VOBE: Verifiable Outsourced Batched Encryption for Secure Delegation of Batched Decryption

Kwangsu Lee*

Abstract

Batched encryption (BE) has emerged as a novel public-key cryptographic paradigm that enables the efficient decryption of a designated batch of B ciphertexts simultaneously. By incorporating threshold decryption capabilities into this framework, batched threshold encryption (BTE) further decentralizes the decryption process. While both BE and BTE serve as highly effective solutions for mitigating Miner Extractable Value (MEV) attacks in blockchain networks by providing robust mempool privacy, ciphertext integrity, and communication efficiency, they still suffer from heavy computational overhead during the ciphertext decryption phase. In this paper, we address this computational bottleneck by introducing a novel framework that delegates the heavy decryption workloads to an untrusted cloud server while enabling verifiability of the outsourced computations. To achieve this, we first propose an outsourced batched identity-based encryption (O-BIBE) scheme by integrating outsourcing functionalities into the conventional BIBE paradigm, accompanied by a rigorous security proof. We then construct a verifiable outsourced batched encryption (VOBE) scheme by strategically combining O-BIBE with other core cryptographic building blocks and formally prove its security. To eliminate the single point of failure and enhance threshold resiliency, we extend our framework to the threshold setting by developing an outsourced threshold batched identity-based encryption (O-TBIBE) scheme. Building upon this, we propose a verifiable outsourced batched threshold encryption (VOBTE) scheme, which successfully achieves decentralized threshold resilience. Our proposed VOBE and VOBTE schemes are the first to concurrently guarantee ciphertext integrity and mempool privacy against sophisticated blockchain attacks, while significantly reducing decryption costs via efficient and verifiable outsourcing.

Keywords: Batched encryption, Batched threshold encryption, Verifiable outsourcing, Bilinear maps.

*Sejong University, Seoul, Korea. Email: kwangsu@sejong.ac.kr.

1 Introduction

Batched encryption (BE) is a special kind of public-key encryption (PKE) designed to achieve highly efficient decryption for a batch of ciphertexts; once a specific batch of B ciphertexts is determined, a single batch decryption key is issued by a decryption server for a system to decrypt the entire batch of ciphertexts. When this primitive is integrated with a threshold framework—where N decentralized servers of a decryption committee securely perform distributed decryption operations—it evolves into a batched threshold encryption (BTE) scheme, which guarantees threshold-based resiliency [12, 13]. A primary and highly consequential application of BTE is mitigating miner extractable value (MEV) attacks in blockchain environments [14]. An MEV attack occurs when a greedy miner or validator exploits its authority to inspect pending, unencrypted transactions in the public mempool and arbitrarily manipulates their execution order within a block, thereby extracting financial benefits before the transactions are processed by decentralized applications. To prevent such adversarial manipulation, a robust defense mechanism requires users to submit their transactions in an encrypted form. Only after these encrypted transactions are irreversibly committed and ordered within the blockchain do the decentralized decryption servers collectively decrypt the batch, thereby effectively concealing transaction contents until their execution order is finalized.

However, a simple approach to build a BTE scheme from threshold PKE (TPKE) with distributed key generation (DKG) introduces a severe bottleneck such that the communication overhead between the decryption servers and the system scales up to $O(NB)$ [3]. To mitigate this communication burden, one might consider employing threshold identity-based encryption (TIBE) [6] by treating a block identifier as the identity; in this approach, ciphertexts are encrypted under a specific block ID, and servers release their key shares for that ID to achieve an $O(N)$ communication size [18, 29]. Nevertheless, this approach inherently fails to guarantee mempool privacy, because the leaked decryption key for a block ID inevitably decrypts all ciphertexts associated with that ID, including those outside of the targeted batch. An ideal BTE scheme must provide efficient communication and robust mempool privacy. To resolve this critical privacy issue in blockchains, several BTE schemes have recently been proposed by leveraging sophisticated algebraic tools, such as polynomial commitments or key-homomorphic puncturable pseudo-random functions [17, 28], to strictly bind a selected batch of ciphertexts into a single aggregate structure [1, 9, 10, 12, 13]. These modern BTE schemes achieve high communication efficiency, as the size of the decryption key remains independent of the batch size B, while successfully ensuring mempool privacy by restricting decryption exclusively to the selected batch of ciphertexts while maintaining confidentiality for all others.

Despite these advancements, existing BTE schemes still suffer from a significant computational bottleneck: decrypting a single ciphertext requires an $O(B\log B)$ overhead to compute the associated cryptographic proof, which inherently scales the total computational complexity up to $O(B^2\log B)$ for a batch of B ciphertexts. Although this batch decryption overhead can be theoretically mitigated to $O(B\log B)$ by leveraging the discrete Fourier transform (DFT) technique [19], it still imposes a substantial computational burden on the system as the batch size B expands. Consequently, minimizing the cryptographic workload of batch decryption while simultaneously maintaining a compact decryption key size and robust mempool privacy remains a critical, unresolved challenge in the design of scalable BTE systems.

1.1 Our Contribution

To resolve the computational efficiency bottlenecks inherent in existing BTE schemes, in this paper, we propose a methodology that delegates the heavy cryptographic workloads of batch decryption to an untrusted cloud server, inspired by the concept of outsourced attribute-based encryption (OABE) of Green et al. [27], while providing a robust mechanism to verify the correctness of the returned results. Based on this core

approach, the principal contributions of our work are outlined as follows:

Our first contribution is the introduction of an outsourced batched identity-based encryption (O-BIBE) scheme, which delegates the heavy decryption operations of a conventional BIBE scheme to a cloud server. To achieve this, we first extend the traditional BIBE syntax into the O-BIBE paradigm and formalize a robust security model that captures the unique vulnerabilities arising from outsourcing. In terms of algorithmic structure, O-BIBE enhances the standard BIBE architecture by incorporating three additional algorithms: Delegate, Transform, and Recover. Specifically, a client utilizes the Delegate algorithm to authorize outsourcing tasks for a ciphertext batch; the cloud server then executes the delegated work using the Transform algorithm; and finally, the client derives the plaintext using the Recover algorithm on the server's intermediate response. We realize an efficient O-BIBE scheme by modifying the GWWW-BIBE scheme of Gong et al. [26], which is proven secure under a q-type assumption. The core technical mechanism of our outsourcing approach relies on re-randomizing the decryption key DK to generate a transformation key TK that completely hides the original key information. By virtue of this information-theoretic blinding, we prove that even if an adversary observes multiple delegation tasks including TK for batches containing the challenge ciphertext, it remains computationally infeasible to deduce any partial information about the challenge ciphertext.

