decrypt})$$

of (non-threshold) *subset-policy batched IBE*. Here, setup takes input $(1^\lambda, 1^n)$, and we write key computation as

$$\mathsf{ComputeKey}(\mathsf{msk}, \mathsf{dig}_S, \tau) \to \mathsf{sk}.$$

An adversary is admissible if it makes at most one key query with batch label $\tau^*$, and this query $(S, \tau^*)$ satisfies $X^* \not\subseteq S$. We write $\mathsf{Exp}^{\mathsf{SUB}}_{\Sigma,\mathcal{A}}(\lambda, n, b)$ and $\mathsf{Adv}^{\mathsf{SUB}}_{\Sigma,\mathcal{A}}(\lambda, n)$ for the resulting experiment and advantage.

*Remark* 3.2 (Threshold batched IBE). Restricting every policy of a threshold subset-policy scheme over $[n]$ to a singleton $X = \{\mathsf{id}\}$ yields a threshold batched IBE scheme with $\mathcal{I} = [n]$ and $B = n$. A key for $(S, \tau)$, decrypts every ciphertext for $(\mathsf{id}, \tau)$ with $\mathsf{id} \in S$. An adversary is admissible if every key-share query $(S, \tau)$ satisfies $|S| \leq B$ and it makes at for most one key query with batch label $\tau^*$, and this query $(S, \tau^*)$ satisfies $\mathsf{id}^* \notin S$. We write $\mathsf{Exp}^{\mathsf{ThBIBE},\chi}_{\Pi,\mathcal{A}}(\lambda, B, L, \theta, b)$ and $\mathsf{Adv}^{\mathsf{ThBIBE},\chi}_{\Pi,\mathcal{A}}(\lambda, B, L, \theta)$ for the resulting experiment and advantage, and $\mathsf{Exp}^{\mathsf{BIBE}}_{\Pi,\mathcal{A}}(\lambda, B, b)$ and $\mathsf{Adv}^{\mathsf{BIBE}}_{\Pi,\mathcal{A}}(\lambda, B)$ for $L = \theta = 1$.

# 4 A Generic Cover-Free Compiler

We give a generic compiler that turns a threshold subset-policy batched IBE scheme over a polynomial-size universe into a threshold batched IBE scheme with an exponentially large identity space and a polynomial batch bound $B$. Each identity is represented by a block of a $B$-cover-free family over $[n]$, and each batch by the union of the blocks of its identities. Cover-freeness guarantees that the subset-policy decryption condition holds exactly for the identities in the batch. The compiler uses only the syntax and security of the underlying scheme. It therefore applies to the composite-order construction of Section 5, and the non-threshold prime-order construction of Section 6.

The cover-free transformation for subset policies already appears in [14, Remark C.15 in the full version]. The identity encoding as sets, representation of batches as union of sets, and subset decryption predicate are the same in both frameworks. The classical cover-free family used below also has the same bound $n = O(\lambda^2 B^2)$ on the underlying universe. The differences lie in the underlying subset-policy constructions and our threshold formulation. The Appendix C instantiation of Gong et al. allows adaptive challenge identities but requires the challenge label to be chosen selectively, and its security relies on a $q$-type assumption. Our constructions support adaptive choices of both the identity and the label under static assumptions. We state the transformation for the threshold syntax of Definition 3.2, with partial-key computation and aggregation inherited unchanged from the underlying scheme. Consequently, the compiler preserves its threshold, corruption bound, and security. In particular, if the underlying scheme is fully adaptively secure, then so is the compiled scheme, and the same holds for adaptive challenge security under static corruption.

## 4.1 Cover-Free Families and the Compiler

**Definition 4.1** (Cover-free family [10])**.** Let $B, n \geq 1$ and let $\mathcal{I}$ be an index set. A family of sets over $[n]$, denoted $\mathcal{C} = \{C_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}$, is said to be $B$-*cover-free* if for every $\mathsf{id} \in \mathcal{I}$ and every $\Gamma \subseteq \mathcal{I} \setminus \{\mathsf{id}\}$ with $|\Gamma| \leq B$, it holds that:

$$C_{\mathsf{id}} \not\subseteq \bigcup_{\mathsf{id}'\in\Gamma} C_{\mathsf{id}'}.$$

Notice that taking $\Gamma = \emptyset$ implies each block $C_{\mathsf{id}}$ is non-empty.

For any subset of identities $\Gamma \subseteq \mathcal{I}$, we define the union of their corresponding blocks as:

$$S_\Gamma := \bigcup_{\mathsf{id}\in\Gamma} C_{\mathsf{id}} \subseteq [n]. \tag{1}$$

In particular, $S_\emptyset = \emptyset$. The cover-free property naturally implies that for every $\mathsf{id} \in \mathcal{I}$ and every batch $\Gamma \subseteq \mathcal{I}$ of size $|\Gamma| \leq B$:

$$\mathsf{id} \in \Gamma \quad \Longleftrightarrow \quad C_{\mathsf{id}} \subseteq S_\Gamma. \tag{2}$$

This equivalence is the main idea underlying our compiler. We additionally require the family $\mathcal{C}$ to be *efficiently computable*, i.e., it must have a succinct, polynomial-size public description $d_{\mathcal{C}}$ from which $C_{\mathsf{id}}$ can be computed in polynomial time for any $\mathsf{id} \in \mathcal{I}$.

**Construction 4.1** (Cover-Free Compiler)**.** Let $\Theta$ be a threshold subset-policy batched IBE scheme satisfying Definition 3.2, whose algorithms we mark with the subscript $\subseteq$. Let $\mathcal{C} := \{C_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}$ be an efficiently computable $B$-cover-free family over $[n]$ with public description $d_{\mathcal{C}}$. Both $n$ and $d_{\mathcal{C}}$ are generated deterministically from the security parameter $\lambda$ and batch bound $B$.

The compiled scheme $\Pi$, whose algorithms we denote with superscript $\mathsf{LU}$, is a threshold batched IBE scheme with a large identity space $\mathcal{I}$, batch bound $B$, and the same message ($\mathcal{M}$) and label ($\mathcal{T}$) spaces as $\Theta$. A batch is represented as a list of at most $B$ distinct identities.

$\mathsf{Setup}^{\mathsf{LU}}(1^\lambda, 1^B, 1^L, 1^\theta)$**:** Compute $n$ and $d_{\mathcal{C}}$ deterministically from $(1^\lambda, 1^B)$. Run $\mathsf{pp}_\subseteq \leftarrow \mathsf{Setup}_\subseteq(1^\lambda, 1^n, 1^L, 1^\theta)$ and output $\mathsf{pp}^{\mathsf{LU}} := (\mathsf{pp}_\subseteq, B, n, d_{\mathcal{C}})$.

$\mathsf{KeyGen}^{\mathsf{LU}}(\mathsf{pp}^{\mathsf{LU}})$**:** Run $\left(\{\mathsf{sk}_\ell\}_{\ell\in[L]}, \{\mathsf{pk}_\ell\}_{\ell\in[L]}, \mathsf{mpk}_\subseteq\right) \leftarrow \mathsf{KeyGen}_\subseteq(\mathsf{pp}_\subseteq)$ and output the identically generated secret key shares and public keys, alongside the updated master public key $\mathsf{mpk}^{\mathsf{LU}} := (\mathsf{mpk}_\subseteq, B, n, d_{\mathcal{C}})$.

$\mathsf{Encrypt}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, M, \mathsf{id}, \tau)$**:** Compute $C_{\mathsf{id}}$ using $d_{\mathcal{C}}$ and run:

$$\mathsf{ct}_\subseteq \leftarrow \mathsf{Encrypt}_\subseteq(\mathsf{mpk}_\subseteq, M, C_{\mathsf{id}}, \tau).$$

Parse $\mathsf{ct}_\subseteq = (C_{\mathsf{id}}, \tau, \mathsf{ct})$, and output the compiled ciphertext:

$$\mathsf{ct}^{\mathsf{LU}} := (\mathsf{id}, \tau, \mathsf{ct}).$$

$C_{\mathsf{id}}$ is replaced by the compact identity $\mathsf{id}$, from which the set is recomputed during decryption.

$\mathsf{Digest}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, \Gamma)$**:** Compute $S_\Gamma$ from (1) and output $D_\Gamma := \mathsf{Digest}_\subseteq(\mathsf{mpk}_\subseteq, S_\Gamma)$.

$\mathsf{CompKeyShare}^{\mathsf{LU}}(\mathsf{sk}_\ell, D, \tau)$**:** Return $\sigma_\ell := \mathsf{CompKeyShare}_\subseteq(\mathsf{sk}_\ell, D, \tau)$.

$\mathsf{CompKeyAggregate}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, \{\mathsf{pk}_\ell\}_{\ell\in[L]}, \boldsymbol{\sigma}, D, \tau)$**:** Let $S \subseteq [L]$ with $|S| \geq \theta$ be a set of parties, and let $\boldsymbol{\sigma} := \{\sigma_\ell\}_{\ell\in S}$ be their corresponding partial keys. Return

$$\bar{\sigma} := \mathsf{CompKeyAggregate}_\subseteq(\mathsf{mpk}_\subseteq, \{\mathsf{pk}_\ell\}_{\ell\in[L]}, \boldsymbol{\sigma}, D, \tau)$$

$\mathsf{Decrypt}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, \bar{\sigma}, D, \Gamma, (\mathsf{id}, \tau), \mathsf{ct}^{\mathsf{LU}})$: Parse $\mathsf{ct}^{\mathsf{LU}} = (\mathsf{id}_{\mathsf{ct}}, \tau_{\mathsf{ct}}, \mathsf{ct})$. If $\mathsf{id}_{\mathsf{ct}} \neq \mathsf{id}$, $\tau_{\mathsf{ct}} \neq \tau$, $\mathsf{id} \notin \Gamma$, or $|\Gamma| > B$, return $\perp$. Otherwise, compute the sets $C_{\mathsf{id}}$ and $S_\Gamma$, reconstruct the underlying ciphertext $\mathsf{ct}_\subseteq := (C_{\mathsf{id}}, \tau, \mathsf{ct})$, and return:

$$\mathsf{Decrypt}_\subseteq(\mathsf{mpk}_\subseteq, \bar{\sigma}, D, S_\Gamma, (C_{\mathsf{id}}, \tau), \mathsf{ct}_\subseteq).$$

*Remark.* By construction, the secret key shares, verification keys, digests, partial keys, and aggregated batch keys of $\Pi$ are identical to those of the underlying scheme $\Theta$. The compiler mainly modifies the ciphertext, replacing identities ($\mathsf{id}$) with cover-free sets ($C_{\mathsf{id}}$).

## 4.2 Correctness and Security

**Theorem 4.1.** *Let $\Pi$ be Construction 4.1 instantiated with a threshold subset-policy batched IBE scheme $\Theta$ and an efficiently computable $B$-cover-free family $\mathcal{C}$ over $[n]$.*

(a) *If $\Theta$ is correct, then so is $\Pi$.*

(b) *For either corruption model $\chi \in \{\mathsf{ad}, \mathsf{st}\}$ and every admissible PPT adversary $\mathcal{A}$ against $\Pi$ in that model that makes at most $Q$ key queries, there exists an admissible PPT adversary $\mathcal{B}$ against $\Theta$ in the same model that also makes at most $Q$ key queries and satisfies*

$$\mathsf{Adv}^{\mathsf{ThBIBE},\chi}_{\Pi,\mathcal{A}}(\lambda, B, L, \theta) = \mathsf{Adv}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{B}}(\lambda, n, L, \theta).$$

*Thus, the compiler preserves both fully adaptive security and adaptive challenge security under static corruption, with no additional security loss.*

*Proof. Part (a) (Correctness).* Let $\Gamma \subseteq \mathcal{I}$ with $|\Gamma| \leq B$, $\tau \in \mathcal{T}$, and $M \in \mathcal{M}$. Let

$$D_\Gamma := \mathsf{Digest}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, \Gamma) = \mathsf{Digest}_\subseteq(\mathsf{mpk}_\subseteq, S_\Gamma).$$

Since $\mathsf{CompKeyShare}^{\mathsf{LU}}$ and $\mathsf{CompKeyAggregate}^{\mathsf{LU}}$ simply run the corresponding algorithms of $\Theta$ using $D_\Gamma$ and $\tau$, the completeness of $\Pi$ for the pair $(\Gamma, \tau)$ follows directly from the completeness of $\Theta$ for $(S_\Gamma, \tau)$. For correctness, consider an identity $\mathsf{id} \in \Gamma$. Let $\mathsf{ct}^{\mathsf{LU}} \leftarrow \mathsf{Encrypt}^{\mathsf{LU}}(\mathsf{mpk}^{\mathsf{LU}}, M, \mathsf{id}, \tau)$, and let $\bar{\sigma} \neq \perp$ be the output of $\mathsf{CompKeyAggregate}^{\mathsf{LU}}$ on any valid set of key shares. By the definition of $S_\Gamma$, we have $\emptyset \neq C_{\mathsf{id}} \subseteq S_\Gamma$. During decryption, $\mathsf{Decrypt}^{\mathsf{LU}}$ runs $\mathsf{Decrypt}_\subseteq$ using the digest of $S_\Gamma$ and the reconstructed $\Theta$-ciphertext $(C_{\mathsf{id}}, \tau, \mathsf{ct})$. Because this is an honest encryption of $M$ under $(C_{\mathsf{id}}, \tau)$ and $C_{\mathsf{id}} \subseteq S_\Gamma$, the correctness of $\Theta$ guarantees that the output must be $M$. Note that this argument only uses the non-emptiness of the cover-free blocks.

*Part (b) (Security).* Fix a corruption model $\chi \in \{\mathsf{ad}, \mathsf{st}\}$ and a PPT adversary $\mathcal{A}$ against $\Pi$ in that model, which is admissible in the sense of Definition 3.3. That is, makes at most one query with the challenge batch label. Using $\mathcal{A}$, we construct an adversary $\mathcal{B}$ against $\Theta$ in the same model, which runs $\mathcal{A}$ as a challenger of $\Pi$ and simulates its experiment as follows:

- *Initialization.* Compute $n$ and $d_{\mathcal{C}}$ from $(1^{\lambda}, 1^{B})$. If $\chi = \mathsf{st}$, $\mathcal{A}$ outputs its corruption set $S_{\mathsf{corr}}$ before $\mathsf{Setup}$. $\mathcal{B}$ forwards the same $S_{\mathsf{corr}}$ to the its challenger before setup. If $\chi = \mathsf{ad}$, initialize $S_{\mathsf{corr}} := \emptyset$.
- *Setup.* On receiving $\mathsf{pp}_{\subseteq}$, $\mathsf{mpk}_{\subseteq}$, and $\{\mathsf{pk}_{\ell}\}_{\ell \in [L]}$, give $\mathcal{A}$ the compiled parameters $\mathsf{pp}^{\mathsf{LU}} = (\mathsf{pp}_{\subseteq}, B, n, d_{\mathcal{C}})$, $\mathsf{mpk}^{\mathsf{LU}} = (\mathsf{mpk}_{\subseteq}, B, n, d_{\mathcal{C}})$, and the same partial public keys. If $\chi = \mathsf{st}$, also forward the shares $\{\mathsf{sk}_c\}_{c \in S_{\mathsf{corr}}}$ supplied by the challenger and resume $\mathcal{A}$ from its retained state.
- *Corruption queries.* If $\chi = \mathsf{ad}$, forward each corruption query $\ell$ to the underlying challenger, return its response $\mathsf{sk}_{\ell}$ to $\mathcal{A}$, and update $S_{\mathsf{corr}} := S_{\mathsf{corr}} \cup \{\ell\}$.
- *Key queries.* When $\mathcal{A}$ makes a query $(\Gamma, \tau)$, compute $S_{\Gamma}$ and forward the corresponding query $(S_{\Gamma}, \tau)$ to the challenger for $\Theta$ , and send back the response to $\mathcal{A}$.
- *Challenge.* On receiving $\mathcal{A}$'s challenge $(M_0, M_1, \mathsf{id}^*, \tau*)$, submit its messages $M_0, M_1$ and label $\tau^*$ to the challenger of $\Theta$ with policy $C_{\mathsf{id}^*}$. On receiving its challenge ciphertext $\mathsf{ct}^*$, forward $\mathsf{ct}^*$ as the challenge ciphertext to $\mathcal{A}$.
- *Guess.* Output whatever $\mathcal{A}$ outputs.