Our second contribution is the formalization and construction of a verifiable outsourced batched encryption (VOBE) scheme, which integrates secure decryption outsourcing into the traditional BE framework. Similar to standard CCA-secure BE schemes, VOBE supports efficient batch decryption while inherently guaranteeing ciphertext integrity and mempool privacy, making it particularly well-suited for blockchain environments. To optimize batch processing, we extend the BE paradigm by delegating heavy decryption workloads to an untrusted cloud server, augmented with a verification mechanism to ensure the correctness of the outsourced computation. To conceptualize this, we propose a novel security model that simultaneously captures both ciphertext confidentiality and outsourcing verifiability. We then instantiate an efficient and secure VOBE scheme by strategically combining O-BIBE, one-time signatures (OTS), collision-resistant hash functions (CRHF), pseudo-random functions (PRF), and symmetric-key encryption (SKE). In our VOBE architecture, reminiscent of the classical CHK transformation for achieving CCA security [11], the hash of the OTS verification key is utilized as the identity for the underlying O-BIBE scheme, and the OTS signature guarantees ciphertext integrity. Furthermore, to enable outsourcing verifiability, the encryption algorithm employs a PRF to generate a validation tag, which confirms that the correct O-BIBE encapsulated key has been derived. During the verification phase, the client applies the PRF to the cloud server's intermediate output and cross-checks it against the tag embedded in the ciphertext, thereby seamlessly verifying the correctness of the outsourced computation.

Our third contribution is the formalization and construction of an outsourced threshold batched identity-based encryption (O-TBIBE) scheme and a verifiable outsourced batched threshold encryption (VOBTE) scheme, which extend the O-BIBE and VOBE paradigms to the threshold decryption setting. To realize the O-TBIBE scheme, we incorporate outsourcing functionalities into the GWWW-TBIBE scheme derived from the BF-IBE scheme [6] and analyzed in the generic group model (GGM). Building upon this foundation, we construct our VOBTE scheme by strategically integrating O-TBIBE with several core cryptographic building blocks, including OTS, CRHF, PRF, and SKE. The VOBTE framework achieves a synergistic integration of threshold-based key derivation through a decryption committee, efficient delegation of heavy decryption workloads to an untrusted cloud server, and outsourcing verifiability of the delegated computations. By providing this comprehensive extension, we present a highly efficient and verifiable solution for batch decryption that is robust against single points of failure, making it an ideal candidate for secure and scalable decentralized systems.

1.2 Related Work

Threshold Encryption and Threshold Signatures. The security and decentralization of our proposed schemes are fundamentally rooted in the evolution of threshold cryptography, which enables a distributed committee to collectively perform cryptographic operations without a single point of failure. Since the seminal work on secret sharing by Shamir [31], threshold cryptography has matured through the development of threshold encryption [15] and threshold signatures [24]. Early threshold constructions often relied on interactive distributed key generation (DKG) protocols [25], which impose significant communication overhead and lack scalability in dynamic network environments. To overcome these constraints, recent advancements have focused on non-interactive and silent setups [2, 16, 20–23, 32], which reduce or eliminate the need for repeated interactive setups and complex coordination among participants. While these primitives provide the necessary resilience for decentralized infrastructures, their integration into batched decryption frameworks remains computationally intensive for the end-system.

MEV Attacks and Batched Threshold Encryption. MEV attacks have emerged as a critical security concern in blockchain networks, where greedy miners or validators exploit their authority to inspect pending transactions in the public mempool and maliciously manipulate their execution order or block inclusion timing to extract financial gains [14]. The cryptographic standard to mitigate MEV attacks is the commit-and-reveal paradigm, which mandates that transactions are submitted in an encrypted form and subsequently decrypted by the system only after they are irreversibly committed and ordered on the blockchain.

Early instantiations of this paradigm primarily relied on TPKE [3]. However, as mentioned before, TPKE-based approaches suffer from a severe communication bottleneck, where the overhead between the decryption nodes and the system scales linearly with the batch size, resulting in $O(NB)$ communication complexity, where N denotes the number of decryption nodes and B represents the batch size. Furthermore, they require re-executing a costly distributed key generation (DKG) protocol whenever the decryption committee changes. To mitigate these communication burdens, alternative approaches explored TIBE, where decryption key shares are generated for a specific block identifier ID, reducing the partial decryption communication size to $O(N)$ [18, 29]. Nevertheless, TIBE inherently fails to guarantee mempool privacy; once the decryption key for a block ID is disclosed, all transactions associated with that block are inevitably exposed, including those not selected for the targeted batch.

To resolve this privacy problem, Choudhuri et al. [12] introduced the notion of BTE and proposed an efficient BTE scheme, which utilizes KZG commitments to bind the batch of ciphertexts and zero-knowledge proofs (ZKP) to ensure ciphertext integrity in the chosen-ciphertext setting. While this BTE scheme achieves $O(N)$ communication efficiency and robust mempool privacy by restricting decryption exclusively to the selected ciphertext batch, it requires an inefficient setup phase that uses a multi-party computation (MPC) protocol for every epoch. To eliminate this epoch-dependence, Bormet et al. [9, 10] proposed a CCA-secure, epochless BTE scheme by leveraging a time-lock puzzle based on key-homomorphic puncturable PRF. Recently, Boneh et al. [7] proposed BE and BTE schemes without epoches based on lattices. Unlike epoch-based BTE schemes that require users to re-encrypt unselected transactions for subsequent blocks, this epochless approach allows unselected ciphertexts to naturally transition into the next block. However, this PRF-based BTE approach introduces a practical drawback, as it requires strict coordination and sorting of indices among individual users.

To simultaneously achieve communication efficiency and robust mempool privacy without complex coordination, BIBE schemes were recently proposed [1, 13]. By fixing the selected batch of ciphertexts using a KZG commitment [28] and issuing decryption keys compounded with BLS signatures [8] at each epoch, these BIBE schemes successfully eliminate the necessity of periodic MPC protocols. Building upon

this BIBE paradigm, recent work by Gong et al. [26] proposed a BIBE scheme from the BB-IBE scheme [5] secure under a q-type assumption, alongside its threshold-based variant (TBIBE) based on the BF-IBE scheme [6], and a TBIBE extension supporting silent key setups (TBIBE-SS) based on the silent threshold cryptography of Waters and Wu [32]. While these advancements significantly mature the foundations of batched decryption, optimizing the computational overhead on the verifier side while preserving threshold resiliency remains an open challenge that we address in this work.

2 Preliminaries

In this section, we establish the fundamental notations and cryptographic primitives that serve as the building blocks for our proposed schemes. We begin by defining the properties of bilinear groups and the computational assumptions upon which our security proofs are constructed. Next, we review several essential cryptographic tools, including CRHF, PRF, OTS, and SKE, which provide the necessary framework to design and analyze the security of our proposed schemes.

2.1 Notations

Let $\lambda \in \mathbb{N}$ denote the security parameter. A function $f(\lambda)$ is said to be negligible, denoted as $f(\lambda) = \mathsf{negl}(\lambda)$, if for every positive polynomial $p(\lambda)$ there exists an integer λ_0 such that $|f(\lambda)| < 1/p(\lambda)$ for all $\lambda > \lambda_0$. We use the abbreviation PPT to denote probabilistic polynomial-time.

For a finite set S, $s \leftarrow S$ represents the operation of sampling an element s uniformly at random from S, and $|S|$ denotes its cardinality. For a positive integer B, we define $[B] := \{1, 2, \ldots, B\}$ and $[0, B] := \{0, 1, \ldots, B\}$. The string $\{0,1\}^\lambda$ represents the set of all binary strings of length λ. An algorithmic assignment $y \leftarrow \mathsf{Alg}(x)$ outputs y by running a deterministic or probabilistic algorithm Alg on input x. If Alg is probabilistic, the output is a random variable determined by the internal random coins of Alg. We use $\perp$ to denote a special failure or error symbol returned by an algorithm.

2.2 Bilinear Groups and Complexity Assumptions

Definition 2.1 (Prime-Order Asymmetric Bilinear Group)**.** An asymmetric bilinear group generator $\mathcal{G}$ takes as input the security parameter λ and outputs the description of a bilinear group $\mathsf{GD} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g})$ where p is a random prime, $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ are three cyclic groups of prime order p, and g and $\hat{g}$ are generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. The bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u \in \mathbb{G}, \forall \hat{v} \in \hat{\mathbb{G}}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, \hat{v}^b) = e(u, \hat{v})^{ab}$.
2. Non-degeneracy: $\exists g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$ such that $e(g, \hat{g})$ has order p in $\mathbb{G}_T$.

We say that $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are asymmetric bilinear groups with no efficiently computable isomorphisms if the group operations in $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable, but there are no efficiently computable isomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Assumption 1 (ℓ-Bilinear Diffie-Hellman Exponent Variant [26])**.** Let $\mathsf{GD} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g})$ be an asymmetric bilinear group generated by $\mathcal{G}(1^\lambda)$. The ℓ-bilinear Diffie-Hellman exponent variant (ℓ-BDHEV) assumption is that if the challenge tuple

$$D = \left(\mathsf{GD},\ g^b,\ g^s,\ g^{ab},\ \hat{g}^a,\ \hat{g}^b,\ g^\tau,\ g^{\tau ab},\ g^{\tau abs},\ \left\{\hat{g}^{\tau^i},\ \right\}_{i\in[\ell]},\ \left\{\hat{g}^{\tau^i ab}\right\}_{i\in[\ell]}\right) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = e(g,\hat{g})^{abs}$ from $Z = Z_1 = e(g,\hat{g})^{d}$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\ell\text{-BDHEV}}_{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(D,Z_0)=0] - \Pr[\mathcal{A}(D,Z_1)=0]\right|$ where the probability is taken over random choices of $\tau, a, b, s, d \in \mathbb{Z}_p$.

2.3 Collision-Resistant Hash Function

Definition 2.2 (Collision-Resistant Hash Function)**.** A family of hash functions (HF) is a collection of efficiently computable functions $\mathcal{H} = \{\mathsf{H}_s : \mathcal{X} \to \mathcal{Y}\}_{s\in\mathcal{K}}$ where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range. The collision-resistance of an HF is defined as the following experiment $\mathsf{Exp}^{\mathsf{CR}}_{HF,\mathcal{A}}(\lambda)$:

1. A hash function H_s is sampled from $\mathcal{H}$ uniformly at random.
2. The adversary $\mathcal{A}$ is given the hash function H_s and outputs a pair of inputs $(x,x') \in \mathcal{X}^2$.
3. The experiment returns 1 if $x \neq x'$ and $\mathsf{H}_s(x) = \mathsf{H}_s(x')$. Otherwise, it returns 0.

The advantage of $\mathcal{A}$ in the collision resistance experiment is defined as $\mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{CR}}_{HF,\mathcal{A}}(\lambda) = 1]$. We say that an HF is collision-resistant if this advantage is negligible for all PPT adversaries.

2.4 Pseudo-Random Function

Definition 2.3 (Pseudo-Random Function)**.** A pseudo-random function (PRF) is an efficiently computable function $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range. Let $\mathcal{F}_{all} = \{f : \mathcal{X} \to \mathcal{Y}\}$ be the set of all functions mapping $\mathcal{X}$ to $\mathcal{Y}$. We say that F satisfies pseudo-randomness if for all PPT adversaries $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{A}}(\lambda) = \left|\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{F}(k,\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{\$} \mathcal{F}_{all} : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]\right|$$

is negligible, where the first probability is over the uniform choice of $k \in \mathcal{K}$ and the second is over the uniform choice of $f \in \mathcal{F}_{all}$, and both include the internal coins of $\mathcal{A}$.

2.5 One-Time Signature

Definition 2.4 (One-Time Signature)**.** A one-time signature (OTS) scheme with message space of $\mathcal{M}$ consists of the following three algorithms:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{SK},\mathsf{VK})$: The key generation algorithm takes as input a security parameter λ. It outputs a signing key SK and a verification key VK.

$\mathsf{Sign}(\mathsf{SK},M) \to \sigma$: The signing algorithm takes as input the signing key SK and a message $M \in \mathcal{M}$. It outputs a signature σ.

$\mathsf{Verify}(\mathsf{VK},M,\sigma) \to \{0,1\}$: The (deterministic) verification algorithm takes as input the verification key VK, a message $M \in \mathcal{M}$, and a signature σ. It outputs 1 if the signature is valid and 0 otherwise.

The correctness of an OTS scheme is defined as follows: For all $(\mathsf{SK},\mathsf{VK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and any message $M \in \mathcal{M}$, it is required that $\mathsf{Verify}(\mathsf{VK},M,\mathsf{Sign}(\mathsf{SK},M)) = 1$.

Definition 2.5 (Strong Unforgeability)**.** The strong unforgeability (SUF) of an OTS scheme is defined via the following experiment $\mathsf{Exp}^{\mathsf{SUF}}_{OTS,\mathcal{A}}(\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. $\mathcal{C}$ first generates a key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives VK to $\mathcal{A}$.

2. $\mathcal{A}$ requests at most one signature query on a message M. $\mathcal{C}$ generates a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, M)$ and gives σ to $\mathcal{A}$.

3. Finally, $\mathcal{A}$ outputs a forged pair (M^*, σ^*). The experiment returns 1 if the forged pair satisfies the following conditions, and returns 0 otherwise: 1) $\mathsf{Verify}(\mathsf{VK}, M^*, \sigma^*) = 1$, 2) $(M^*, \sigma^*) \neq (M, \sigma)$ where (M, σ) is the pair of the signature query.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{SUF}}_{OTS,\mathcal{A}}(\lambda) = 1]$ where the probability is taken over all internal randomness of the experiment. An OTS scheme is said to be SUF secure if the advantage is negligible for all PPT adversaries.