We argue that $\mathcal{B}$ is *admissible* as long as $\mathcal{A}$ is admissible. $\mathcal{B}$'s initial corruption set in the static model and its corruption queries in the adaptive model are those of $\mathcal{A}$, so the corruption bound and admissibility are preserved. Its key queries have the same labels as those of $\mathcal{A}$. The inputs to the challenger are well formed since $S_{\Gamma} \subseteq [n]$ and $C_{\mathsf{id}^*}$ is a nonempty subset of $[n]$. If $\mathcal{B}$ makes a key query for $(S_{\Gamma}, \tau^*)$, then $\mathcal{A}$ made the corresponding query $(\Gamma, \tau^*)$. Admissibility of $\mathcal{A}$ gives $\mathsf{id}^* \notin \Gamma$ and $|\Gamma| \leq B$ and by (2), this implies $C_{\mathsf{id}^*} \not\subseteq S_{\Gamma}$. Therefore, $\mathcal{B}$ is admissible and makes at most $Q$ key queries.

We now argue that $\mathcal{B}$ perfectly simulates the experiment for $\mathcal{A}$. The reduction maintains the order of all the queries and responses so both experiments have the exact same corrupted set $S_{\mathsf{corr}}$ at the time of each key query. Since $\Pi$ runs the algorithms of $\Theta$ unchanged, the challenge ciphertext returned to $\mathcal{A}$ is distributed exactly as an honest encryption of $M_b$ under $\Pi$ for $(\mathsf{id}^*, \tau^*)$. Therefore, the simulation preserves the joint distribution of the entire view of $\mathcal{A}$, and the winning probability of $\mathcal{A}$ gives the winning probability of $\mathcal{B}$. $\mathcal{B}$ forwards the bit output by $\mathcal{A}$ unchanged. Therefore, for each $b \in \{0, 1\}$, we obtain

$$\begin{aligned}&\Pr\left[\mathsf{Exp}^{\mathsf{ThBIBE},\chi}_{\Pi,\mathcal{A}}(\lambda, B, L, \theta, b) = 1\right] \\ &\quad = \Pr\left[\mathsf{Exp}^{\mathsf{ThSUB},\chi}_{\Theta,\mathcal{B}}(\lambda, n, L, \theta, b) = 1\right].\end{aligned}$$

The equality of the advantages follows from taking the difference between the cases $b = 0$ and $b = 1$. □

*Remark* 4.1 (Non-threshold Compiler). For the non-threshold case where $L = \theta = 1$ the compiler is defined in the same way. Let $\Sigma$ be a subset-policy batched IBE scheme as per Remark 3.1 in Definition 3.2. In that case the compiler takes a standard subset-policy

batched IBE scheme $\Sigma$ and produces a standard batched IBE scheme $\Pi$ (Remark 3.2). The proof of Theorem 4.1 applies verbatim. Thus, $\Pi$ is perfectly correct if $\Sigma$ is, and

$$\mathsf{Adv}^{\mathsf{BIBE}}_{\Pi,\mathcal{A}}(\lambda, B) = \mathsf{Adv}^{\mathsf{SUB}}_{\Sigma,\mathcal{B}}(\lambda, n).$$

## 4.3 A Cover-free Family for an Exponential Identity Space

We instantiate the compiler with the cover-free family obtained from Reed–Solomon codes [15, 10]. The core intuition is to represent each identity as a unique low-degree polynomial over a finite field $\mathbb{F}_q$. The block $C_{\mathsf{id}}$ corresponding to an identity $\mathsf{id}$ is simply the evaluation graph of that polynomial over all points in the field.

For a prime power $q$ and an integer $\lambda \geq 1$, let $\mathbb{F}_q[Z]_{<\lambda}$ denote the set of polynomials over $\mathbb{F}_q$ of degree strictly less than $\lambda$. Because any two distinct polynomials of degree less than $\lambda$ can agree on at most $\lambda - 1$ points, their evaluation graphs intersect at most $\lambda - 1$ points. As a result, any $B$ polynomials distinct from $f$ cover at most $B(\lambda - 1)$ of the $q$ points of the graph of $f$, and choosing $q > B(\lambda - 1)$ leaves at least one point uncovered. By choosing the field size $q$ to be sufficiently larger than this intersection bound, we can guarantee cover-freeness.

**Lemma 4.1.** *Let $\lambda \geq 2$ and $B \geq 1$. Let $q$ be the smallest power of 2 such that $q > (\lambda + 1)B$, and let $n = q^2$. Then there exists a $B$-cover-free family $\{C_f\}_{f \in \mathbb{F}_q[Z]_{<\lambda}}$ with $q^\lambda$ blocks, each of size $q$. Each block can be computed with $O(q\lambda)$ operations in $\mathbb{F}_q$, and our choice of $q$ guarantees:*

$$(\lambda + 1)B < q \leq 2(\lambda + 1)B \quad \textit{and} \quad n \leq 4(\lambda + 1)^2 B^2.$$

*Proof.* Fix a bijection $\iota : \mathbb{F}_q^2 \to [n]$. For each polynomial $f \in \mathbb{F}_q[Z]_{<\lambda}$, we define its corresponding block as the set of mapped points in its graph:

$$C_f := \{\iota(a, f(a)) : a \in \mathbb{F}_q\} \subseteq [n].$$

Since the polynomial is evaluated at every element in $\mathbb{F}_q$, the size of every block is exactly $|C_f| = q$.

To prove cover-freeness, let $f \notin \Gamma$ be a target polynomial and let $\Gamma$ be a batch of polynomials with $|\Gamma| \leq B$. For every $f' \in \Gamma$, the difference polynomial $f - f'$ is non-zero and has degree at most $\lambda - 1$. A non-zero polynomial of degree at most $\lambda - 1$ has at most $\lambda - 1$ roots in $\mathbb{F}_q$, which means $f$ and $f'$ agree on at most $\lambda - 1$ evaluations. Thus, $|C_f \cap C_{f'}| \leq \lambda - 1$.

Taking the union over all $f' \in \Gamma$, the maximum number of points $C_f$ can share with the entire batch is:

$$\left| C_f \cap \bigcup_{f' \in \Gamma} C_{f'} \right| \leq \sum_{f' \in \Gamma} |C_f \cap C_{f'}| \leq B(\lambda - 1).$$

Because we chose $q > B(\lambda - 1)$, it strictly follows that $B(\lambda - 1) < |C_f|$. Therefore, $C_f$ cannot be fully covered by the union, yielding $C_f \not\subseteq \bigcup_{f' \in \Gamma} C_{f'}$. The bounds on $q$ and $n$ follow immediately from the properties of powers of 2. □

**Identities as bit strings.** In practice, identities are bit strings. To map these to polynomials, write $q = 2^s$ and fix a bijection $\beta : \{0,1\}^s \to \mathbb{F}_q$. For $\mathsf{id} \in \{0,1\}^\lambda$, pad $\mathsf{id}$ with zeros to length $s\lambda$, split it into chunks $u_0, \ldots, u_{\lambda-1} \in \{0,1\}^s$, and let $f_{\mathsf{id}} := \sum_{i=0}^{\lambda-1} \beta(u_i) Z^i \in \mathbb{F}_q[Z]_{<\lambda}$. The map $\mathsf{id} \mapsto f_{\mathsf{id}}$ is injective, so $\{C_{f_{\mathsf{id}}}\}_{\mathsf{id}\in\{0,1\}^\lambda}$ is a subfamily of the family in Lemma 4.1 and is therefore $B$-cover-free.

**Efficiency.** By Theorem 4.1, the compiler does not change the size of ciphertexts, digests, partial keys, and batch keys. A compiled ciphertext contains the identity $\mathsf{id}$ instead of the block $C_{\mathsf{id}}$, which has $q$ elements. Encryption and decryption compute $C_{\mathsf{id}} = \{\iota(a, f(a)) : a \in \mathbb{F}_q\}$ from $\mathsf{id}$, where $f := f_{\mathsf{id}}$. This involves evaluating $f$ at all $q$ points of $\mathbb{F}_q$ and takes $O(q\lambda)$ operations in $\mathbb{F}_q$. With the family of Lemma 4.1, the underlying scheme uses the universe $[n]$ with $n = q^2 = O(\lambda^2 B^2)$, so all parameters are polynomial in $\lambda$ and $B$. The concrete parameters of the two instantiations are given in Sections 5 and 6.

**Offline preprocessing for encryption.** The compiler introduces some public computation that scales with the batch bound $B$. Specifically, encryption computes the block $C_{\mathsf{id}}$ and evaluates the underlying encryption algorithm on a policy of size $|C_{\mathsf{id}}|$. Because this work depends strictly on the identity, it can be decoupled and performed offline.

- *Identity-dependent work.* In both of our schemes, $\mathsf{Encrypt}_\subseteq$ uses the policy $X$ only in one public value $L_X$, namely $L_X := g_1^u \prod_{i\in X} g_1^{w_i}$ in Section 5 and $L_X := [\![\boldsymbol{A}^\top(\boldsymbol{U} + \boldsymbol{W}_X)]\!]_1$ in Section 6, from which the ciphertext component $C_1$ is computed. Hence, once an identity $\mathsf{id}$ is fixed, anyone can compute $C_{\mathsf{id}}$ and $L_{C_{\mathsf{id}}}$ before the message and the tag are known. For the cover-free family from Section 4.3, this takes $O(q\lambda)$ operations in $\mathbb{F}_q$ and $O(q)$ group operations, and the result can be reused for every ciphertext under $\mathsf{id}$. Given $L_{C_{\mathsf{id}}}$, online encryption computes the ciphertext components from $L_{C_{\mathsf{id}}}$ and the public key, with only a constant number of group operations.

In the context of encrypted mempools, an identity corresponds to a fresh transaction identifier, and a tag represents a block height. A wallet can therefore prepare transaction identifiers, their cover-free blocks $C_{\mathsf{id}}$, and the products $L_{C_{\mathsf{id}}}$ in the background, completing the fast online encryption only at the moment a transaction is submitted.

# 5 Threshold Batched IBE in Composite-Order Groups

In this section we construct a threshold subset policy batched IBE scheme over the universe [n] in asymmetric bilinear groups of composite order $N = p_1p_2p_3$. A ciphertext consists of four group elements. Each partial key and the batch key consists of two group elements each. All sizes are independent of $n$, $L$, and $\theta$. On applying the compiler from Section 4 yields a threshold batched IBE scheme for identities in $\{0,1\}^\lambda$ with same aggregate key and ciphertext sizes (Section 5.2). The non-threshold scheme follows as the case $L = \theta = 1$ Remark 3.1.

**Secret sharing over $\mathbb{Z}_N$.** The master secrets $\alpha, u \in \mathbb{Z}_N$ are shared with Shamir's scheme over the ring $\mathbb{Z}_N$, with evaluation point $\xi_\ell := \ell$ for authority $\ell \in [L]$. We require $L < \min\{p_1, p_2, p_3\}$, which setup can check since it knows the primes. Then every $\xi_\ell$ and every difference $\xi_\ell - \xi_k$ with $\ell \neq k$ is invertible modulo $N$, so for every $V \subseteq [L]$ with $|V| \geq \theta$ the Lagrange coefficients

$$\omega_\ell^V := \prod_{k \in V \setminus \{\ell\}} \frac{-\xi_k}{\xi_\ell - \xi_k} \bmod N \qquad (\ell \in V)$$

are well defined and satisfy $\sum_{\ell \in V} \omega_\ell^V a(\xi_\ell) = a(0)$ for every $a \in \mathbb{Z}_N[Z]$ of degree at most $\theta - 1$, by interpolation modulo each $p_i$ and the Chinese Remainder Theorem. We instantiate $\mathcal{G}$ with $\Theta(\lambda)$-bit primes, so every polynomial number of authorities is supported for all sufficiently large $\lambda$.

## 5.1 Construction

We present the threshold batched IBE construction below, following the notation in Section 3.1.

$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^L, 1^\theta) \to \mathsf{pp}}$

- Run $\mathsf{pg} := (N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, where $N = p_1 p_2 p_3$. Output $\perp$ if $L \geq min(p_1, p_2, p_3)$.
- Sample $g_1 \leftarrow\$ G_1$, $h_1 \leftarrow\$ H_1$, $h_3 \leftarrow\$ H_3$, and output $\mathsf{pp} := (\mathsf{pg}, n, L, \theta, g_1, h_1, h_3)$.

$\underline{\mathsf{KeyGen}(\mathsf{pp}) \to (\{\mathsf{sk}_\ell\}_{\ell \in [L]}, \{\mathsf{pk}_\ell\}_{\ell \in [L]}, \mathsf{mpk})}$

- Sample $\alpha, u, \beta, \gamma \leftarrow\$ \mathbb{Z}_N$ and $w_i, r_i \leftarrow\$ \mathbb{Z}_N$ for $i \in [n]$.
- For $j \in [n]$, set $P_j := h_3^{z_j} h_1^{r_j}$. For $i \neq j$, set $P_{i,j} := h_3^{z_{i,j}} h_1^{w_i r_j}$. Here all $z_j, z_{i,j} \leftarrow\$ \mathbb{Z}_N$. The diagonal elements $P_{i,i}$ are not computed.
- Sample uniform random polynomials $q_\alpha, q_u \in \mathbb{Z}_N[Z]$ of degree at most $\theta - 1$ with $q_\alpha(0) = \alpha$ and $q_u(0) = u$.
- For $\ell \in [L]$, set $\alpha_\ell := q_\alpha(\xi_\ell)$, $u_\ell := q_u(\xi_\ell)$, and

$$\mathsf{pk}_\ell := (e(g_1, h_1)^{\alpha_\ell}, g_1^{u_\ell}) = (A_\ell, U_\ell), \qquad \mathsf{sk}_\ell := \left(h_1^{\alpha_\ell}, u_\ell\right).$$

- Set $\Omega := e(g_1, h_1)^\alpha$ and output $\{\mathsf{sk}_\ell\}_{\ell \in [L]}$, $\{\mathsf{pk}_\ell\}_{\ell \in [L]}$, and

$$\mathsf{mpk} := \left(\mathsf{pp}, g_1^u, g_1^\beta, g_1^\gamma, h_1^\beta, h_1^\gamma, \Omega, \{g_1^{w_i}\}_{i \in [n]}, \{P_j\}_{j \in [n]}, \{P_{i,j}\}_{i \neq j}, \{\mathsf{pk}_\ell\}_{\ell \in [L]}\right).$$

$\underline{\mathsf{Encrypt}(\mathsf{mpk}, M, X, \tau) \to \mathsf{ct}}$

- Sample $s \leftarrow\$ \mathbb{Z}_N$ and output $\mathsf{ct} := (C_0, C_1, C_2, C_3)$, where

$$C_0 := g_1^s, \quad C_1 := \left(g_1^u \prod_{i \in X} g_1^{w_i}\right)^s = g_1^{us \sum_{i \in X} w_i}, \quad C_2 := g_1^{s(\beta + \gamma\tau)}, \quad C_3 := \Omega^s \cdot M.$$

$\underline{\mathsf{Digest}(\mathsf{mpk}, S) \to D_S}$

- Output $\perp$ unless $S \subseteq [n]$. Otherwise output $D_S := \prod_{j \in [n] \setminus S} P_j = h_3^{\sum_{j \notin S} z_j} h_1^{\sum_{j \notin S} r_j}$.

$\underline{\mathsf{CompKeyShare}(\mathsf{sk}_\ell, D, \tau) \to \sigma_\ell}$

- Sample $\rho_\ell, z_{\ell,0}, z_{\ell,1} \leftarrow\!\!\$\ \mathbb{Z}_N$ and output $\sigma_\ell := (K_{\ell,0}, K_{\ell,1})$, where

$$K_{\ell,0} := h_3^{z_{\ell,0}} h_1^{\rho_\ell}, \qquad K_{\ell,1} := h_3^{z_{\ell,1}} D^{u_\ell} h_1^{\alpha_\ell + (\beta + \gamma\tau)\rho_\ell}$$

$\underline{\mathsf{VerifyKeyShare}(\mathsf{pk}, D, \tau, \sigma) \to 0/1}$

- Parse the partial public key as $(A, U)$ and the key-share as $(K_0, K_1)$. Output 1 if

$$e(g_1, K_1) = A \cdot e(g_1^{\beta+\gamma\tau}, K_0) \cdot e(U, D)$$

  and 0 otherwise.

$\underline{\mathsf{CompKeyAggregate}(\mathsf{mpk}, \{\mathsf{pk}_\ell\}_{\ell \in [L]}, \boldsymbol{\sigma}, D, \tau) \to \bar{\sigma}/\perp}$

- Let $V$ be the set of indices $\ell \in [L]$ such that

$$\mathsf{VerifyKeyShare}(\mathsf{pk}_\ell, D, \tau, \sigma_\ell) = 1$$

- If $|V| < \theta$, output $\perp$. Otherwise output $\bar{\sigma} := (\overline{K}_0, \overline{K}_1)$, where $\overline{K}_a := \prod_{\ell \in V} K_{\ell,a}^{\omega_\ell^V}$ for $a \in \{0, 1\}$.

$\underline{\mathsf{Decrypt}(\mathsf{mpk}, \bar{\sigma}, D, S, (X, \tau), \mathsf{ct}) \to M/\perp}$

- Parse $\mathsf{ct} = (X', \tau', C_0, C_1, C_2, C_3)$ and $\bar{\sigma} = (\overline{K}_0, \overline{K}_1)$. Output $\perp$ unless $X' = X$, $\tau' = \tau$, $\emptyset \neq X \subseteq S \subseteq [n]$, and $D = D_S$.
- Compute $E_{X,S} := \prod_{i \in X} \prod_{j \in [n] \setminus S} P_{i,j}$. Since $X \subseteq S$, this only uses $P_{i,j}$ with $i \neq j$.
- Output

$$C_3 \cdot \frac{e(C_1, D) \cdot e(C_2, \overline{K}_0)}{e(C_0, \overline{K}_1) \cdot e(C_0, E_{X,S})}. \tag{3}$$

**Theorem 5.1.** *The scheme satisfies completeness and perfect correctness, as per Definition 3.3.*

*Proof.* Let $\sigma_\ell$ be an honest generated key share. Then using orthogonality,

$$\begin{aligned} e(g_1, K_{\ell,1}) &= e(g_1, h_1)^{\alpha_\ell} \cdot e(g_1, h_1^{\beta+\gamma\tau})^{\rho_\ell} \cdot e(g_1, D_S)^{u_\ell} \cdot 1 \\ &= A_\ell \cdot e(g_1^{\beta+\gamma\tau}, h_1^{\rho_\ell}) \cdot e(U_\ell, D_S) \\ &= A_\ell \cdot e(g_1^{\beta+\gamma\tau}, K_{\ell,0}) \cdot e(U_\ell, D_S) \end{aligned}$$

which is the verification check in VerifyKeyShare. Hence, the scheme satisfies *completeness*.

Let $\mathsf{ct} = (X, \tau, C_0, C_1, C_2, C_3)$ be an honestly generated ciphertext for $M$ under $(X, \tau)$.

$$C_0 := g_1^s, \quad C_1 := g_1^{us \sum_{i \in X} w_i}, \quad C_2 := g_1^{s(\beta+\gamma\tau)}, \quad C_3 := \Omega^s \cdot M$$

Let $\bar{\sigma} = (\overline{K}_0, \overline{K}_1)$ be the aggregate key for a $S$ such that $X \subseteq S$, then aggregating the above equation over $\theta$ shares gives

$$e(g_1, \overline{K}_1) = \Omega \cdot e(g_1^{\beta+\gamma\tau}, \overline{K}_0) \cdot e(g_1^u, D_S) \tag{4}$$

Raising (5) to the power $s$ gives

$$e(g_1^s, \overline{K}_1) = \Omega^s \cdot e(g_1^{s(\beta+\gamma\tau)}, \overline{K}_0) \cdot e(g_1^{us}, D_S) \tag{5}$$

which essentially is

$$e(C_0, \overline{K}_1) = \Omega^s \cdot e(C_2, \overline{K}_0) \cdot e(g_1^{su}, D_S)$$

Therefore

$$Y = \frac{e(C_1, D) \cdot e(C_2, \overline{K}_0)}{e(C_0, \overline{K}_1) \cdot e(C_0, E_{X,S})} = \frac{\Omega^{-s} \cdot e(C_1, D_S)}{e(g_1^{su}, D_S) \cdot e(C_0, E_{X,S})}$$

Now, $D_S = h_1^{R_S} Y_3$, and $E_{X,S} = h_1^{W_X R_S} Y_3'$ for some $Y_3, Y_3' \in H_3$. Let $W_X = \sum_{i \in X} w_i$. Substituting we get,

$$\begin{aligned}
e(C_1, D_S) &= e\big(g_1^{s(u+W_X)}, h_1^{R_S}\big) \cdot e\big(g_1^{s(u+W_X)}, Y_3\big) = e(g_1, h_1)^{s(u+W_X)R_S}, \\
e(g_1^{su}, D_S) &= e\big(g_1^{su}, h_1^{R_S}\big) \cdot e\big(g_1^{su}, Y_3\big) = e(g_1, h_1)^{suR_S}, \\
e(C_0, E_{X,S}) &= e\big(g_1^s, h_1^{W_X R_S}\big) \cdot e\big(g_1^s, Y_3'\big) = e(g_1, h_1)^{sW_X R_S}.
\end{aligned}$$

Thus,

$$Y = \Omega^{-s} \cdot e(g_1, h_1)^{s(u+W_X)R_S - suR_S - sW_X R_S} = \Omega^{-s} \cdot e(g_1, h_1)^0 = \Omega^{-s}$$

and decryption outputs

$$C_3 \cdot Y = \Omega^s \cdot M \cdot \Omega^{-s} = M.$$

□

**Sizes and efficiency.** The public key is made up of $n^2 + n + 2L + 9$ group elements: $n + L + 4$ in $\mathbb{G}_N$, $n^2 + 4$ in $\mathbb{H}_N$ and $L + 1$ in $\mathbb{G}_T$. A ciphertext is three elements of $\mathbb{G}_N$ and one element of $\mathbb{G}_T$. A digest is one element of $\mathbb{H}_n$. Both the partial key and the batch key are two elements of $\mathbb{H}_N$ each. The authority's secret key share is simply one group element and one exponent. Encrypt is $O(|X|)$ group operations and Digest is $O(n - |S|)$ multiplications. CompKeyShare is only $O(1)$ exponentiations so is independent of $n, |S|, L$ and $\theta$. To validate the key shares we evaluate at most $3L$ pairings in CompKeyAggregate. Lastly, Decrypt performs $O(n + |X|(n - |S|))$ source group multiplications and $O(1)$ pairings.

## 5.2 Security Analysis

We now prove adaptive challenge security under static corruption described in Definition 3.3.

**Theorem 5.2.** *Let $(n, L, \theta)$ be as in Definition 3.2 and $\mathcal{A}$ be an admissible PPT adversary in the static-corruption experiment of Definition 3.3 that makes at most $Q$ key queries. Then there exist PPT adversaries $\mathcal{B}_{\mathrm{SDG}}$ against $\mathsf{SD}_G$, $\mathcal{B}_{\mathrm{PAIR}}$ against $\mathsf{PAIR}$, and $\mathcal{B}_{\mathrm{res}}, \mathcal{B}_1, \ldots, \mathcal{B}_Q, \mathcal{B}_{\mathrm{col}}, \mathcal{B}_{\mathrm{rm}}$ against $\mathsf{SD}_H$, such that*

$$\mathsf{Adv}^{\mathsf{ThSUB},\mathsf{st}}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta) \;\le\; 2n(Q+1)\,\Delta, \tag{6}$$

*where*

$$\begin{aligned}\Delta := {} & \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\mathrm{res}}}(\lambda) + \mathsf{Adv}^{\mathsf{SD}_G}_{\mathcal{B}_{\mathrm{SDG}}}(\lambda) + \sum_{t=1}^{Q} \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_t}(\lambda) + \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\mathrm{col}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\mathrm{rm}}}(\lambda) + \mathsf{Adv}^{\mathsf{PAIR}}_{\mathcal{B}_{\mathrm{PAIR}}}(\lambda) + \frac{2Q+7}{2^{\lambda}}.\end{aligned} \tag{7}$$

*In particular, if the $\mathsf{SD}_G$, $\mathsf{SD}_H$, and $\mathsf{PAIR}$ assumptions (Assumptions 3.1, 3.2, and 3.3) hold, then the scheme $\Theta$ of Section 5 has adaptive challenge security against the static corruption of at most $\theta - 1$ authorities.*

We prove the theorem by a sequence of hybrids. For the security proof, we first define the following:

**Sharing polynomials.** For the corruption set $C := S_{\mathsf{corr}}$ submitted by $\mathcal{A}$, let

$$v_C(Z) := \prod_{c \in C} \Big(1 - \frac{Z}{\xi_c}\Big), \qquad v_\ell := v_C(\xi_\ell).$$

Then $\deg v_C = |C| \le \theta - 1$, $v_C(0) = 1$, and $v_c = 0$ for every $c \in C$. For $C = \emptyset$ we have $v_C = 1$. We sample the sharing polynomials as

$$q_\alpha := \widetilde{q}_\alpha + \alpha\, v_C, \qquad q_u := \widetilde{q}_u + u\, v_C, \tag{8}$$

where $\widetilde{q}_\alpha, \widetilde{q}_u$ are independent uniform polynomials of degree at most $\theta - 1$ with constant term 0, and we write $\widetilde{\alpha}_\ell := \widetilde{q}_\alpha(\xi_\ell)$ and $\widetilde{u}_\ell := \widetilde{q}_u(\xi_\ell)$. This gives the same distribution of $(q_\alpha, q_u)$ as $\mathsf{KeyGen}$. Moreover $\alpha_\ell = \widetilde{\alpha}_\ell + v_\ell \alpha$ and $u_\ell = \widetilde{u}_\ell + v_\ell u$, so the verification keys and the corrupted secret key shares are

$$A_\ell = e(g_1, h_1)^{\widetilde{\alpha}_\ell}\, \Omega^{v_\ell}, \qquad U_\ell = g_1^{\widetilde{u}_\ell} (g_1^u)^{v_\ell}, \qquad sk_c = \big(h_1^{\widetilde{\alpha}_c}, \widetilde{u}_c\big) \quad (c \in C). \tag{9}$$

Recall that

$$\begin{aligned}\mathsf{mpk} := \big(&\mathsf{pp}, g_1^u, g_1^\beta, g_1^\gamma, h_1^\beta, h_1^\gamma, \Omega, \\ &\{g_1^{w_i}\}_{i\in[n]}, \{P_j\}_{j\in[n]}, \{P_{i,j}\}_{i\neq j}, \{\mathsf{pk}_\ell\}_{\ell\in[L]}\big).\end{aligned}$$

Therefore, these values can be computed from the values $\Omega$, $g_1^u$, $\widetilde{q}_\alpha$, and $\widetilde{q}_u$ without knowing $\alpha$ or $u$, which allows the reductions below to embed $\alpha$ or $u$ from their challenge instances.

**Notation.** For the $k$-th key query $(S_k, \tau_k)$ we write $R_{S_k} := \sum_{j \notin S_k} r_j$ and $t_k := \beta + \gamma\tau_k$, and for a policy $X$ we write $W_X := \sum_{i \in X} w_i$. Let $T_\tau^G = g_1^\beta (g_1^\gamma)^\tau$ and $T_\tau^H = h_1^\beta (h_1^\gamma)^\tau$. We let $g_2$ and $h_2$ denote generators of $\mathbb{G}_2$ and $\mathbb{H}_2$. They are used only in the security proof, and their exponents matter only modulo $p_2$.

**Semi-functional Objects.** We use the dual-system technique of Waters and Lewko–Waters [23, 19].

For a challenge policy-tag pair $(X^*, \tau^*)$, a semi-functional challenge ciphertext has the form

$$\begin{aligned} C_0 &= g_1^s\, g_2^x, & C_1 &= g_1^{s(u+W_{X^*})}\, g_2^{x\mu}, \\ C_2 &= g_1^{s(\beta+\gamma\tau^*)}\, g_2^{xz^*}, & C_3 &= e(g_1, h_1)^{\alpha s} M_b, \end{aligned} \tag{10}$$

where $s \leftarrow\$ \mathbb{Z}_N$ and $x, \mu, z^* \leftarrow\$ \mathbb{Z}_{p_2}$.

In the threshold setting, the answer to the $k$-th key query $(S_k, \tau_k)$ consists of one partial key for every honest authority. We call the tuple $sk_k := (K_{\ell,0,k}, K_{\ell,1,k})_{\ell \notin C}$ the *key* for the $k$-th query. A key is normal if it is honestly generated using $\mathsf{CompKeyShare}(sk_\ell, D_{S_k}, \tau_k)_{\ell \notin C}$. A semi-functional key for the $k$-th query has the form

$$\begin{aligned} K_{\ell,0,k} &= R_{3,\ell,0,k}\, h_1^{\rho_{\ell,k}}\, h_2^{v_\ell y_k}, \\ K_{\ell,1,k} &= R_{3,\ell,1,k}\, h_1^{\alpha_\ell + t_k \rho_{\ell,k} + u_\ell R_{S_k}}\, h_2^{\widetilde{u}_\ell \delta_{S_k} + v_\ell c_k} \qquad (\ell \notin C), \end{aligned} \tag{11}$$

where $\rho_{\ell,k} \leftarrow\$ \mathbb{Z}_N$ are independent for every authority, as in $\mathsf{CompKeyShare}$, and $y_k, c_k \leftarrow\$ \mathbb{Z}_{p_2}$ are fresh and independent for each key and shared by all honest keys of the query. The value $\delta_{S_k}$ is the $\mathbb{H}_2$-exponent of the digest $D_{S_k}$, defined in (13) below. The term $\widetilde{u}_\ell \delta_{S_k}$ is its contribution to the factor $D_{S_k}^{u_\ell}$, up to the part $v_\ell u \delta_{S_k}$, which is absorbed in $c_k$. A normal key is the case $y_k = c_k = 0$ with a normal public key. Corrupted authorities compute their partial keys locally from $sk_c$.

For the guessed coordinate $\widehat{\imath}$, a type-$\widehat{\imath}$ public key is a public key in which the column indexed by $\widehat{\imath}$ has an $\mathbb{H}_2$-component, as in the broadcast encryption scheme of Gay, Kowalczyk, and Wee [13]:

$$P_{\widehat{\imath}} = R_{3,\widehat{\imath}}\, h_1^{r_{\widehat{\imath}}}\, h_2^{r'}, \qquad P_{a,\widehat{\imath}} = R_{3,a,\widehat{\imath}}\, h_1^{w_a r_{\widehat{\imath}}}\, h_2^{w_a r'} \quad (a \neq \widehat{\imath}), \tag{12}$$

where $r' \leftarrow\$ \mathbb{Z}_{p_2}$. The $\mathbb{H}_2$-exponent of a digest $D_S$ is

$$\delta_S := \begin{cases} r' & \text{if the public key is of type } \widehat{\imath} \text{ and } \widehat{\imath} \notin S, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Let $Q$ be a polynomial upper bound on the number of key queries. Let $j^* \in [Q] \cup \{\perp\}$ be the index of the key query with tag $\tau^*$, or $\perp$ if there is none; it is well defined because there can be at most one query with the challenge tag. Let $S^* := S_{j^*}$ if $j^* \neq \perp$ and $S^* := \emptyset$ otherwise and let $i^\dagger := \min(X^* \setminus S^*)$, which is well defined by the admissibility condition. The challenger samples the challenge bit $b \leftarrow\$ \{0,1\}$ and encrypts $M_b$. The game outputs 1 if $\mathcal{A}$ outputs $b'$ and $b' = b$. For a game $\mathsf{G}$ we write

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}} := \left| \Pr[\mathsf{G} = 1] - \frac{1}{2} \right|,$$

**Game $\mathsf{G}_{\text{real}}$.** This is the real experiment for adaptive challenge security under static corruption from Definition 3.3, with the sharing polynomials sampled as in (8).

**Game $\mathsf{G}_{\text{guess}}$.** At the beginning of the game, the challenger guesses

$$\widehat{\imath} \leftarrow\$ [n], \qquad \widehat{\jmath} \leftarrow\$ \{1, \ldots, Q, \perp\}.$$

The value $\widehat{\jmath} = \perp$ means that no key query will be made on the challenge tag. The challenger aborts unless $\widehat{\imath} = i^{\dagger}$ and $\widehat{\jmath} = j^{*}$. When the query $\widehat{\jmath}$ arrives, the challenger aborts if $\widehat{\imath} \in S_{\widehat{\jmath}}$. We call the key for the query $\widehat{\jmath}$ the *special* key, and all other keys *non-special*.