Remark 1. An OTS scheme can be efficiently instantiated by leveraging an ephemeral (one-time) variant of the classical Schnorr signature scheme [30]. To achieve a highly compact signature size, we implement this one-time Schnorr signature over the prime-order group $\mathbb{G}$ of a bilinear pairing group. Furthermore, to minimize the verification cost of many signatures signed by different signers, we use the fast batch verification methodology based on the small exponent test proposed by Bellare et al. [4]. By computing a random linear combination of the verification equations using small random exponents, the verifier can validate the integrity of B ciphertexts, drastically scales down the total number of multi-scalar multiplications.

2.6 Symmetric Key Encryption

Definition 2.6 (Symmetric Key Encryption)**.** A symmetric key encryption (SKE) scheme consists of the following three algorithms:

$\mathsf{KeyGen}(1^\lambda) \to K$: The key generation algorithm takes as input a security parameter λ. It outputs a symmetric key $K \in \{0,1\}^\lambda$.

$\mathsf{Encrypt}(K, M) \to \mathsf{CT}$: The encryption algorithm takes as input a symmetric key K and a message M. It outputs a ciphertext CT.

$\mathsf{Decrypt}(K, \mathsf{CT}) \to M$: The decryption algorithm takes as input a symmetric key K and a ciphertext CT. It outputs a message M.

The correctness of an SKE scheme is defined as follows: For all $K \leftarrow \mathsf{KeyGen}(1^\lambda)$ and any message M, it is required that $\mathsf{Decrypt}(K, \mathsf{Encrypt}(K, M)) = M$.

Definition 2.7 (Indistinguishability under One-Time Security)**.** The indistinguishability under one-time security (IND-OT) of a SKE scheme is defined via the following experiment $\mathsf{Exp}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{A}}(\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. $\mathcal{C}$ generates a symmetric key $K \leftarrow \mathsf{KeyGen}(1^\lambda)$.

2. $\mathcal{A}$ submits two challenge messages M_0^*, M_1^* of equal length ($|M_0^*| = |M_1^*|$). $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$, computes the challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(K, M_\mu^*)$, and gives CT^* to $\mathcal{A}$.

3. Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. The experiment returns 1 if $\mu = \mu'$ and 0 otherwise.

The advantage of $\mathcal{A}$ in the IND-OT experiment is defined as $\mathsf{Adv}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$ where the probability is taken over all internal randomness of the experiment. An SKE scheme is said to be IND-OT secure if the advantage is negligible for all PPT adversaries.

3 Outsourced Batched Identity-Based Encryption

In this section, we present the formal definition, construction, and security analysis of our O-BIBE scheme. We first establish the formal syntax and the security model, which incorporates the unique oracle-based interactions necessary for secure outsourcing. Subsequently, we describe the concrete instantiation of the O-BIBE scheme, demonstrating how it efficiently handles batched identities while delegating heavy decryption tasks to an untrusted server. Finally, we provide a comprehensive security proof to show that our construction achieves the desired level of security under the predefined computational assumptions.

3.1 Definition

An O-BIBE scheme is an extension of the BIBE scheme, where a single decryption key is associated with a set of identities S rather than a single ID, enabling the decryption of multiple ciphertexts with one key. Our O-BIBE scheme further extends this capability by allowing the outsourcing of heavy decryption operations to a cloud server.

In the O-BIBE framework, a trusted central authority first initializes the system by generating the master key (MK), the public encryption key (EK), and the public key (PK). Individual senders use the EK to generate ciphertext headers (CH) associated with their specific ID and a batch label BL, which are then uploaded to a common ciphertext pool. To perform decryption, a decryptor selects a batch of B ciphertexts, generates a digest DIG for the set S of identities associated with these ciphertexts, and requests the corresponding decryption key DK from the central authority. With this DK, the decryptor can locally derive the capsule keys for any ciphertext whose ID is included in the set S. If the decryptor (as a client) chooses to delegate the decryption of these B ciphertexts to a cloud server, it employs a delegation algorithm to generate a recovery key (RK), a transformation key (TK), and a set of work tasks (WT). The client then transmits the TK and the WT to the cloud server. The cloud server processes the B work tasks using the TK through a transformation algorithm to produce transformed tasks, which are returned to the client. Finally, the client uses the RK to efficiently derive the B capsule keys from the transformed tasks.

Following this high-level operational flow, we formally define the syntax of the O-BIBE scheme as follows:

Definition 3.1 (O-BIBE)**.** An outsourced batched identity-based encryption (O-BIBE) scheme with a maximum batch size $B \in \mathbb{N}$, identity space of $\mathcal{I}$, batch label space of $\mathcal{L}$, and capsule key space of $\mathcal{K}$ consists of the following eight algorithms:

$\mathsf{Setup}(1^\lambda, 1^B) \to (\mathsf{MK}, \mathsf{EK}, \mathsf{PK})$: The setup algorithm takes as input the security parameter λ and the batch size B. It outputs a master key MK, a public encryption key EK, and a public key PK.

$\mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}, \mathsf{BL}) \to (\mathsf{CH}, \mathsf{CK})$: The encapsulation algorithm takes as input the public encryption key EK, an identity $\mathsf{ID} \in \mathcal{I}$, and a batch label $\mathsf{BL} \in \mathcal{L}$. It outputs a ciphertext header CH and a capsule key $\mathsf{CK} \in \mathcal{K}$.

$\mathsf{Digest}(\mathsf{PK}, S) \to \mathsf{DIG}$: The digest algorithm takes as input the public key PK and a set of identities $S \subseteq \mathcal{I}$. It outputs a digest DIG associated with S.

$\mathsf{ComputeKey}(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL}) \to \mathsf{DK}$: The key computation algorithm takes as input the master key MK, a digest DIG, and a batch label BL. It outputs a decryption key DK.

$\mathsf{Decaps}(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, \mathsf{ID}, \mathsf{BL}) \to \mathsf{CK}$: The decapsulation algorithm takes as input the public key PK, a decryption key DK, a set of identities S, a ciphertext header CH, an identity ID, and a batch label BL. It outputs the corresponding capsule key $\mathsf{CK} \in \mathcal{K}$.

$\mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$: The delegation algorithm takes as input the public key PK, a decryption key DK, ciphertext headers $(\mathsf{CH}_i)_{i\in U}$ for a subset $U \subseteq [B]$, and a batch label BL. It outputs a recovery key RK, a transformation key TK, and work tasks $(\mathsf{WT}_i)_{i\in U}$.