**Game $\mathsf{G}_{\text{res}}$.** This is the same as $\mathsf{G}_{\text{g}}$, except that the challenger additionally aborts if there exists a key query $k \neq \widehat{\jmath}$ such that

$$\tau_k \equiv \tau^{*} \pmod{p_2}.$$

**Game $\mathsf{G}_0$.** This is the same as $\mathsf{G}_{\text{res}}$, except that the challenge ciphertext is semi-functional.

**Games $\mathsf{G}_t$ for $t = 1, \ldots, Q$.** Let $\mathcal{L}$ be the ordered list of all key-query indices excluding $\widehat{\jmath}$ and if $\widehat{\jmath} = \perp$, then $\mathcal{L}$ is the list of all key-query indices. Game $\mathsf{G}_t$ is the same as $\mathsf{G}_0$, except that the first $t$ keys in the list $\mathcal{L}$ are semi-functional. All remaining keys, including the special key, are normal. The challenge ciphertext remains semi-functional.

**Game $\mathsf{G}_{\text{col}}$.** This hybrid is needed only when $\widehat{\jmath} \neq \perp$. It is the same as $\mathsf{G}_Q$, except that the public key is type $\widehat{\imath}$ and the special key $\widehat{\jmath}$ is also semi-functional. All digests are computed from the type-$\widehat{\imath}$ public key, so every key has the form (11) with $\delta_{S_k}$ as in (13).

**Game $\mathsf{G}_{\text{all}}$.** This is the same as $\mathsf{G}_{\text{col}}$, except that the public key is returned to the normal form. All keys remain semi-functional.

**Game $\mathsf{G}_{\alpha'}$.** This is the same as $\mathsf{G}_{\text{all}}$, except that the the second component $K_{\ell,1,k}$ of every honest partial key is multiplied by $h_2^{v_\ell \alpha'}$ for a single $\alpha' \leftarrow\$ \mathbb{Z}_{p_2}$. Thus the $k$-th key has $c_k$ replaced by $\alpha' + c_k$.

**Game $\mathsf{G}_{\text{final}}$.** This is the same as $\mathsf{G}_{\alpha'}$, except that the last component of the challenge ciphertext is replaced by a uniform random element $R \leftarrow\$ \mathbb{G}_T$: $C_3 := R$.

**Lemma 5.1.** *For any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_{\text{guess}}} = \frac{1}{n(Q+1)} \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_{\text{real}}}.$$

*Proof.* The guesses $(\widehat{i}, \widehat{j})$ are made before the adversary's view is generated and are independent of the transcript. By admissibility, there is exactly one pair $(\widehat{i}, \widehat{j})$ that equals $(i^\dagger, j^*)$, so the probability that both guesses are correct is $1/(n(Q+1))$. Conditioned on this event, this game is identical to the real game. Hence, $\Pr[\mathsf{G}_{\text{guess}} = 1] = \frac{1}{n(Q+1)} \Pr[\mathsf{G}_{\text{real}} = 1]$.

**Lemma 5.2.** *There is a PPT algorithm $\mathcal{B}_{\text{res}}$ such that*

$$\left|\Pr[\mathsf{G}_{\text{guess}} = 1] - \Pr[\mathsf{G}_{\text{res}} = 1]\right| \leq \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\text{res}}}(\lambda) + \frac{3}{p_2}.$$

*Proof.* Let $\mathsf{Bad}$ be the event that $\mathsf{G}_{\text{guess}}$ does not abort and some key query $k \neq \widehat{j}$ satisfies $\tau_k \equiv \tau^* \pmod{p_2}$. The two games are identical unless $\mathsf{Bad}$ occurs, so their output probabilities differ by at most $\Pr[\mathsf{Bad}]$. We bound $\Pr[\mathsf{Bad}^{(b)}]$ as in [19, Lemma 5].

The reduction takes an $\mathsf{SD}_H$ instance $(D, T)$ with $D = (g_1, X_{12}, X_3, Y_{23}, h_1, h_3, W_{23})$ and $T \leftarrow\!\!\$\ \mathbb{H}_{13}$ or $T \leftarrow\!\!\$\ \mathbb{H}_{123}$. It samples all scheme parameters itself and simulates $\mathsf{G}_{\text{guess}}$ exactly as in the real game, without using $T$. For every key query $k \neq \widehat{j}$ with $\tau_k \neq \tau^*$, it computes $d := \gcd(\tau_k - \tau^*, N)$ and distinguishes three cases.

- If $g_1^d = h_3^d = 1$, then $d = p_1 p_3$ and $p_2 \nmid d$.
- If $g_1^d \neq 1$, then $p_1 \mid N/d$. The element $T^{N/d}$ has no $\mathbb{H}_1$-component, and $e(X_{12}, T^{N/d})$ is nontrivial exactly when $p_2 \mid d$ and $T$ has an $\mathbb{H}_2$-component.
- If $g_1^d = 1$ and $h_3^d \neq 1$, then $p_3 \mid N/d$. The element $T^{N/d}$ has no $\mathbb{H}_3$-component, and $e(Y_{23}, T^{N/d})$ is nontrivial exactly when $p_2 \mid d$ and $T$ has an $\mathbb{H}_2$-component.

Here we used orthogonality and that the $\mathbb{G}_2$-components of $X_{12}$ and $Y_{23}$ are nontrivial. The reduction outputs 1 if one of these pairings is nontrivial and the guess is correct, and 0 otherwise.

If $T \in \mathbb{H}_{13}$, the reduction never outputs 1. If $T \in \mathbb{H}_{123}$, it outputs 1 whenever $\mathsf{Bad}$ occurs, unless the $\mathbb{G}_2$-component of $X_{12}$ or of $Y_{23}$ or the $\mathbb{H}_2$-component of $T$ is trivial, which happens with probability at most $3/p_2$. Hence $\Pr[\mathsf{Bad}] \leq \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\text{res}}}(\lambda) + 3/p_2$.

**Lemma 5.3.** *There is a PPT algorithm $\mathcal{B}_{\text{SDG}}$ such that*

$$\left|\Pr[\mathsf{G}_{\text{res}} = 1] - \Pr[\mathsf{G}_0 = 1]\right| \leq \mathsf{Adv}^{\mathsf{SD}_G}_{\mathcal{B}_{\text{SDG}}}(\lambda).$$

*Proof.* The reduction $\mathcal{B}_{\text{SDG}}$ receives an $\mathsf{SD}_G$ instance $(D, T)$, where $D = (N, \mathbb{G}_N, \mathbb{H}_N, \mathbb{G}_T, e, g_1, X_3, h_1, Y_3)$ and $T \leftarrow\!\!\$\ \mathbb{G}_1$ or $T \leftarrow\!\!\$\ \mathbb{G}_{12}$. It sets $h_3 := Y_3$ and samples $\alpha, u, \beta, \gamma, \{w_i, r_i\}_{i \in [n]}$ , and all $\mathbb{H}_3$-masks. Publishes the normal public key. It knows every $sk_\ell$ and answers all key queries normally.

For the challenge ciphertext for $(X^*, \tau^*)$, it sets

$$C_0 = T, \quad C_1 = T^{u + W_{X^*}}, \quad C_2 = T^{\beta + \gamma\tau^*}, \quad C_3 = e(T, h_1)^\alpha M_b.$$

If $T \in G_1$, this is $\mathsf{G}_{\text{res}}$. If $T = g_1^s g_2^x \in G_{12}$, the $G_2$-exponents of $C_1$ and $C_2$ are $x(u + W_{X^*})$ and $xt^*$, and $C_3 = e(g_1, h_1)^{\alpha s} M_b$ by orthogonality. The residues $u \bmod p_2$ and $\beta \bmod p_2$ appear

only in these two $G_2$-exponents: the public key and the keys contain $u$, $u_\ell$, and $\beta$ only in $G_1$, $H_1$, and $H_3$ components, and the corrupted states contain $\tilde{u}_c$ but not $u$. Replacing them by fresh residues makes $\mu = u + W_{X^*}$ and $z^* = t^*$ uniform and independent modulo $p_2$, so this is $\mathsf{G}_0$.

**Lemma 5.4.** *Let $\mathcal{L}[t]$ be the $t$-th index in the ordered list $\mathcal{L}$. For every $t = 1, \ldots, Q$, there is a PPT algorithm $\mathcal{B}_t$ such that*

$$\left|\Pr[\mathsf{G}_{t-1} = 1] - \Pr[\mathsf{G}_t = 1]\right| \leq \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_t}(\lambda) + \frac{2}{p_2}.$$

*Proof.* If $\mathcal{L}$ has fewer than $t$ entries, the two games coincide. Otherwise let $k = \mathcal{L}[t]$ be the key query being switched. The reduction receives an $\mathsf{SD}_H$ instance

$$(D, T) = (g_1, X_{12}, X_3, Y_{23}, h_1, h_3, W_{23}, T),$$

where $T \leftarrow\!\!\$\ \mathbb{H}_{13}$ or $T \leftarrow\!\!\$\ \mathbb{H}_{123}$. It samples $\alpha, u, \beta, \gamma, \{w_i, r_i\}_{i\in[n]}$ and publishes a normal public key, whose $\mathbb{H}_3$-masks are random powers of $h_3$. It uses $X_{12}$ to form the semi-functional challenge ciphertext:

$$C_0 = X_{12}, \quad C_1 = X_{12}^{u+W_{X^*}}, \quad C_2 = X_{12}^{\beta+\gamma\tau^*}, \quad C_3 = e(X_{12}, h_1)^\alpha M_b. \tag{14}$$

Let $W_{23} = h_2^\eta h_3^\vartheta$. For every previously switched key $k' = \mathcal{L}[j]$ with $j < t$, the reduction samples fresh $a_{k'}, b_{k'} \leftarrow\!\!\$\ \mathbb{Z}_N$ and $\rho_{\ell,k'} \leftarrow\!\!\$\ \mathbb{Z}_N$, and outputs

$$K_{\ell,0,k'} = R_{3,\ell,0,k'}\, h_1^{\rho_{\ell,k'}}\, W_{23}^{v_\ell a_{k'}}, \qquad K_{\ell,1,k'} = R_{3,\ell,1,k'}\, h_1^{\alpha_\ell}\, (T^H_{\tau_{k'}})^{\rho_{\ell,k'}}\, D_{S_{k'}}^{u_\ell}\, W_{23}^{v_\ell b_{k'}} \qquad (\ell \notin C). \tag{15}$$

Conditioned on $\eta \not\equiv 0 \bmod p_2$, the values $y_{k'} = \eta a_{k'}$ and $c_{k'} = \eta b_{k'}$ are fresh and uniform over $\mathbb{Z}_{p_2}$ across all switched keys, so these keys are semi-functional. All future non-special keys and the special key are answered normally.

For the $k$-th query $(S_k, \tau_k)$, the reduction samples fresh $z_{\ell,k} \leftarrow\!\!\$\ \mathbb{Z}_N$ and outputs

$$K_{\ell,0,k} = R_{3,\ell,0,k}\, h_1^{z_{\ell,k}}\, T^{v_\ell}, \qquad K_{\ell,1,k} = R_{3,\ell,1,k}\, h_1^{\alpha_\ell}\, (T^H_{\tau_k})^{z_{\ell,k}}\, D_{S_k}^{u_\ell}\, T^{v_\ell t_k} \qquad (\ell \notin C).$$

Write $T = h_1^{\rho_k} h_2^{y_k} h_3^{\phi_k}$. The $\mathbb{H}_1$-exponents of this key are $\rho_{\ell,k} := z_{\ell,k} + v_\ell \rho_k$ and $\alpha_\ell + t_k \rho_{\ell,k} + u_\ell R_{S_k}$; since the $z_{\ell,k}$ are fresh, the $\rho_{\ell,k}$ are uniform and independent, as in a normal key.

If $T \in \mathbb{H}_{13}$, then $y_k = 0$ and the $k$-th key is normal, so $\mathcal{B}_t$ simulates $\mathsf{G}_{t-1}$.

If $T \in \mathbb{H}_{123}$, the $k$-th key has $\mathbb{H}_2$-exponents $v_\ell y_k$ and $v_\ell y_k(\beta + \gamma\tau_k)$, that is, $c_k = y_k \zeta_k$ with $\zeta_k := \beta + \gamma\tau_k \bmod p_2$; in the terminology of [19], it is nominally semi-functional. By the Chinese Remainder Theorem, the residues of $\beta, \gamma$ modulo $p_2$ are independent of their residues modulo $p_1$ and $p_3$, which determine the public key, the corrupted shares, and the normal keys, and the previously switched keys use independent powers of $W_{23}$. Hence $\beta, \gamma \bmod p_2$ are hidden from $\mathcal{A}$ except through $\zeta_k$ and $z^* = \beta + \gamma\tau^*$. Since $k \neq \widehat{\jmath}$, the restricted game guarantees $\tau_k \not\equiv \tau^* \pmod{p_2}$, so the map

$$(\beta, \gamma) \longmapsto (\zeta_k, z^*)$$

has matrix $\begin{pmatrix} 1 & \tau_k \\ 1 & \tau^* \end{pmatrix}$ with determinant $\tau^* - \tau_k \neq 0$ in $\mathbb{Z}_{p_2}$, and is a bijection. This is the pairwise independence of the affine tag of [19]. It holds in either order of the $k$-th query and the challenge: whichever of $\zeta_k$ and $z^*$ is fixed first, the other equals it plus $\gamma(\tau_k - \tau^*)$ or $\gamma(\tau^* - \tau_k)$, and $\gamma \bmod p_2$ is still uniform at that time. Therefore $\zeta_k$ is uniform and independent of $z^*$. Conditioned on $y_k \not\equiv 0 \bmod p_2$, the value $c_k = y_k \zeta_k$ is uniform and independent of $y_k$, of $z^*$, and of the pads of all previously switched keys. Thus the $k$-th key is semi-functional with fresh independent pads, and $\mathcal{B}_t$ simulates $\mathsf{G}_t$.

The excluded events $\eta \equiv 0$ and $y_k \equiv 0 \bmod p_2$ have probability $1/p_2$ each.

**Lemma 5.5.** *There is a PPT algorithm $\mathcal{B}_{\mathrm{col}}$ such that*

$$\left|\Pr[\mathsf{G}_Q = 1] - \Pr[\mathsf{G}_{\mathrm{col}} = 1]\right| \leq \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\mathrm{col}}}(\lambda) + \frac{2}{p_2}.$$

*Proof.* If $\widehat{\jmath} = \perp$, the two games are identical by definition. Otherwise, let the special query be $(S_{\widehat{\jmath}}, \tau_{\widehat{\jmath}})$. On the non-abort branch,

$$\tau_{\widehat{\jmath}} = \tau^*, \qquad \widehat{\imath} \notin S_{\widehat{\jmath}}, \qquad \widehat{\imath} \in X^*.$$

The reduction receives an $\mathsf{SD}_H$ instance $(D, T) = (g_1, X_{12}, X_3, Y_{23}, h_1, h_3, W_{23}, T)$, where $T \leftarrow\!\!\$\ \mathbb{H}_{13}$ or $T \leftarrow\!\!\$\ \mathbb{H}_{123}$. It samples $\alpha, u, \beta, \gamma, \{w_i\}_{i\in[n]}$, $\{r_j\}_{j\neq\widehat{\imath}}$, and places $T$ in the $\widehat{\imath}$ public-key column:

$$P_{\widehat{\imath}} := T, \qquad P_{a,\widehat{\imath}} := T^{w_a} R_{3,a,\widehat{\imath}} \quad (a \neq \widehat{\imath}).$$

The rest of the public key is formed as in the scheme. Write $T = h_1^{r_{\widehat{\imath}}} h_2^{r'} h_3^{\phi}$. If $T \in \mathbb{H}_{13}$, then $r' = 0$ and this is a normal public key. If $T \in \mathbb{H}_{123}$, this is a type-$\widehat{\imath}$ public key. The reduction does not know $r_{\widehat{\imath}}$, but it computes every digest from the public key it publishes. The challenge ciphertext is generated from $X_{12}$ as in (14).

*Non-special keys.* For each non-special key query $k \neq \widehat{\jmath}$, the reduction answers with (15). If $\widehat{\imath} \in S_k$, the digest $D_{S_k}$ does not contain $T$. If $\widehat{\imath} \notin S_k$, the factor $D^{u_\ell}_{S_k}$ contains $T^{u_\ell}$, which supplies the term $h_1^{u_\ell r_{\widehat{\imath}}}$ needed in the exponent of the second component. Thus the non-special keys have pads

$$y_k = \eta a_k, \qquad c_k = \eta b_k + u\delta_{S_k}.$$

*Special key.* The reduction samples $\bar{\rho} \leftarrow\!\!\$\ \mathbb{Z}_N$ and fresh $z_{\ell,\widehat{\jmath}} \leftarrow\!\!\$\ \mathbb{Z}_N$, and outputs

$$K_{\ell,0,\widehat{\jmath}} = R_{3,\ell,0,\widehat{\jmath}} h_1^{z_{\ell,\widehat{\jmath}}} T^{v_\ell \bar{\rho}}, \qquad K_{\ell,1,\widehat{\jmath}} = R_{3,\ell,1,\widehat{\jmath}} h_1^{\alpha_\ell} (T^H_{\tau_{\widehat{\jmath}}})^{z_{\ell,\widehat{\jmath}}} D^{u_\ell}_{S_{\widehat{\jmath}}} T^{v_\ell \bar{\rho} t_{\widehat{\jmath}}} \qquad (\ell \notin C).$$

Its $\mathbb{H}_1$-exponents are those of a normal key with $\rho_{\ell,\widehat{\jmath}} = z_{\ell,\widehat{\jmath}} + v_\ell \bar{\rho}\, r_{\widehat{\imath}}$, which is uniform and independent across $\ell$ because of $z_{\ell,\widehat{\jmath}}$. Thus, in the $T \in \mathbb{H}_{13}$ case, the special key is normal, the non-special keys are semi-functional, and the challenge ciphertext is semi-functional by the same Chinese Remainder Theorem argument as in Lemma 5.3; the reduction simulates $\mathsf{G}_Q$.

Now suppose $T \in \mathbb{H}_{123}$, and let $z^* := \beta + \gamma\tau^* = t_{\widehat{\jmath}} \bmod p_2$. The $\mathbb{H}_2$-exponents of the special key are $v_\ell \bar{\rho} r'$ and $\widetilde{u}_\ell r' + v_\ell(u + \bar{\rho} z^*) r'$, so it has the form (11) with

$$y_{\widehat{\jmath}} = \bar{\rho} r', \qquad c_{\widehat{\jmath}} = (u + \bar{\rho} z^*) r'.$$

The $\mathbb{G}_2$-exponent of $C_1$ is $x\mu$ with

$$\mu = u + W_- + w_{\widehat{\imath}}, \qquad W_- := \sum_{a \in X^* \setminus \{\widehat{\imath}\}} w_a.$$

It remains to show that $y_{\widehat{\jmath}}, c_{\widehat{\jmath}}, \mu, z^*$ and the non-special pads $\{c_k\}_{k \neq \widehat{\jmath}}$ are uniform and independent.

By the Chinese Remainder Theorem, the residues of $u, w_{\widehat{\imath}}, \beta, \bar{\rho}, \{b_k\}_{k \neq \widehat{\jmath}}$ modulo $p_2$ are independent of their residues modulo $p_1$ and $p_3$, which determine the $\mathbb{G}_1$-, $\mathbb{H}_1$-, and $\mathbb{H}_3$-parts of the view, and the corrupted shares do not involve $u$. The type-$\widehat{\imath}$ column fixes $r'$ and $w_a$ mod $p_2$ for every row identity $a \neq \widehat{\imath}$, and hence $W_-$, but not $w_{\widehat{\imath}}$ mod $p_2$, since the diagonal element $P_{\widehat{\imath},\widehat{\imath}}$ is not published; this is the missing-diagonal argument of [13].

Condition on $r' \not\equiv 0$ and $\eta \not\equiv 0 \bmod p_2$. Each of the residues $w_{\widehat{\imath}}$, $\beta$, $\bar{\rho}$, and $b_k$ occurs in exactly one exponent visible to $\mathcal{A}$, with a nonzero coefficient: $w_{\widehat{\imath}}$ only in $\mu$, $\beta$ only through $z^*$, $\bar{\rho}$ only in $y_{\widehat{\jmath}}$, and $b_k$ only in $c_k$. Each therefore acts as a one-time pad, so $\mu$, $z^*$, $y_{\widehat{\jmath}}$, and every $c_k$ are uniform and independent; this holds even though $X^*$, $\tau^*$, and the sets $S_k$ are chosen adaptively, since each pad is used only once. The residue $u$ then occurs only in $c_{\widehat{\jmath}} = ur' + y_{\widehat{\jmath}} z^*$, so $c_{\widehat{\jmath}}$ is uniform and independent as well. Thus the reduction simulates $\mathsf{G}_{\mathrm{col}}$.

The excluded events $r' \equiv 0$ and $\eta \equiv 0 \bmod p_2$ have probability $1/p_2$ each.

**Lemma 5.6.** *There is a PPT algorithm $\mathcal{B}_{\mathrm{rm}}$ such that*

$$\left|\Pr[\mathsf{G}_{\mathrm{col}} = 1] - \Pr[\mathsf{G}_{\mathrm{all}} = 1]\right| \leq \mathsf{Adv}^{\mathsf{SD}_H}_{\mathcal{B}_{\mathrm{rm}}}(\lambda) + \frac{1}{p_2}.$$

*Proof.* If $\widehat{\jmath} = \perp$, the two games are identical. Otherwise, the reduction again receives an $\mathsf{SD}_H$ instance $(D, T)$ with $D = (g_1, X_{12}, X_3, Y_{23}, h_1, h_3, W_{23})$ and target $T \leftarrow\!\!\$\ \mathbb{H}_{13}$ or $T \leftarrow\!\!\$\ \mathbb{H}_{123}$. It places $T$ in the $\widehat{\imath}$ column exactly as in Lemma 5.5, and forms the challenge ciphertext from $X_{12}$ as in (14). All key queries, including the special one, are answered as semi-functional keys by (15); for a query with $\widehat{\imath} \notin S_k$, the factor $D_{S_k}^{u_\ell}$ again uses $T^{u_\ell}$ to supply the missing $h_1^{u_\ell r_{\widehat{\imath}}}$ term.

If $T \in \mathbb{H}_{13}$, the public key is normal, and the keys have pads $(\eta a_k, \eta b_k)$, so the reduction simulates $\mathsf{G}_{\mathrm{all}}$. If $T \in \mathbb{H}_{123}$, the public key is of type $\widehat{\imath}$, and $T^{u_\ell}$ contributes the shift $ur'$ to the second pad of every key with $\widehat{\imath} \notin S_k$. As in the previous lemma, the independent fresh values $\eta b_k$ absorb this shift, so the pads remain jointly uniform and independent, and the reduction simulates $\mathsf{G}_{\mathrm{col}}$. In both cases the challenge ciphertext is semi-functional by the Chinese Remainder Theorem: the residues of $u, \beta, \gamma, w_1, \ldots, w_n$ modulo $p_2$ are independent of their residues modulo $p_1$ and $p_3$, $u$ mod $p_2$ is hidden by the pads $\eta b_k$, and $w_{\widehat{\imath}}$ mod $p_2$ occurs only in $C_1$. The excluded event $\eta \equiv 0 \bmod p_2$ has probability $1/p_2$.

**Lemma 5.7.**

$$\Pr[\mathsf{G}_{\mathrm{all}} = 1] = \Pr[\mathsf{G}_{\alpha'} = 1].$$

*Proof.* In $\mathsf{G}_{\mathrm{all}}$, the values $c_1, \ldots, c_q$ appearing in the second components of the semi-functional keys are independent uniform elements of $\mathbb{Z}_{p_2}$. If $\alpha' \leftarrow\!\!\$\ \mathbb{Z}_{p_2}$ is sampled once and shared across all keys, then

$$(c_1, \ldots, c_q) \quad \text{and} \quad (\alpha' + c_1, \ldots, \alpha' + c_q)$$

are identically distributed, because the $c_k$ are independent one-time pads. In $\mathsf{G}_{\alpha'}$, the second $\mathbb{H}_2$-exponent of the partial key of authority $\ell$ is $v_\ell(\alpha' + c_k)$, so all keys are distributed as in $\mathsf{G}_{\text{all}}$. Since $v_c = 0$ for $c \in C$, the corrupted shares are not affected. This is only a relabeling of the independent pads.

**Lemma 5.8.** *There is a PPT algorithm $\mathcal{B}_{\text{PAIR}}$ such that*

$$\left|\Pr[\mathsf{G}_{\alpha'} = 1] - \Pr[\mathsf{G}_{\text{final}} = 1]\right| \le \mathsf{Adv}^{\mathsf{PAIR}}_{\mathcal{B}_{\text{PAIR}}}(\lambda) + \frac{1}{p_2}.$$

*Proof.* The reduction receives a PAIR instance

$$D = \big(g_1,\ g_1^\alpha X_2,\ X_3,\ g_1^s Y_2,\ Z_2,\ h_1,\ h_3,\ h_1^\alpha X_2',\ W_{23}\big)$$

and a target $T$, which is either $e(g_1, h_1)^{\alpha s}$ or a random element of $\mathbb{G}_T$. It samples $u, \beta, \gamma, \{w_i, r_i\}_{i\in[n]}$ and generates the normal public key, computing

$$\Omega = e(g_1, h_1)^\alpha = e(g_1, h_1^\alpha X_2')$$

by orthogonality. It computes the verification keys and the corrupted shares by (9), which does not require $\alpha$.

Write, for analysis, $W_{23} = h_2^\eta h_3^\vartheta$ and $X_2' = h_2^{\alpha'}$. For every key query $(S_k, \tau_k)$, the reduction samples fresh $a_k, b_k, \rho_{\ell,k} \leftarrow\!\!\$\ \mathbb{Z}_N$ and returns

$$K_{\ell,0,k} = R_{3,\ell,0,k}\, h_1^{\rho_{\ell,k}}\, W_{23}^{v_\ell a_k}, \qquad K_{\ell,1,k} = R_{3,\ell,1,k}\, h_1^{\widetilde{\alpha}_\ell} (h_1^\alpha X_2')^{v_\ell}\, (T^H_{\tau_k})^{\rho_{\ell,k}}\, D_{S_k}^{u_\ell}\, W_{23}^{v_\ell b_k} \qquad (\ell \notin C).$$

The $\mathbb{H}_1$-exponents of this key are those of a normal key, and its $\mathbb{H}_2$-exponents are $v_\ell \eta a_k$ and $v_\ell(\alpha' + \eta b_k)$. Conditioned on the single event $\eta \not\equiv 0 \bmod p_2$, the values $y_k = \eta a_k$ and $c_k = \eta b_k$ are fresh and independent across all keys, and $\alpha'$ is shared by all keys. Thus the key distribution is exactly that of $\mathsf{G}_{\alpha'}$.

For the challenge ciphertext, the reduction outputs

$$C_0 = g_1^s Y_2, \quad C_1 = (g_1^s Y_2)^{u + W_{X^*}}, \quad C_2 = (g_1^s Y_2)^{\beta + \gamma\tau^*}, \quad C_3 = T \cdot M_b.$$

Writing $Y_2 = g_2^x$, the $\mathbb{G}_2$-exponents of $C_1, C_2$ are $x(u + W_{X^*})$ and $x(\beta + \gamma\tau^*)$. By the Chinese Remainder Theorem, the residues of $u, \beta, \gamma, w_1, \ldots, w_n$ modulo $p_2$ are independent of their residues modulo $p_1$ and $p_3$, which determine the public key, the corrupted shares, and the $\mathbb{H}_1$-parts of the keys, so $\mu = u + W_{X^*}$ and $z^* = \beta + \gamma\tau^*$ are uniform and independent. Hence the ciphertext is distributed as in $\mathsf{G}_{\alpha'}$.

If $T = e(g_1, h_1)^{\alpha s}$, the simulation is exactly $\mathsf{G}_{\alpha'}$. If $T$ is uniform in $\mathbb{G}_T$, then $C_3$ is uniform and independent of $b$, which is $\mathsf{G}_{\text{final}}$. The excluded event $\eta \equiv 0 \bmod p_2$ has probability $1/p_2$.

**Lemma 5.9.** $\mathsf{Adv}^{\mathsf{G}_{\text{final}}}_{\mathcal{A}} = 0$.

*Proof.* In $\mathsf{G}_{\text{final}}$, the challenge component $C_3$ is uniform in $\mathbb{G}_T$ and independent of $b$, and no other value given to $\mathcal{A}$ and no abort decision depends on $b$. Hence $\mathcal{A}$ outputs $b$ with probability exactly $1/2$, and an abort also yields a uniform bit.

*Proof of Theorem 5.2.* As

$$\mathsf{Adv}^{\mathsf{ThSUB},\mathsf{st}}_{\Theta,\mathcal{A}}(\lambda, n, L, \theta) = 2\mathsf{Adv}^{\mathsf{G}_{\mathrm{real}}}_{\mathcal{A}} = 2n(Q+1)\mathsf{Adv}^{\mathsf{G}_{\mathrm{guess}}}_{\mathcal{A}}.$$

Combining the hybrids and using $p_2 \geq 2^{\lambda}$ , we obtain

$$\left|\Pr[\mathsf{G}_{\mathrm{guess}} = 1] - \Pr[\mathsf{G}_{\mathrm{final}} = 1]\right| \leq \Delta.$$

Since $\Pr[\mathsf{G}_{\mathrm{final}} = 1] = \frac{1}{2}$ by Lemma 5.9, this gives $\mathsf{Adv}^{\mathsf{G}_{\mathrm{guess}}}_{\mathcal{A}} \leq \Delta$, which yields (6).

*Remark* 5.1. The polynomial $v_C$ depends on the corruption set selected by the adversary, and the reduction uses it to generate the public keys by (9), which does not require $\alpha$ or $u$. This is why the proof requires the adversary to fix its corruption set before setup, and therefore establishes adaptive challenge security only under static corruption.

**Exponential-size Identity Space** The scheme $\Theta$ of Section 5.1 supports a polynomial-size universe $[n]$. We now apply the cover-free compiler of Section 4 to obtain a threshold batched IBE scheme for the identity space $\mathcal{I} := \{0,1\}^{\lambda}$.

**Theorem 5.3.** *Let $B, L, Q \geq 1$ be polynomially bounded, let $(n, L, \theta)$ be the threshold parameters, and let $\Pi$ be the scheme of construction 4.1 instantiated with $\Theta$ and the family (4.3). Then $\Pi$ is a threshold batched IBE scheme with identity space $\{0,1\}^{\lambda}$ and batch bound $B$. If the $\mathsf{SD}_G$, $\mathsf{SD}_H$, and $\mathsf{PAIR}$ assumptions hold, then $\Pi$ has adaptive challenge security under static corruption. That is, for every admissible PPT adversary $\mathcal{A}$ in the static-corruption experiment of Definition 3.3 that makes at most $Q$ key queries,*

$$\mathsf{Adv}^{\mathsf{ThBIBE},\mathsf{st}}_{\Pi,\mathcal{A}}(\lambda, B, L, \theta) \leq 2q^2(Q+1)\,\Delta,$$

*where $\Delta$ is defined in (7) for the algorithms given by Theorem 5.2 with $n = q^2$.*

*Proof.* Completeness and correctness follow from Theorem 5.1 and part (a) of Theorem 4.1. For security, part (b) of Theorem 4.1 with $\chi = \mathsf{st}$ turns $\mathcal{A}$ into an admissible adversary against $\Theta$ in the static-corruption model, which forwards the corruption set of $\mathcal{A}$ before setup. The resulting adversary makes the same number of key queries and has the same advantage. The bound then follows from Theorem 5.2 with $n = q^2$. □

The compiler does not change public keys, ciphertexts, digests, or keys.

**Efficiency.** Compared to $\Theta$ with $n = q^2$, the compiled scheme additionally computes the blocks $C_{\mathsf{id}}$. Encryption computes $C_{\mathsf{id}}$, which takes $O(q\lambda)$ operations in $\mathbb{F}_q$, and $\mathsf{Digest}$ and $\mathsf{Decrypt}$ compute the union $S_\Gamma$ of at most $B$ blocks, which takes $O(Bq\lambda)$ operations in $\mathbb{F}_q$. $\mathsf{CompKeyShare}$ and $\mathsf{CompKeyAggregate}$ are those of $\Theta$. In encryption, the element $L_{C_{\mathsf{id}}} := g_1^u \prod_{i \in C_{\mathsf{id}}} g_1^{w_i}$ takes $O(q)$ group operations, but it depends only on the identity, so it can be computed offline and reused, as described in Section 4.3. Given $L_{C_{\mathsf{id}}}$, online encryption computes five exponentiations, independently of $B$.