$\mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, (\mathsf{ID}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{TT}_i)_{i\in U}$: The transformation algorithm takes as input the public key PK, a transformation key TK, work tasks $(\mathsf{WT}_i)_{i\in U}$, identities $(\mathsf{ID}_i)_{i\in U}$, and a batch label BL. It outputs transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

$\mathsf{Recover}(\mathsf{RK}, (\mathsf{TT}_i)_{i\in U}) \to (\mathsf{CK}_i)_{i\in U}$: The recovery algorithm takes as input a recovery key RK and transformed tasks $(\mathsf{TT}_i)_{i\in U}$. It outputs a sequence of capsule keys $(\mathsf{CK}_i)_{i\in U} \in \mathcal{K}^{|U|}$.

Remark 2 (Functional Extension of BIBE). The O-BIBE scheme is constructed by extending the foundational algorithms of a standard BIBE scheme consisting of Setup, Encaps, Digest, ComputeKey, and Decaps. Specifically, to facilitate the secure delegation of decryption operations, O-BIBE incorporates three additional specialized algorithms: Delegate, Transform, and Recover. This structural composition allows the scheme to maintain the core functionality of batched identity-based encryption while providing a modular framework for efficient decryption outsourcing.

Definition 3.2. An O-BIBE scheme is correct if for any security parameter λ and any batch size $B \leq \mathrm{poly}(\lambda)$, the following two conditions hold:

- Correctness of Decryption: For any identity $\mathsf{ID} \in \mathcal{I}$, any batch label $\mathsf{BL} \in \mathcal{L}$, and any set $S \subseteq \mathcal{I}$ such that $\mathsf{ID} \in S$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Decaps}(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, \mathsf{ID}, \mathsf{BL}) & (\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B) \\ = \mathsf{CK} & (\mathsf{CH}, \mathsf{CK}) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}, \mathsf{BL}) \\ & \mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S) \\ & \mathsf{DK} \leftarrow \mathsf{ComputeKey}(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL}) \end{array}\right] = 1.$$

- Correctness of Outsourced Batch Decryption: For $S = \{\mathsf{ID}_i\}_{i\in[B]} \subseteq \mathcal{I}^B$ and $\mathsf{BL} \in \mathcal{L}$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Recover}(\mathsf{RK}, (\mathsf{TT}_i)_{i\in[B]}) & (\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B) \\ = (\mathsf{CK}_i)_{i\in[B]} & \forall i \in [B] : (\mathsf{CH}_i, \mathsf{CK}_i) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}_i, \mathsf{BL}) \\ & \mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S) \\ & \mathsf{DK} \leftarrow \mathsf{ComputeKey}(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL}) \\ & (\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in[B]}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in[B]}, \mathsf{BL}) \\ & (\mathsf{TT}_i)_{i\in[B]} \leftarrow \mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in[B]}, (\mathsf{ID}_i)_{i\in[B]}, \mathsf{BL}) \end{array}\right] = 1.$$

We define the security model for our O-BIBE scheme by extending the BIBE security model introduced by Gong et al. [26], specifically by granting the adversary access to an additional outsourcing oracle. In the selective IND-CPA model, the adversary first commits to a target identity ID^* and a target batch label BL^*, after which it receives the encryption key EK and the public key PK from the challenger. The adversary is then allowed to adaptively query the OCreateDK, ORevealDK, ODelegate, and ORevealRK oracles. Specifically, the ODelegate oracle provides the adversary with a transformation key and the associated

work tasks, thereby simulating the outsourcing of decryption operations. To further model the potential leakage of session-specific secrets, the $\mathsf{ORevealRK}$ oracle grants the adversary access to the corresponding recovery key for a given session. In the challenge phase, the adversary receives a challenge ciphertext header CH^* along with a challenge capsule key CK^*_μ, which is either the valid capsule key or a uniformly random element from the key space. Following the challenge, the adversary may continue to perform additional oracle queries. Finally, the adversary outputs a guess μ' for the challenge bit μ and wins if $\mu' = \mu$. To prevent trivial winning conditions, standard restrictions are imposed on the key exposure oracles (e.g., the adversary cannot reveal a decryption key for a set S that includes ID^* for the target label BL^*). Notably, no additional restrictions are placed on the delegation oracle queries, even for the target identity.

A more rigorous definition of the selective IND-CPA security model for the O-BIBE scheme is given as follows:

Definition 3.3 (Selective IND-CPA Security of O-BIBE)**.** The selective IND-CPA security (SE-IND-CPA) of O-BIBE is defined via the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{\mathit{O\text{-}BIBE},\mathcal{A}}(\lambda, B)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: The adversary $\mathcal{A}$ commits a challenge identity-label pair $(\mathsf{ID}^*, \mathsf{BL}^*)$.
2. **Setup**: The challenger $\mathcal{C}$ runs $(\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B)$ and initializes the experiment by running the following oracle:
   - $\mathsf{OInit}()$: This oracle initializes two empty lists, $L_{\text{key}} \leftarrow \emptyset$ and $L_{\text{del}} \leftarrow \emptyset$, which are used to track key exposure and delegation queries, respectively.

   Next, $\mathcal{C}$ provides $(\mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.
3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{OCreateDK}$, $\mathsf{ORevealDK}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRK}$ queries. $\mathcal{C}$ handles these queries as follows:
   - $\mathsf{OCreateDK}(S, \mathsf{BL})$: If there exists $(S, \mathsf{BL}, \cdot, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
     (a) It computes a decryption key DK by using the master key MK.
     (b) Next, it picks a unique key handle $\mathsf{hk} \in \{0,1\}^\lambda$, updates $L_{\text{key}} \leftarrow L_{\text{key}} \cup \{(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK})\}$, and responds with hk.
   - $\mathsf{ORevealDK}(\mathsf{hk})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise, it retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK}) \leftarrow L_{\text{key}}$ and responds with DK.
   - $\mathsf{ODelegate}(\mathsf{hk}, (\mathsf{CH}_i)_{i \in U})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
     (a) It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK})$ from L_{key} and computes $(\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i \in U}, \mathsf{BL})$.
     (b) It picks a unique delegation handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\text{del}} \leftarrow L_{\text{del}} \cup \{(\mathsf{hd}, \mathsf{RK})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U})$.
   - $\mathsf{ORevealRK}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\text{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd}, \mathsf{RK}) \leftarrow L_{\text{del}}$ and responds with RK.
4. **Challenge**: $\mathcal{C}$ computes a ciphertext pair $(\mathsf{CH}^*, \mathsf{CK}^*) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}^*, \mathsf{BL}^*)$. It sets $\mathsf{CK}^*_0 = \mathsf{CK}^*$ and $\mathsf{CK}^*_1 \xleftarrow{\$} \mathcal{K}$. It gives $(\mathsf{CH}^*, \mathsf{CK}^*_\mu)$ to $\mathcal{A}$.
5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. $\mathcal{C}$ handles these queries exactly as in the pre-challenge queries.