# 6 Batched IBE from MDDH in Prime-Order Groups

In this section, we give a prime-order counterpart of the construction of Section 5 for the non-threshold case $L = \theta = 1$. We first construct a subset-policy batched IBE scheme in the sense of Remark 3.1, with message space $\mathbb{G}_T$ and tag space $\mathbb{Z}_p$, and prove its adaptive security under the $\mathcal{U}_k$-MDDH assumption (Assumption 3.4) in both source groups. Applying the compiler of Section 4 then gives a batched IBE scheme with an exponential identity space (Section 6.2).

**Notation.** Let $\mathsf{pg} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^\lambda)$ be an asymmetric prime-order bilinear group, where $p$ is a $\Theta(\lambda)$-bit prime, $g$ and $h$ generate $\mathbb{G}$ and $\mathbb{H}$, and $e$ is nondegenerate. We write all three groups additively. For a matrix $\boldsymbol{X}$ over $\mathbb{Z}_p$, we write $[\![\boldsymbol{X}]\!]_1$, $[\![\boldsymbol{X}]\!]_2$, and $[\![\boldsymbol{X}]\!]_\mathsf{T}$ for its entry-wise encoding in $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$, and for compatibly shaped $\boldsymbol{X}$ and $\boldsymbol{Y}$ we let

$$e\big([\![\boldsymbol{X}]\!]_1, [\![\boldsymbol{Y}]\!]_2\big) := [\![\boldsymbol{X}\boldsymbol{Y}]\!]_\mathsf{T}. \tag{16}$$

Given $[\![\boldsymbol{X}]\!]_1$ and a matrix $\boldsymbol{Y}$ over $\mathbb{Z}_p$, one can compute $[\![\boldsymbol{X}\boldsymbol{Y}]\!]_1$ and $[\![\boldsymbol{Y}\boldsymbol{X}]\!]_1$, and similarly in $\mathbb{H}$ and $\mathbb{G}_T$.

## 6.1 Construction

Fix $n \geq 1$ and $m = k + 1$. We use the following algorithms to define $\Pi_\subseteq$. We write $\boldsymbol{Z}_\tau := \boldsymbol{Z}_0 + \tau \boldsymbol{Z}_1$ and $\boldsymbol{W}_X := \sum_{i \in X} \boldsymbol{W}_i$.

$\underline{\mathsf{Setup}_\subseteq(1^\lambda, 1^n) \to \mathsf{pp}_\subseteq}$.

Run $\mathsf{pg} \leftarrow \mathcal{G}(1^\lambda)$ and output $\mathsf{pp}_\subseteq := (\mathsf{pg}, n, k)$.

$\underline{\mathsf{KeyGen}_\subseteq(\mathsf{pp}_\subseteq) \to \mathsf{mpk}_\subseteq, \mathsf{msk}_\subseteq}$. Sample

$$\boldsymbol{A}, \boldsymbol{B} \leftarrow\!\!\$\, \mathcal{U}_{m,k}, \qquad \boldsymbol{\alpha} \leftarrow\!\!\$\, \mathbb{Z}_p^m, \qquad \boldsymbol{U}, \boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_n \leftarrow\!\!\$\, \mathbb{Z}_p^{m \times m}.$$

For each $j \in [n]$, independently sample $\boldsymbol{r}_j \leftarrow\!\!\$\, \mathbb{Z}_p^k$ and set $\boldsymbol{d}_j := \boldsymbol{B}\boldsymbol{r}_j$. The public key contains $\mathsf{pp}_\subseteq$ and the following objects:

$$\begin{aligned}
&\boldsymbol{A}_0 := [\![\boldsymbol{A}^\mathsf{T}]\!]_1, && \overline{\boldsymbol{U}} := [\![\boldsymbol{A}^\mathsf{T}\boldsymbol{U}]\!]_1,\\
&\overline{\boldsymbol{Z}}_b := [\![\boldsymbol{A}^\mathsf{T}\boldsymbol{Z}_b]\!]_1 \quad (b \in \{0,1\}), && \overline{\boldsymbol{W}}_i := [\![\boldsymbol{A}^\mathsf{T}\boldsymbol{W}_i]\!]_1 \quad (i \in [n]),\\
&\boldsymbol{P}_j := [\![\boldsymbol{d}_j]\!]_2 \quad (j \in [n]), && \boldsymbol{Q}_{i,j} := [\![\boldsymbol{W}_i\boldsymbol{d}_j]\!]_2 \quad (i \neq j),\\
&\boldsymbol{\Omega} := [\![\boldsymbol{A}^\mathsf{T}\boldsymbol{\alpha}]\!]_\mathsf{T}.
\end{aligned} \tag{17}$$

$$\begin{aligned}
\mathsf{pk}_\subseteq := \Big(&\mathsf{pp}_\subseteq,\ [\![\boldsymbol{A}^\top]\!]_1,\ [\![\boldsymbol{A}^\top\boldsymbol{U}]\!]_1,\ [\![\boldsymbol{A}^\top\boldsymbol{Z}_0]\!]_1,\ [\![\boldsymbol{A}^\top\boldsymbol{Z}_1]\!]_1,\ \big\{[\![\boldsymbol{A}^\top\boldsymbol{W}_i]\!]_1\big\}_{i\in[n]},\\
&\big\{[\![\boldsymbol{d}_j]\!]_2\big\}_{j\in[n]},\ \big\{[\![\boldsymbol{W}_i\boldsymbol{d}_j]\!]_2\big\}_{i\neq j},\ [\![\boldsymbol{A}^\top\boldsymbol{\alpha}]\!]_\mathsf{T}\Big).
\end{aligned}$$

The master secret is $\mathsf{msk}_\subseteq := (\boldsymbol{B}, \boldsymbol{\alpha}, \boldsymbol{U}, \boldsymbol{Z}_0, \boldsymbol{Z}_1)$.

$\underline{\mathsf{Encrypt}_{\subseteq}(\mathsf{pk}_{\subseteq}, M, X, \tau) \to \mathsf{ct}.}$ Sample $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^k$ and compute

$$\begin{aligned} \boldsymbol{C}_0 &:= \boldsymbol{s}^\mathsf{T}\boldsymbol{A}_0, & \boldsymbol{C}_1 &:= \boldsymbol{s}^\mathsf{T}\Big(\overline{\boldsymbol{U}} + \sum_{i\in X}\overline{\boldsymbol{W}}_i\Big), \\ \boldsymbol{C}_2 &:= \boldsymbol{s}^\mathsf{T}(\overline{\boldsymbol{Z}}_0 + \tau\overline{\boldsymbol{Z}}_1), & C_3 &:= M + \boldsymbol{s}^\mathsf{T}\boldsymbol{\Omega}. \end{aligned} \tag{18}$$

Output $\mathsf{ct} := (\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, C_3)$.

$\underline{\mathsf{Digest}_{\subseteq}(\mathsf{pk}_{\subseteq}, S) \to \boldsymbol{D}_S.}$ For $S \subseteq [n]$, output

$$\boldsymbol{D}_S := \sum_{j\notin S}\boldsymbol{P}_j = [\![\boldsymbol{d}_S]\!]_2, \qquad \boldsymbol{d}_S := \sum_{j\notin S}\boldsymbol{d}_j. \tag{19}$$

$\underline{\mathsf{ComputeKey}_{\subseteq}(\mathsf{msk}_{\subseteq}, \boldsymbol{D}, \tau) \to \mathsf{sk}_{\tau,\boldsymbol{D}}.}$ Sample $\boldsymbol{r}_0 \leftarrow\!\!\$\ \mathbb{Z}_p^k$, set $\boldsymbol{d}_0 := \boldsymbol{B}\boldsymbol{r}_0$, and compute

$$\boldsymbol{K}_0 := [\![\boldsymbol{d}_0]\!]_2, \qquad \boldsymbol{K}_1 := [\![\boldsymbol{\alpha} + \boldsymbol{Z}_\tau\boldsymbol{d}_0]\!]_2 + \boldsymbol{U}\boldsymbol{D}. \tag{20}$$

Output $\mathsf{sk}_{\tau,\boldsymbol{D}} := (\boldsymbol{K}_0, \boldsymbol{K}_1)$.

$\underline{\mathsf{Decrypt}_{\subseteq}(\mathsf{pk}_{\subseteq}, \mathsf{sk}, \boldsymbol{D}, S, (X, \tau), \mathsf{ct}) \to M.}$ Parse the key and ciphertext as

$$\mathsf{sk} = (\tau', \boldsymbol{K}_0, \boldsymbol{K}_1), \qquad \mathsf{ct} = (X_\mathsf{ct}, \tau_\mathsf{ct}, \boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, C_3).$$

Recompute $\boldsymbol{D}_S$ and require $\boldsymbol{D} = \boldsymbol{D}_S$. Compute the vector

$$\boldsymbol{E}_{X,S} := \sum_{i\in X}\sum_{j\notin S}\boldsymbol{Q}_{i,j} = [\![\boldsymbol{W}_X\boldsymbol{d}_S]\!]_2, \tag{21}$$

and output $M = C_3 - T$, where

$$T := e(\boldsymbol{C}_0, \boldsymbol{K}_1) - e(\boldsymbol{C}_1, \boldsymbol{D}_S) + e(\boldsymbol{C}_0, \boldsymbol{E}_{X,S}) - e(\boldsymbol{C}_2, \boldsymbol{K}_0) = [\![\boldsymbol{s}^\top\boldsymbol{A}^\top\boldsymbol{\alpha}]\!]_\mathsf{T}. \tag{22}$$

**Theorem 6.1.** *The construction satisfies perfect correctness.*

*Proof.* For an valid ciphertext with $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s}$, the four pairings in (22) give

$$\begin{aligned} e(\boldsymbol{C}_0, \boldsymbol{K}_1) &= [\![\boldsymbol{c}^\mathsf{T}\boldsymbol{\alpha} + \boldsymbol{c}^\mathsf{T}\boldsymbol{Z}_\tau\boldsymbol{d}_0 + \boldsymbol{c}^\mathsf{T}\boldsymbol{U}\boldsymbol{d}_S]\!]_\mathsf{T}, \\ e(\boldsymbol{C}_1, \boldsymbol{D}_S) &= [\![\boldsymbol{c}^\mathsf{T}\boldsymbol{U}\boldsymbol{d}_S + \boldsymbol{c}^\mathsf{T}\boldsymbol{W}_X\boldsymbol{d}_S]\!]_\mathsf{T}, \\ e(\boldsymbol{C}_0, \boldsymbol{E}_{X,S}) &= [\![\boldsymbol{c}^\mathsf{T}\boldsymbol{W}_X\boldsymbol{d}_S]\!]_\mathsf{T}, \\ e(\boldsymbol{C}_2, \boldsymbol{K}_0) &= [\![\boldsymbol{c}^\mathsf{T}\boldsymbol{Z}_\tau\boldsymbol{d}_0]\!]_\mathsf{T}. \end{aligned}$$

Clearly, $T$ using Equation (22) leave $T = [\![\boldsymbol{c}^\mathsf{T}\boldsymbol{\alpha}]\!]_\mathsf{T}$. Consequently, $C_3 - T = M$.

□

**Sizes and efficiency.** Excluding the group description and the metadata, the numbers of group elements are

$$\begin{aligned} \mathsf{pk}_{\subseteq}: &\quad (n+4)k(k+1) \text{ in } \mathbb{G}, \quad n^2(k+1) \text{ in } \mathbb{H}, \quad k \text{ in } \mathbb{G}_T, \\ \mathsf{ct}: &\quad 3(k+1) \text{ in } \mathbb{G}, \quad 1 \text{ in } \mathbb{G}_T, \qquad \boldsymbol{D}_S: \quad k+1 \text{ in } \mathbb{H}, \qquad \mathsf{sk}: \quad 2(k+1) \text{ in } \mathbb{H}. \end{aligned} \tag{23}$$

For $k = 1$, a ciphertext has six elements of $\mathbb{G}$ and one of $\mathbb{G}_T$, a digest has two elements of $\mathbb{H}$, and a key has four elements of $\mathbb{H}$. Encryption takes $O(k^2|X|)$ group operations, digest computation $O(k(n - |S|))$, and key computation $O(k^2)$. Decryption takes $4(k+1)$ pairings for (22), and $O(k|X|(n - |S|))$ group operations.

## 6.2 Security Analysis

We use the experiment $\mathsf{Exp}^{\mathsf{SUB}}_{\Pi_\subseteq,\mathcal{A}}(\lambda, n, b)$ of Remark 3.1 where every key query $(S_\ell, \tau_\ell)$ is answered with a key for the honest digest $\boldsymbol{D}_{S_\ell}$, at most one key query has the challenge tag $\tau^*$, and if this query is $(S^*, \tau^*)$, then $X^* \not\subseteq S^*$.

**Theorem 6.2.** *Let $n, Q$ be polynomially bounded and let $k \geq 1$ be fixed. For every admissible PPT adversary $\mathcal{A}$ that makes at most $Q$ key queries, there exist PPT adversaries $\mathcal{B}_0$, $\mathcal{B}_1, \ldots, \mathcal{B}_Q$, and $\mathcal{B}_{\mathsf{col}}$ such that*

$$\mathsf{Adv}^{\mathsf{SUB}}_{\Pi_\subseteq,\mathcal{A}}(\lambda, n) \leq 2n(Q+1)\,\Delta, \tag{24}$$

*where*

$$\Delta := \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,1,\mathcal{B}_0}(\lambda) + \sum_{t=1}^{Q} \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,2,\mathcal{B}_t}(\lambda) + \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,2,\mathcal{B}_{\mathsf{col}}}(\lambda) + \frac{Q+2}{p}. \tag{25}$$

*In particular, if the $\mathcal{U}_k$-MDDH assumption holds in $\mathbb{G}$ and in $\mathbb{H}$, then $\Pi_\subseteq$ is adaptively secure.*

**Basis.** For $\boldsymbol{A}, \boldsymbol{B} \leftarrow\!\!\$\, \mathcal{U}_k$, fix nonzero vectors $\boldsymbol{a}^\perp, \boldsymbol{b}^\perp \in \mathbb{Z}_p^{k+1}$, such that

$$\boldsymbol{A}^\top \boldsymbol{a}^\perp = \boldsymbol{0}, \qquad \boldsymbol{B}^\top \boldsymbol{b}^\perp = \boldsymbol{0}, \qquad \text{and let} \qquad \boldsymbol{V} := \boldsymbol{a}^\perp (\boldsymbol{b}^\perp)^\top \in \mathbb{Z}_p^{(k+1)\times(k+1)}.$$

As in [5], $\boldsymbol{a}^\perp$ spans the semi-functional space for keys.

**Parameter hiding.** The statistical steps use the parameter-hiding lemma of [5], which is the prime-order counterpart of the Chinese Remainder Theorem in Section 5 [6, Lemma 5].