6. **Output**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. The experiment returns 1 if $\mu' = \mu$ and the following oracle constraints are satisfied:

   (a) **(Challenge Privacy)**: For the challenge label BL^* and any set S such that $\mathsf{ID}^* \in S$, the adversary $\mathcal{A}$ is restricted as follows:
   - $\mathcal{A}$ never queries $\mathsf{ORevealIDK}(\mathsf{hk})$ for the key handle hk associated with (S, BL^*).
   - $\mathcal{A}$ never queries $\mathsf{ORevealIRK}(\mathsf{hd})$ for any delegation handle hd that was generated from the key handle hk associated with (S, BL^*).

   (b) **(Label Coherence)**: For the challenge label BL^* and any set S such that $\mathsf{ID}^* \notin S$, the adversary $\mathcal{A}$'s queries are restricted as follows:
   - $\mathcal{A}$ is allowed to query $\mathsf{ORevealIDK}(\mathsf{hk})$ for at most one hk associated with (S, BL^*) such that $\mathsf{ID}^* \notin S$.
   - $\mathcal{A}$ is allowed to query $\mathsf{ORevealIRK}(\mathsf{hd})$ for at most one hd derived from (S, BL^*) such that $\mathsf{ID}^* \notin S$.
   - Furthermore, if $\mathcal{A}$ queries both $\mathsf{ORevealIDK}(\mathsf{hk})$ and $\mathsf{ORevealIRK}(\mathsf{hd})$ for the challenge label BL^* where $\mathsf{ID}^* \notin S$, then hd must be a delegation handle generated from that specific hk (i.e., they must share the same set S).

   Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{\textit{O-BIBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{\textit{O-BIBE},\mathcal{A}}(\lambda, B) = 1] - \frac{1}{2} \right|$ where the probability is taken over all internal randomness of the experiment. An O-BIBE scheme is said to be SE-IND-CPA secure if the advantage is negligible for all PPT adversaries.

Remark 3 (Relation to Standard BIBE Security Model). The SE-IND-CPA security model for a standard BIBE scheme can be naturally derived from our O-BIBE security model. Specifically, if the $\mathsf{ODelegate}$ and $\mathsf{ORevealIRK}$ oracles are removed while the $\mathsf{OCreateDK}$ and $\mathsf{ORevealIDK}$ oracles are combined into a single $\mathsf{OComputeKey}$ oracle, the resulting game becomes identical to the standard security model for BIBE [26]. This highlights that the O-BIBE model is a strict extension of the BIBE model, incorporating the additional adversarial capability of observing delegated tasks without compromising the underlying semantic security.

3.2 Construction

The proposed O-BIBE scheme is constructed upon the GWWW-BIBE scheme [26], with several strategic refinements designed to enhance practical efficiency and structural modularity.

First, considering a centralized trust model where a single authority manages the master secret, we consolidate the traditional Setup and KeyGen algorithms into a unified Setup algorithm. This streamlining reflects the operational reality of the system while simplifying the initialization and key management step. Second, to minimize the storage overhead of an encryptor, we separate the master public key (MPK) into two distinct components: a public encryption key (EK) and a public key (PK). This separation effectively reduces the size of the encryption parameters that users must maintain for ciphertext creation. Finally, we transition from the conventional PKE paradigm to a Key Encapsulation Mechanism (KEM) framework. Instead of encrypting messages directly, our O-BIBE scheme is designed to output a capsule key. This KEM-based approach is adopted not only to adhere to modern cryptographic standards but also to ensure that the O-BIBE scheme can serve as a robust, modular building block. Such a modular design facilitates the

seamless integration of the O-BIBE scheme into higher-level cryptographic protocols, such as the verifiable outsourcing systems proposed in this work.

The basic algorithms of our O-BIBE scheme from Setup to Decaps algorithms excluding the specialized outsourcing components are described as follows:

O-BIBE.Setup$(1^\lambda, 1^B)$**:** It first samples an asymmetric bilinear group $\mathsf{GD} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g}) \leftarrow \mathcal{G}(1^\lambda)$. It samples random exponents $\tau, \alpha, w, u', h' \in \mathbb{Z}_p$ sets $u = g^{u'}, h = g^{h'}, \hat{u} = \hat{g}^{u'}, \hat{h} = \hat{g}^{h'}$. It outputs the master key $\mathsf{MK} = \big(\mathsf{GD}, \alpha, w\big)$, the public encryption key EK, and the public key PK as

$$\mathsf{EK} = \Big(\mathsf{GD},\ g^w,\ g^{\tau w},\ u,\ h,\ \Omega = e(g,\hat{g})^\alpha\Big),\ \mathsf{PK} = \Big(\mathsf{GD},\ g^\tau,\ \hat{g}^\tau,\ \hat{g}^{\tau^2},\ \ldots,\ \hat{g}^{\tau^B},\ \hat{u},\ \hat{h}\Big).$$

O-BIBE.Encaps$(\mathsf{EK}, \mathsf{ID}, \mathsf{BL})$**:** It samples a random exponent $t \in \mathbb{Z}_p$ and outputs a ciphertext header

$$\mathsf{CH} = \Big(C_1 = g^t,\ C_2 = (g^{\tau w})^t (g^w)^{-\mathsf{ID}t} = g^{(\tau - \mathsf{ID})wt},\ C_3 = \big(u^{\mathsf{BL}} h\big)^t\Big)$$

and a capsule key $\mathsf{CK} = \Omega^t$.

O-BIBE.Digest(PK, S)**:** Let $S = \{\mathsf{ID}_i\}_{i \in U}$ where $U \subseteq [B]$. It defines the polynomial $F_S(x) = \prod_{\mathsf{ID}_i \in S}(x - \mathsf{ID}_i) = \sum_{k \in [0,|S|]} f_k x^k$ and outputs the digest

$$\mathsf{DIG} = \prod_{k \in [0,|S|]} \big(\hat{g}^{\tau^k}\big)^{f_k} = \hat{g}^{\sum_{k \in [0,|S|]} f_k \tau^k} = \hat{g}^{F_S(\tau)}.$$

O-BIBE.ComputeKey$(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL})$**:** Let $\mathsf{MK} = (\mathsf{GD}, \alpha, w)$. It samples random exponents $r \in \mathbb{Z}_p$ and $y \in \mathbb{Z}_p^*$, and outputs a decryption key

$$\mathsf{DK} = \Big(K_1 = \hat{g}^\alpha \big(\mathsf{DIG}\big)^{wy} \big(\hat{u}^{\mathsf{BL}} \hat{h}\big)^r,\ K_2 = \hat{g}^r,\ y\Big).$$

O-BIBE.Decaps$(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, \mathsf{ID}, \mathsf{BL})$**:** Let $\mathsf{DK} = (K_1, K_2, y)$ and $\mathsf{CH} = (C_1, C_2, C_3)$ where DK is associated with (S, BL), CH is associated with $(\mathsf{ID}, \mathsf{BL})$, and $\mathsf{ID} \in S$.

1. It defines the polynomial $F_{S \setminus \{\mathsf{ID}\}}(x) = \prod_{\mathsf{ID}_k \in S \setminus \{\mathsf{ID}\}}(x - \mathsf{ID}_k) = \sum_{k \in [0,|S|-1]} \tilde{f}_k x^k$ and computes the witness
$$\pi = \prod_{k \in [0,|S|-1]} \big(\hat{g}^{\tau^k}\big)^{\tilde{f}_k} = \hat{g}^{F_{S \setminus \{\mathsf{ID}\}}(\tau)}.$$
2. It outputs the capsule key $\mathsf{CK} = e(C_1, K_1) \cdot e(C_2, \pi)^{-y} \cdot e(C_3, K_2)^{-1}$.

The specialized algorithms of our O-BIBE scheme for outsourcing decryption Delegate, Transform, and Recover are defined based on the following principles. The core idea of our outsourcing design is centered on a re-randomization process performed by the outsourcing client. Specifically, the client re-randomizes the original decryption key DK to generate a recovery key RK and a transformation key TK, ensuring that the TK reveals no information about the DK to the outsourcing server. During the Transform phase, the outsourcing server performs the heavy decryption operations using the provided TK. However, due to the deliberate noise embedded within TK, the server is prevented from obtaining the correct capsule key. Finally, in the Recover phase, the client who possesses the secret RK efficiently removes the noise from the transformed results to derive the original capsule key. This mechanism ensures that the computational burden is successfully offloaded to the outsourcing server while maintaining strict data confidentiality.

The outsourcing algorithms of our O-BIBE scheme from Delegate to Recover algorithms are described as follows:

O-BIBE.Delegate$(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in U}, \mathsf{BL})$**:** Let $\mathsf{DK} = (K_1, K_2, y)$ and $\mathsf{CH}_i = (C_{i,1}, C_{i,2}, C_{i,3})$ for $i \in U$ where $U \subseteq [B]$.

1. It samples $y_1 \in \mathbb{Z}_p^*$ and $z_1, z_2 \in \mathbb{Z}_p$ uniformly at random and sets a recover key $\mathsf{RK} = (y_1, z_1)$. Then, it constructs a transformation key

$$\mathsf{TK} = \big(V_1 = K_1^{y_1} \cdot \hat{g}^{z_1} \cdot \big(\hat{u}^{\mathsf{BL}}\hat{h}\big)^{z_2},\ V_2 = K_2^{y_1} \cdot \hat{g}^{z_2},\ y_n = y \cdot y_1 \mod p\big).$$

2. It sets a work task $\mathsf{WT}_i = (D_{i,1} = C_{i,1}, D_{i,2} = C_{i,2}, D_{i,3} = C_{i,3})$ for each $i \in U$.
3. It outputs the recover key RK, the transformation key TK, and the work tasks $(\mathsf{WT}_i)_{i\in U}$.

O-BIBE.Transform$(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, (\mathsf{ID}_i)_{i\in U}, \mathsf{BL})$**:** Let $\mathsf{TK} = (V_1, V_2, y_n)$ and $\mathsf{WT}_i = (D_{i,1}, D_{i,2}, D_{i,3})$ where WT_i is associated with ID_i and BL for $i \in U$. It sets $S = \{\mathsf{ID}_i\}_{i\in U}$.

1. For each $i \in U$, it defines the polynomial $F_{S\setminus\{\mathsf{ID}_i\}}(x) = \prod_{\mathsf{ID}_k \in S\setminus\{\mathsf{ID}_i\}}(x - \mathsf{ID}_k) = \sum_{k\in[0,|S|-1]} \tilde{f}_{i,k} x^k$ and computes the witness

$$\pi_i = \prod_{k\in[0,|S|-1]} \big(\hat{g}^{\tau^k}\big)^{\tilde{f}_{i,k}} = \hat{g}^{F_{S\setminus\{\mathsf{ID}_i\}}(\tau)}.$$

2. For each $i \in U$, it constructs a transformed task $\mathsf{TT}_i = (E_{i,1}, E_{i,2})$ by computing

$$E_{i,1} = e(D_{i,1}, V_1) \cdot e(D_{i,2}, \pi_i)^{-y_n} \cdot e(D_{i,3}, V_2)^{-1} \quad \text{and} \quad E_{i,2} = e(D_{i,1}, \hat{g}).$$

3. It outputs the transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

O-BIBE.Recover$(\mathsf{RK}, (\mathsf{TT}_i)_{i\in U})$**:** Let $\mathsf{RK} = (y_1, z_1)$ and $\mathsf{TT}_i = (E_{i,1}, E_{i,2})$ for $i \in U$. For each $i \in U$, it recovers the capsule key $\mathsf{CK}_i = E_i$ by computing

$$E_i = \big(E_{i,1} \cdot E_{i,2}^{-z_1}\big)^{1/y_1}.$$

It outputs the capsule keys $(\mathsf{CK}_i)_{i\in U}$.

3.3 Correctness

Theorem 3.1. *The proposed O-BIBE scheme satisfies correctness.*

Proof. The correctness of decryption is easily derived from the property of the underlying BIBE scheme as follows:

$$\begin{aligned}
&e(C_{i,1}, K_1) \cdot e(C_{i,2}, \pi_i)^{-y} \cdot e(C_{i,3}, K_2)^{-1} \\
&= e(g^t, K_1) \cdot e\big(g^{(\tau - \mathsf{ID}_i)wt}, \hat{g}^{F_{S\setminus\{\mathsf{ID}_i\}}(\tau)}\big)^{-y} \cdot e\big((u^{\mathsf{BL}}h)^t, \hat{g}^r\big)^{-1} \\
&= e\big(g^t, \hat{g}^{\alpha}\big(\hat{g}^{F_S(\tau)}\big)^{wy}\big(\hat{u}^{\mathsf{BL}}\hat{h}\big)^r\big) \cdot e\big(g^t, \hat{g}^{F_S(\tau)\cdot wy}\big)^{-1} \cdot e\big(g^t, (\hat{u}^{\mathsf{BL}}\hat{h})^r\big)^{-1} \\
&= e(g^t, \hat{g}^{\alpha}) = \mathsf{CK}_i.
\end{aligned}$$