**Lemma 6.1** ([5, Lemma 4])**.** *For every $\boldsymbol{M} \in \mathbb{Z}_p^{(k+1)\times(k+1)}$, $\hat{u} \in \mathbb{Z}_p$, $\boldsymbol{r} \in \mathbb{Z}_p^k$, and $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{Z}_p^{k+1}$,*

$$\begin{aligned} \boldsymbol{A}^\top(\boldsymbol{M} + \hat{u}\boldsymbol{V}) &= \boldsymbol{A}^\top \boldsymbol{M}, & (\boldsymbol{M} + \hat{u}\boldsymbol{V})\boldsymbol{B}\boldsymbol{r} &= \boldsymbol{M}\boldsymbol{B}\boldsymbol{r}, \\ \boldsymbol{c}^\top(\boldsymbol{M} + \hat{u}\boldsymbol{V}) &= \boldsymbol{c}^\top \boldsymbol{M} + \hat{u}\,(\boldsymbol{c}^\top \boldsymbol{a}^\perp)(\boldsymbol{b}^\perp)^\top, & (\boldsymbol{M} + \hat{u}\boldsymbol{V})\boldsymbol{d} &= \boldsymbol{M}\boldsymbol{d} + \hat{u}\,\big((\boldsymbol{b}^\perp)^\top \boldsymbol{d}\big)\boldsymbol{a}^\perp. \end{aligned}$$

*In particular, if $\boldsymbol{M} \leftarrow\!\!\$\, \mathbb{Z}_p^{(k+1)\times(k+1)}$ and $\hat{u} \leftarrow\!\!\$\, \mathbb{Z}_p$ are independent, then $\boldsymbol{M} + \hat{u}\boldsymbol{V}$ is uniform and independent of $\hat{u}$.*

*Proof.* The identities follow from $\boldsymbol{A}^\top \boldsymbol{a}^\perp = \boldsymbol{0}$ and $(\boldsymbol{b}^\perp)^\top \boldsymbol{B} = \boldsymbol{0}$, and the last claim holds since $\boldsymbol{M} \mapsto \boldsymbol{M} + \hat{u}\boldsymbol{V}$ is a bijection for every $\hat{u}$. □

Thus, in a game that samples a secret matrix $\boldsymbol{M}$ uniformly, we may equivalently sample $\boldsymbol{M}$ and $\hat{u}$ uniformly and use $\boldsymbol{M} + \hat{u}\boldsymbol{V}$ in place of $\boldsymbol{M}$. The public key and every value computed from normal vectors then do not depend on $\hat{u}$.
We define the following distributions, following [5, 13]. Let $(M_0, M_1, X^*, \tau^*)$ be the challenge query, and let $(S_\ell, \tau_\ell)$ be the $\ell$-th key query.

**Ciphertexts.** For $\boldsymbol{c} \in \mathbb{Z}_p^{k+1}$, let

$$\mathsf{CT}_b(\boldsymbol{c}) := \Big(X^*, \tau^*,\ [\![\boldsymbol{c}^\top]\!]_1,\ [\![\boldsymbol{c}^\top(\boldsymbol{U} + \boldsymbol{W}_{X^*})]\!]_1,\ [\![\boldsymbol{c}^\top \boldsymbol{Z}_{\tau^*}]\!]_1,\ [\![\boldsymbol{c}^\top \boldsymbol{\alpha}]\!]_\mathsf{T} + M_b\Big).$$

The challenge ciphertext is *normal* if $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s}$ with $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^k$, and *semi-functional* if $\boldsymbol{c} \leftarrow\!\!\$\ \mathbb{Z}_p^{k+1}$.

**Keys.** For $\boldsymbol{d} \in \mathbb{Z}_p^{k+1}$ and $\gamma \in \mathbb{Z}_p$, let

$$\mathsf{K}_\ell(\boldsymbol{d}, \gamma) := \Big(\tau_\ell,\ [\![\boldsymbol{d}]\!]_2,\ [\![\boldsymbol{\alpha} + \boldsymbol{Z}_{\tau_\ell}\boldsymbol{d} + \boldsymbol{U}\boldsymbol{d}_{S_\ell} + \gamma\, \boldsymbol{a}^\perp]\!]_2\Big),$$

where $\boldsymbol{d}_{S_\ell}$ is the exponent of the digest $\boldsymbol{D}_{S_\ell}$ computed from the public key. With fresh $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^k$, $\boldsymbol{d} \leftarrow\!\!\$\ \mathbb{Z}_p^{k+1}$, and $\gamma \leftarrow\!\!\$\ \mathbb{Z}_p$ for every key, we call

$$\mathsf{K}_\ell(\boldsymbol{B}\boldsymbol{r}, 0), \qquad \mathsf{K}_\ell(\boldsymbol{d}, 0), \qquad \mathsf{K}_\ell(\boldsymbol{d}, \gamma), \qquad \mathsf{K}_\ell(\boldsymbol{B}\boldsymbol{r}, \gamma)$$

a *normal*, *pseudo-normal*, *pseudo-SF*, and *semi-functional* key, respectively, as in [5]. Since $\boldsymbol{A}^\top \boldsymbol{a}^\perp = \boldsymbol{0}$, all of them decrypt normal ciphertexts.

**Public keys.** For $\widehat{\imath} \in [n]$, a *type-$\widehat{\imath}$* public key is generated as in Section 6.1, except that $\boldsymbol{d}_{\widehat{\imath}} \leftarrow\!\!\$\ \mathbb{Z}_p^{k+1}$. The values $[\![\boldsymbol{d}_{\widehat{\imath}}]\!]_2$ and $[\![\boldsymbol{W}_a \boldsymbol{d}_{\widehat{\imath}}]\!]_2$ for $a \neq \widehat{\imath}$ are computed from this vector, the diagonal $[\![\boldsymbol{W}_{\widehat{\imath}} \boldsymbol{d}_{\widehat{\imath}}]\!]_2$ is not published, and the digest of a set $S$ has exponent

$$\boldsymbol{d}_S = \sum_{j \notin S,\ j \neq \widehat{\imath}} \boldsymbol{B}\boldsymbol{r}_j + \boldsymbol{1}[\widehat{\imath} \notin S]\, \boldsymbol{d}_{\widehat{\imath}}. \tag{26}$$

**Game sequence.** We consider the following sequence of games.

$$\mathsf{Game}_{\mathsf{real}} \to \mathsf{Game}_{\mathsf{guess}} \to \mathsf{Game}_0 \to \big(\mathsf{Game}_{t,1} \to \mathsf{Game}_{t,2}\big)_{t=1,\ldots,Q} \to \mathsf{Game}_{\mathsf{col},1} \to \mathsf{Game}_{\mathsf{col},2}.$$

They follow the games of Section 5.2, except that the modular restriction is not needed since tags are elements of $\mathbb{Z}_p$.

**$\mathsf{Game}_{\mathsf{real}}$.** This is the experiment of Remark 3.1.

$\mathsf{Game}_{\mathsf{guess}}$. This is the same as $\mathsf{Game}_{\mathsf{real}}$, except that the challenger samples

$$\widehat{\imath} \leftarrow\!\$\ [n], \qquad \widehat{\jmath} \leftarrow\!\$\ [Q] \cup \{\bot\}$$

at the beginning and aborts as described above. The query response and the challenge ciphertext are computed as in $\mathsf{Game}_{\mathsf{real}}$.

$\mathsf{Game}_0$. This is the same as $\mathsf{Game}_{\mathsf{guess}}$, except that the challenge ciphertext is semi-functional:

$$\mathsf{ct}^* := \mathsf{CT}_b(\boldsymbol{c}), \qquad \boldsymbol{c} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}.$$

$\mathsf{Game}_{t,1}$ **and** $\mathsf{Game}_{t,2}$ **for** $t \in [Q]$. These games switch the non-special keys one by one, first to pseudo-normal and then to pseudo-SF. $\mathsf{Game}_{t,1}$ is the same as $\mathsf{Game}_{t-1,2}$, except that if the $t$-th key is non-special, its randomness is uniform:

$$\mathsf{sk}_t := \mathsf{K}_t(\boldsymbol{d}_t, 0), \qquad \boldsymbol{d}_t \leftarrow\!\$\ \mathbb{Z}_p^{k+1}.$$

$\mathsf{Game}_{t,2}$ is the same as $\mathsf{Game}_{t,1}$, except that if the $t$-th key is non-special, it also has a semi-functional component:

$$\mathsf{sk}_t := \mathsf{K}_t(\boldsymbol{d}_t, \gamma_t), \qquad \boldsymbol{d}_t \leftarrow\!\$\ \mathbb{Z}_p^{k+1},\ \gamma_t \leftarrow\!\$\ \mathbb{Z}_p.$$

Thus, in $\mathsf{Game}_{t,2}$, the $\ell$-th key is

$$\mathsf{sk}_\ell = \begin{cases} \mathsf{K}_\ell(\boldsymbol{d}_\ell, \gamma_\ell) & \text{if } \ell \le t \text{ and } \ell \ne \widehat{\jmath}, \\ \mathsf{K}_\ell(\boldsymbol{B}\boldsymbol{r}_{0,\ell}, 0) & \text{if } \ell > t \text{ or } \ell = \widehat{\jmath}, \end{cases}$$

with fresh $\boldsymbol{d}_\ell \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$, $\gamma_\ell \leftarrow\!\$\ \mathbb{Z}_p$, and $\boldsymbol{r}_{0,\ell} \leftarrow\!\$\ \mathbb{Z}_p^k$ for every key. In $\mathsf{Game}_{Q,2}$, every non-special key is pseudo-SF, and the special key is normal.

$\mathsf{Game}_{\mathsf{col},1}$. This is the same as $\mathsf{Game}_{Q,2}$, except that the public key is of type $\widehat{\imath}$, i.e., $\boldsymbol{d}_{\widehat{\imath}} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$. Accordingly, the digest of every key query $(S_\ell, \tau_\ell)$ is computed from this public key and has the exponent $\boldsymbol{d}_{S_\ell}$ of (26). The keys have the same form as in $\mathsf{Game}_{Q,2}$.

$\mathsf{Game}_{\mathsf{col},2}$. This is the same as $\mathsf{Game}_{\mathsf{col},1}$, except that the special key, if it is queried, is semi-functional:

$$\mathsf{sk}_{\widehat{\jmath}} := \mathsf{K}_{\widehat{\jmath}}(\boldsymbol{B}\boldsymbol{r}_{0,\widehat{\jmath}}, \gamma_{\widehat{\jmath}}), \qquad \boldsymbol{r}_{0,\widehat{\jmath}} \leftarrow\!\$\ \mathbb{Z}_p^k,\ \gamma_{\widehat{\jmath}} \leftarrow\!\$\ \mathbb{Z}_p.$$

Thus, every key now has an independent uniform semi-functional component.

**Lemma 6.2.** *For any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{guess},\mathcal{A}} = \dfrac{1}{n(Q+1)}\mathsf{Adv}_{\mathsf{real},\mathcal{A}}$.

*Proof.* The guesses are independent of the view of $\mathcal{A}$, and they are correct with probability $1/(n(Q+1))$. If they are correct, $\mathsf{Game}_{\mathsf{guess}}$ is identical to $\mathsf{Game}_{\mathsf{real}}$, and otherwise it outputs a uniform bit.

**Lemma 6.3.** *For a PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}_0$ such that*

$$|\mathsf{Adv}_{\mathsf{guess},\mathcal{A}} - \mathsf{Adv}_{0,\mathcal{A}}| \leq \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,1,\mathcal{B}_0}(\lambda).$$

*Proof.* Given a challenge instance $(\mathsf{pg}, [\![\boldsymbol{A}]\!]_1, [\![\boldsymbol{v}]\!]_1)$ for the $\mathcal{U}_k$-MDDH assumption in $\mathbb{G}$, $\mathcal{B}_0$ picks $b \leftarrow\!\$\ \{0,1\}$, $(\widehat{\imath},\widehat{\jmath})$, $\boldsymbol{B}$, $\boldsymbol{\alpha}$, $\boldsymbol{U}, \boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$, and $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$, sets $[\![\boldsymbol{c}]\!]_1 := [\![\boldsymbol{v}]\!]_1$, and simulates the public key, the keys, and the challenge ciphertext as in $\mathsf{Game}_{\mathsf{guess}}$, using $[\![\boldsymbol{c}]\!]_1$ for the challenge ciphertext. It computes $[\![\boldsymbol{A}^\top\boldsymbol{\alpha}]\!]_\mathsf{T} = e([\![\boldsymbol{A}^\top]\!]_1, [\![\boldsymbol{\alpha}]\!]_2)$ and $[\![\boldsymbol{c}^\top\boldsymbol{\alpha}]\!]_\mathsf{T} = e([\![\boldsymbol{c}^\top]\!]_1, [\![\boldsymbol{\alpha}]\!]_2)$, and it computes all other values from the known exponents. When $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{w}$ for $\boldsymbol{w} \leftarrow\!\$\ \mathbb{Z}_p^k$, $\mathcal{B}_0$ simulates $\mathsf{Game}_{\mathsf{guess}}$, and when $\boldsymbol{v} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$, it simulates $\mathsf{Game}_0$.

**Lemma 6.4.** *For all $t \in [Q]$, there exists an adversary $\mathcal{B}_t$ such that*

$$|\mathsf{Adv}_{t-1,2} - \mathsf{Adv}_{t,1}| \leq \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,2,\mathcal{B}_t}(\lambda).$$

*Proof.* Upon receiving a challenge $(\mathsf{pg}, [\![\boldsymbol{B}]\!]_2, [\![\boldsymbol{v}]\!]_2)$ for the $\mathcal{U}_k$-MDDH assumption in $\mathbb{H}$, $\mathcal{B}_t$ picks $b \leftarrow\!\$\ \{0,1\}$, $(\widehat{\imath},\widehat{\jmath})$, $\boldsymbol{A}$, $\boldsymbol{\alpha}$, $\boldsymbol{U}, \boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$, and $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$, and computes $\boldsymbol{a}^\perp$ from $\boldsymbol{A}$. It sets $[\![\boldsymbol{d}_j]\!]_2 := [\![\boldsymbol{B}]\!]_2\boldsymbol{r}_j$ and $[\![\boldsymbol{W}_i\boldsymbol{d}_j]\!]_2 := \boldsymbol{W}_i[\![\boldsymbol{d}_j]\!]_2$, and it picks $\boldsymbol{c} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$ and outputs the semi-functional challenge $\mathsf{CT}_b(\boldsymbol{c})$. On the $\ell$-th key query, it outputs the following.

- If $\ell < t$ and $\ell \neq \widehat{\jmath}$: the pseudo-SF key $\mathsf{K}_\ell(\boldsymbol{d}, \gamma)$ with its own $\boldsymbol{d} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$ and $\gamma \leftarrow\!\$\ \mathbb{Z}_p$.
- If $\ell = t \neq \widehat{\jmath}$: $\big(\tau_t,\ [\![\boldsymbol{v}]\!]_2,\ [\![\boldsymbol{\alpha}]\!]_2 + \boldsymbol{Z}_{\tau_t}[\![\boldsymbol{v}]\!]_2 + \boldsymbol{U}\boldsymbol{D}_{S_t}\big)$.
- Otherwise: $\big(\tau_\ell,\ \boldsymbol{K}_0,\ [\![\boldsymbol{\alpha}]\!]_2 + \boldsymbol{Z}_{\tau_\ell}\boldsymbol{K}_0 + \boldsymbol{U}\boldsymbol{D}_{S_\ell}\big)$ with $\boldsymbol{K}_0 := [\![\boldsymbol{B}]\!]_2\boldsymbol{r}$ and $\boldsymbol{r} \leftarrow\!\$\ \mathbb{Z}_p^k$.

When $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{w}$ for $\boldsymbol{w} \leftarrow\!\$\ \mathbb{Z}_p^k$, the $t$-th key is normal and $\mathcal{B}_t$ simulates $\mathsf{Game}_{t-1,2}$, and when $\boldsymbol{v} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$, the $t$-th key is pseudo-normal and $\mathcal{B}_t$ simulates $\mathsf{Game}_{t,1}$. If the $t$-th key is special or is never queried, the two games are identical.

**Lemma 6.5.** *For all $t \in [Q]$,*

$$|\mathsf{Adv}_{t,1} - \mathsf{Adv}_{t,2}| \leq \frac{1}{p}.$$

*Proof.* The two games differ only if the $t$-th key is non-special and queried, in which case its tag satisfies $\tau_t \neq \tau^*$. Let $\boldsymbol{d}$ be the randomness of this key, and let $\rho := (\boldsymbol{b}^\perp)^\top\boldsymbol{d}$, which is 0 with probability $1/p$ in both games. It suffices to show that the two games are identically distributed if $\rho \neq 0$.

By parameter hiding (Lemma 6.1), we may replace $(\boldsymbol{Z}_0, \boldsymbol{Z}_1)$ by $(\boldsymbol{Z}_0 + \hat{u}_0\boldsymbol{V}, \boldsymbol{Z}_1 + \hat{u}_1\boldsymbol{V})$ with $\hat{u}_0, \hat{u}_1 \leftarrow\!\$\ \mathbb{Z}_p$ in both games. Write $u_\tau := \hat{u}_0 + \tau\hat{u}_1$, so that $\boldsymbol{Z}_\tau$ is replaced by $\boldsymbol{Z}_\tau + u_\tau\boldsymbol{V}$. This does not change the public key, the normal keys, or the special key. It adds $u_{\tau^*}(\boldsymbol{c}^\top\boldsymbol{a}^\perp)[\![(\boldsymbol{b}^\perp)^\top]\!]_1$ to the third component of the challenge ciphertext, and it adds $u_{\tau_\ell}((\boldsymbol{b}^\perp)^\top\boldsymbol{d}_\ell)\boldsymbol{a}^\perp$ to the exponent of each pseudo-SF key $\ell < t$ with randomness $\boldsymbol{d}_\ell$, which is absorbed by its fresh $\gamma_\ell$. Finally, it adds $\rho u_{\tau_t}\boldsymbol{a}^\perp$ to the $t$-th key. In $\mathsf{Game}_{t,2}$, this key additionally has a fresh $\gamma_t$, so its semi-functional component is uniform and independent of $(\hat{u}_0, \hat{u}_1)$. In $\mathsf{Game}_{t,1}$, its semi-functional component is $\rho u_{\tau_t}$, and $(\hat{u}_0, \hat{u}_1)$ appears elsewhere only through $u_{\tau^*}$. As $\tau_t \neq \tau^*$, the map $(\hat{u}_0, \hat{u}_1) \mapsto (u_{\tau_t}, u_{\tau^*})$ is a bijection of $\mathbb{Z}_p^2$, so $u_{\tau_t}$ and $u_{\tau^*}$ are independent and uniform. Since $\rho \neq 0$, the semi-functional component of the $t$-th key is uniform and independent of the rest of the view in both games. As in [5, proof of Lemma 7], this holds even if $\mathcal{A}$ chooses $\tau^*$

after seeing the $t$-th key, or $\tau_t$ after seeing the challenge ciphertext: whichever of $u_{\tau_t}$ and $u_{\tau^*}$ is revealed first, the other one equals it plus $(\tau^* - \tau_t)\hat{u}_1$, respectively $(\tau_t - \tau^*)\hat{u}_1$, and $\hat{u}_1$ is uniform given the view at that point.

**Lemma 6.6.** *TThere exists an adversary $\mathcal{B}_{\mathsf{col}}$ such that*

$$|\mathsf{Adv}_{Q,2} - \mathsf{Adv}_{\mathsf{col},1}| \le \mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{U}_k,2,\mathcal{B}_{\mathsf{col}}}(\lambda).$$

*Proof.* Upon receiving a challenge $(\mathsf{pg}, [\![\boldsymbol{B}]\!]_2, [\![\boldsymbol{v}]\!]_2)$ for the $\mathcal{U}_k$-MDDH assumption in $\mathbb{H}$, $\mathcal{B}_{\mathsf{col}}$ picks $b$, $(\widehat{\imath}, \widehat{\jmath})$, $\boldsymbol{A}$, $\boldsymbol{\alpha}$, $\boldsymbol{U}, \boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$, and $\boldsymbol{r}_j$ for $j \neq \widehat{\imath}$, and sets

$$\begin{aligned} [\![\boldsymbol{d}_{\widehat{\imath}}]\!]_2 &:= [\![\boldsymbol{v}]\!]_2, & [\![\boldsymbol{W}_a \boldsymbol{d}_{\widehat{\imath}}]\!]_2 &:= \boldsymbol{W}_a [\![\boldsymbol{v}]\!]_2 \quad (a \neq \widehat{\imath}), \\ [\![\boldsymbol{d}_j]\!]_2 &:= [\![\boldsymbol{B}]\!]_2 \boldsymbol{r}_j, & [\![\boldsymbol{W}_i \boldsymbol{d}_j]\!]_2 &:= \boldsymbol{W}_i [\![\boldsymbol{d}_j]\!]_2 \quad (j \neq \widehat{\imath},\ i \neq j). \end{aligned}$$

It outputs a semi-functional challenge ciphertext as in the proof of Lemma 6.4, answers non-special key queries with pseudo-SF keys, and answers the special key query with a normal key computed from $[\![\boldsymbol{B}]\!]_2$, using the digests computed from the simulated public key. When $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{w}$, the public key is normal and $\mathcal{B}_{\mathsf{col}}$ simulates $\mathsf{Game}_{Q,2}$, and when $\boldsymbol{v} \leftarrow\!\!\$\ \mathbb{Z}_p^{k+1}$, the public key is of type $\widehat{\imath}$ and $\mathcal{B}_{\mathsf{col}}$ simulates $\mathsf{Game}_{\mathsf{col},1}$.

**Lemma 6.7.** *For a PPT adversary $\mathcal{A}$,*

$$|\mathsf{Adv}_{\mathsf{col},1} - \mathsf{Adv}_{\mathsf{col},2}| \le \frac{1}{p}.$$

*Proof.* The two games differ only if $\widehat{\jmath} \neq \bot$ and the special key is queried, in which case $\widehat{\imath} \in X^* \setminus S^*$. Let $\delta := (\boldsymbol{b}^\perp)^\top \boldsymbol{d}_{\widehat{\imath}}$, which is 0 with probability $1/p$ in both games. It suffices to show that the two games are identically distributed if $\delta \neq 0$.

By parameter hiding (Lemma 6.1), we may replace $(\boldsymbol{U}, \boldsymbol{W}_{\widehat{\imath}})$ by $(\boldsymbol{U} + \hat{u}\boldsymbol{V}, \boldsymbol{W}_{\widehat{\imath}} + \hat{w}\boldsymbol{V})$ with $\hat{u}, \hat{w} \leftarrow\!\!\$\ \mathbb{Z}_p$ in both games. This does not change $[\![\boldsymbol{A}^\top \boldsymbol{U}]\!]_1$, $[\![\boldsymbol{A}^\top \boldsymbol{W}_{\widehat{\imath}}]\!]_1$, or $[\![\boldsymbol{W}_{\widehat{\imath}} \boldsymbol{d}_j]\!]_2 = [\![\boldsymbol{W}_{\widehat{\imath}} \boldsymbol{B} \boldsymbol{r}_j]\!]_2$ for $j \neq \widehat{\imath}$, and the only other product involving $\boldsymbol{W}_{\widehat{\imath}}$ is the diagonal $\boldsymbol{W}_{\widehat{\imath}} \boldsymbol{d}_{\widehat{\imath}}$, which is not published. Since $\widehat{\imath} \in X^*$, it adds $(\hat{u} + \hat{w})(\boldsymbol{c}^\top \boldsymbol{a}^\perp)[\![(\boldsymbol{b}^\perp)^\top]\!]_1$ to the second component of the challenge ciphertext. By (26), $(\boldsymbol{b}^\perp)^\top \boldsymbol{d}_S = \mathbf{1}[\widehat{\imath} \notin S]\, \delta$, so it adds $\mathbf{1}[\widehat{\imath} \notin S_\ell]\, \delta \hat{u}\, \boldsymbol{a}^\perp$ to the exponent of the $\ell$-th key. For a non-special key, this is absorbed by its fresh $\gamma_\ell$. For the special key, $\widehat{\imath} \notin S^*$, so this adds $\delta \hat{u}\, \boldsymbol{a}^\perp$. Hence $(\hat{u}, \hat{w})$ appears in the view only through $\hat{u}$ in the special key and through $\hat{u} + \hat{w}$ in the challenge ciphertext, which are independent and uniform. Since $\delta \neq 0$, the special key in $\mathsf{Game}_{\mathsf{col},1}$ has a uniform and independent semi-functional component, as in $\mathsf{Game}_{\mathsf{col},2}$.

**Lemma 6.8.**

$$|\mathsf{Adv}_{\mathsf{col},2}| \le \frac{1}{p}.$$

*Proof.* In $\mathsf{Game}_{\mathsf{col},2}$, every key has a fresh semi-functional component $\gamma_\ell$. We use the fact that the following two distributions are identical:

$$\big(\boldsymbol{\alpha}, \{\gamma_\ell\}_\ell\big) \qquad \text{and} \qquad \big(\boldsymbol{\alpha} + \mu\, \boldsymbol{a}^\perp, \{\gamma_\ell - \mu\}_\ell\big),$$

where $\boldsymbol{\alpha} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$ and $\gamma_\ell, \mu \leftarrow\!\$\ \mathbb{Z}_p$. The extra term $\mu\,\boldsymbol{a}^\perp$ does not appear in the public key since $\boldsymbol{A}^\top(\boldsymbol{\alpha}+\mu\,\boldsymbol{a}^\perp) = \boldsymbol{A}^\top\boldsymbol{\alpha}$, and it does not appear in the keys since $\boldsymbol{\alpha}+\mu\,\boldsymbol{a}^\perp+(\gamma_\ell-\mu)\boldsymbol{a}^\perp = \boldsymbol{\alpha}+\gamma_\ell\,\boldsymbol{a}^\perp$. The challenge ciphertext contains $[\![\boldsymbol{c}^\top\boldsymbol{\alpha}+\mu\,\boldsymbol{c}^\top\boldsymbol{a}^\perp]\!]_{\mathsf{T}} + M_b$. If $\boldsymbol{c}^\top\boldsymbol{a}^\perp \neq 0$, which happens with probability $1-1/p$ over $\boldsymbol{c} \leftarrow\!\$\ \mathbb{Z}_p^{k+1}$, this element is uniform in $\mathbb{G}_T$ and independent of the rest of the view, so $\mathcal{A}$ outputs $b$ with probability exactly $\frac{1}{2}$.

*Proof of Theorem 6.2.* By Lemmas 6.3 to 6.8, $|\mathsf{Adv}_{\mathsf{guess}}| \le \Delta$. By Lemma 6.2,

$$\mathsf{Adv}^{\mathsf{SUB}}_{\Pi_\subseteq,\mathcal{A}}(\lambda,n) = 2|\mathsf{Adv}_{\mathsf{real}}| = 2n(Q+1)|\mathsf{Adv}_{\mathsf{guess}}| \le 2n(Q+1)\,\Delta.$$

The proof uses one MDDH transition in $\mathbb{G}$ and at most $Q+1$ in $\mathbb{H}$, and since $p \ge 2^{\Theta(\lambda)}$, the term $(Q+2)/p$ is negligible.

**Exponential-size Identity Space.** We apply the compiler of Section 4, in its non-threshold form (Remark 4.1), to $\Pi_\subseteq$ with the family of Lemma 4.1 and the identity mapping of Section 4.3, exactly as in Section 5.2. We denote the resulting batched IBE scheme by $\Pi^{\mathsf{LU}}$.

**Theorem 6.3.** *Let $B, Q$ be polynomially bounded and let $k \ge 1$ be fixed. The scheme $\Pi^{\mathsf{LU}}$ is a perfectly correct batched IBE scheme with identity space $\{0,1\}^\lambda$ and batch bound $B$. For every admissible PPT adversary $\mathcal{A}$ that makes at most $Q$ key queries,*

$$\mathsf{Adv}^{\mathsf{BIBE}}_{\Pi^{\mathsf{LU}},\mathcal{A}}(\lambda,B) \le 2q^2(Q+1)\,\Delta, \tag{27}$$

*where $\Delta$ is given by (25) with $n = q^2$. In particular, $\Pi^{\mathsf{LU}}$ is adaptively secure under the $\mathcal{U}_k$-MDDH assumption in $\mathbb{G}$ and in $\mathbb{H}$.*

*Proof.* Correctness follows from Theorem 6.1 and part (a) of Theorem 4.1. By part (b) of Theorem 4.1 and Remark 4.1, $\mathcal{A}$ yields an admissible adversary against $\Pi_\subseteq$ with the same advantage and at most $Q$ key queries, and (27) follows from Theorem 6.2 with $n = q^2$. □

The compiler does not change public keys, ciphertexts, digests, or keys.

**Efficiency.** Compared to $\Pi_\subseteq$ with $n = q^2$, the compiled scheme additionally computes $C_{\mathsf{id}}$. Encryption computes $C_{\mathsf{id}}$, which takes $O(q\lambda)$ operations in $\mathbb{F}_q$, and Digest and Decrypt compute the union $S_\Gamma$ of at most $B$ blocks, which takes $O(Bq\lambda)$ operations in $\mathbb{F}_q$. ComputeKey is that of $\Pi_\subseteq$ and takes $O(k^2)$ group operations. In encryption, the vector $[\![\boldsymbol{A}^\top(\boldsymbol{U}+\boldsymbol{W}_{C_{\mathsf{id}}})]\!]_1$ takes $O(k^2q)$ group operations, but it depends only on the identity, so it can be computed offline and reused, as described in Section 4.3. Given this vector, online encryption takes $O(k^2)$ group operations, independently of $B$. In addition to computing $S_\Gamma$, the digest takes $O(k(n-|S_\Gamma|))$ group operations, and decryption takes $4(k+1)$ pairings and $O(kq(n-|S_\Gamma|)) \subseteq O(kq^3)$ group operations.

# References


[1] A. Agarwal, R. Fernando, and B. Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Y. T. Kalai and S. F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 69–100, Santa Barbara, CA, USA, Aug. 17–21, 2025. Springer, Cham, Switzerland.

[2] M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In Y. Ishai, editor, *Theory of Cryptography*, pages 235–252, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275, Santa Barbara, CA, USA, Aug. 14–18, 2005. Springer Berlin Heidelberg, Germany.

[4] D. Boneh, E. Laufer, and E. N. Tas. Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Report 2025/1254, 2025.

[5] J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 595–624, Sofia, Bulgaria, Apr. 26–30, 2015. Springer Berlin Heidelberg, Germany.

[6] J. Chen, J. Gong, L. Kowalczyk, and H. Wee. Unbounded ABE via bilinear entropy expansion, revisited. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 503–534, Tel Aviv, Israel, Apr. 29 – May 3, 2018. Springer, Cham, Switzerland.

[7] A. R. Choudhuri, S. Garg, G.-V. Policharla, and M. Wang. Practical mempool privacy via one-time setup batched threshold encryption. In L. Bauer and G. Pellegrino, editors, *USENIX Security 2025: 34th USENIX Security Symposium*, pages 3477–3495, Seattle, WA, USA, Aug. 13–15, 2025. USENIX Association.

[8] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy*, pages 910–927, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press.

[9] N. Döttling, B. Magri, B. Marsh, and M. S. Rajasree. Batch decryption from new standard assumptions. Cryptology ePrint Archive, Paper 2026/2252, 2026.

[10] P. Erdös, P. Frankel, and Z. Furedi. Families of finite sets in which no set is covered by the union of $r$ others. *Israeli Journal of Mathematics*, 51:79–89, 1985.

[11] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for diffie–hellman assumptions. In *Journal of Cryptology*, 2017.

[12] R. Fernando, G.-V. Policharla, A. Tonkikh, and Z. Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Report 2025/2032, 2025.

[13] R. Gay, L. Kowalczyk, and H. Wee. Tight adaptively secure broadcast encryption with short ciphertexts and keys. In D. Catalano and R. De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 123–139, Amalfi, Italy, Sept. 5–7, 2018. Springer, Cham, Switzerland.

[14] J. Gong, B. Waters, H. Wee, and D. J. Wu. Threshold batched identity-based encryption from pairings in the plain model. Lecture Notes in Computer Science, pages 548–578, 2026.

[15] W. H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

[16] A. Kavousi, D. V. Le, P. Jovanovic, and G. Danezis. BlindPerm: Efficient MEV mitigation with an encrypted mempool and permutation. Cryptology ePrint Archive, Report 2023/1061, 2023.

[17] K. Lee. Verifiable outsourced bte with silent setup: Achieving constant-rate batched delegation. *Cryptology ePrint Archive*, 2026.

[18] A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335, Cambridge, UK, Apr. 15–19, 2012. Springer Berlin Heidelberg, Germany.

[19] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479, Zurich, Switzerland, Feb. 9–11, 2010. Springer Berlin Heidelberg, Germany.

[20] G.-V. Policharla. Labeled multi-key batched ibe. *Cryptology ePrint Archive*, 2026.

[21] R. Sarkar, S. Murshid, and M. Mandal. Batched attribute-based encryption from bilinear pairings. *Cryptology ePrint Archive*, 2026.

[22] R. Stracovsky, B. Waters, and D. J. Wu. Lifting bounded-collusion security to full security in pairing-based encryption schemes. Cryptology ePrint Archive, Paper 2026/2389, 2026.

[23] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, Aug. 16–20, 2009. Springer Berlin Heidelberg, Germany.

[24] H. Wee. Circuit ABE with $\mathsf{poly}(\mathrm{depth}, \lambda)$-sized ciphertexts and keys from lattices. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 178–209, Santa Barbara, CA, USA, Aug. 18–22, 2024. Springer, Cham, Switzerland.

[25] S. Xiong, Y. Zhang, and J. Chen. Efficient batched IBE from lattices in the standard model. Cryptology ePrint Archive, Report 2025/2158, 2025.