Next, we show the correctness of outsourced batch decryption. For each $i \in [B]$, let $\mathsf{TT}_i = (E_{i,1}, E_{i,2})$ be the output of the O-BIBE.Transform algorithm. By substituting the components of $\mathsf{TK} = (V_1, V_2, y_n)$ and $\mathsf{WT}_i = (D_{i,1}, D_{i,2}, D_{i,3})$, we obtain the following equation:

$$\begin{aligned} E_{i,1} &= e(D_{i,1}, V_1) \cdot e(D_{i,2}, \pi_i)^{-y_n} \cdot e(D_{i,3}, V_2)^{-1} \\ &= e(C_{i,1}, K_1^{y_1} \cdot \hat{g}^{z_1}) \cdot e(C_{i,2}, \pi_i)^{-yy_1} \cdot e(C_{i,3}, K_2^{y_1})^{-1} \\ &= \left(e(C_{i,1}, K_1) \cdot e(C_{i,2}, \pi_i)^{-y} \cdot e(C_{i,3}, K_2)^{-1}\right)^{y_1} \cdot e(C_{i,1}, \hat{g})^{z_1} \\ &= \mathsf{CK}_i^{y_1} \cdot e(g, \hat{g})^{t \cdot z_1}. \end{aligned}$$

By using the recover key $\mathsf{RK} = (y_1, z_1)$, the O-BIBE.Recover algorithm derives the following equation:

$$\left(E_{i,1} \cdot E_{i,2}^{-z_1}\right)^{1/y_1} = \left(\mathsf{CK}_i^{y_1} \cdot e(g, \hat{g})^{t \cdot z_1} \cdot e(g, \hat{g})^{-t \cdot z_1}\right)^{1/y_1} = \mathsf{CK}_i.$$

This completes the proof. □

3.4 Security Analysis

Theorem 3.2. *Let BIBE-KEM be the core scheme obtained by excluding the outsourcing algorithms (Delegate, Transform, Recover) from our proposed O-BIBE scheme. The BIBE-KEM scheme is SE-IND-CPA secure for any polynomial batch size B if the B-BDHEV assumption holds.*

Proof Sketch. The security of the BIBE-KEM scheme is reduction-linearly founded on the proof of the GWWW-BIBE scheme [26]. Although this BIBE-KEM scheme incorporates three structural modifications from the original GWWW-BIBE scheme, the validity of the simulation under the B-BDHEV assumption remains the same.

First, the consolidation of Setup and KeyGen into a single Setup algorithm is a functional simplification that does not alter the distribution of the system parameters or the master secret key. Since the underlying algebraic relations between the generated elements remain identical to those in GWWW-BIBE, this integration has no impact on the security reduction.

Second, regarding the parameter exposure, our BIBE-KEM scheme explicitly includes $\hat{u} = \hat{g}^{u'}$ and $\hat{h} = \hat{g}^{h'}$ in the public key PK. Under the B-BDHEV assumption, a simulator can construct these elements using the given term $\hat{g}^b$ without knowing the secret exponents u', h'. Since the simulator in the original GWWW-BIBE proof already utilizes such constructions to simulate public components, this modification does not compromise the validity of the simulation.

Third, concerning the transformation to KEM, our BIBE-KEM scheme adopts a key encapsulation mechanism (KEM) rather than a direct public-key encryption. The shift from direct encryption to a KEM does not alter the underlying algebraic security. The encapsulation process of BIBE-KEM follows the same bilinear map operations as the GWWW-BIBE encryption. Thus, the indistinguishability of the capsule key CK^* is directly reduced to the hardness of the B-BDHEV problem.

In summary, since the modified public key and the KEM structure can be perfectly simulated within the B-BDHEV framework, the SE-IND-CPA security of our BIBE-KEM scheme is guaranteed as in the original GWWW-BIBE scheme. □

Theorem 3.3. *The proposed O-BIBE scheme satisfies SE-IND-CPA security for any polynomial batch size B if the underlying BIBE-KEM scheme is SE-IND-CPA secure.*

Proof. To prove the SE-IND-CPA security of the O-BIBE scheme, we employ the hybrid game technique. We define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$, and $\mathbf{G}_3$ where the first game will be the original SE-IND-CPA security game and the last one will be a game in which an adversary has no advantage. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. The hybrid games are defined as follows:

Game $\mathbf{G}_0$. This game corresponds to the original SE-IND-CPA security experiment in Definition 3.3. Since the simulator follows the honest algorithms to respond to all queries and generate the challenge ciphertext, it follows by definition that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} = \mathsf{Adv}_{\mathit{O\text{-}BIBE},\mathcal{A}}^{\mathsf{SE\text{-}IND\text{-}CPA}}(\lambda).$$

Game $\mathbf{G}_1$. This game refines the oracle simulation of $\mathbf{G}_0$ by incorporating an index-guessing strategy combined with selective simulated sampling. The simulator $\mathcal{B}$ manages the queries as follows:

- In $\mathsf{OCreateDK}$, $\mathcal{B}$ generates the decryption key DK using MK if $\mathsf{BL} \neq \mathsf{BL}^*$. For the challenge label BL^*, $\mathcal{B}$ guesses a target index $\mathsf{gss\text{-}ctr}$ and only computes DK if the current query count matches this index (and $\mathsf{ID}^* \notin S$). In all other cases involving BL^*, $\mathcal{B}$ leaves the key entry empty ($\perp$) to facilitate lazy processing.
- In $\mathsf{ODelegate}$, $\mathcal{B}$ performs the original outsourcing algorithm if a valid DK exists in the list. Otherwise, it generates a simulated transformation key via random sampling to maintain functional consistency.
- In $\mathsf{ORevealIDK}$ and $\mathsf{ORevealIRK}$, $\mathcal{B}$ responds with the requested key only if it exists in the respective lists (L_{key} or L_{del}). If the key is absent (represented by $\perp$), the simulator terminates the game by aborting.

As established in Lemma 3.4, the view of $\mathcal{A}$ is identical to $\mathbf{G}_0$ provided that $\mathcal{B}$ does not abort. Since $\mathcal{B}$ successfully completes the simulation with probability at least $1/q$, the advantages satisfy

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} \geq \frac{1}{q} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}.$$

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$ except that the simulator $\mathcal{B}$ no longer utilizes the master key MK to generate decryption keys. Instead, $\mathcal{B}$ leverages the key extraction oracle $\mathsf{OComputeKey}(\cdot)$ provided by the underlying BIBE-KEM scheme. Specifically, in the $\mathsf{OCreateDK}$ oracle, $\mathcal{B}$ obtains the required decryption key DK by issuing a query to $\mathsf{OComputeKey}(S, \mathsf{BL})$ for the following cases: (i) when $\mathsf{BL} \neq \mathsf{BL}^*$, and (ii) when $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S$ and $\mathsf{key\text{-}ctr} = \mathsf{gss\text{-}ctr}$. Since the keys obtained through $\mathsf{OComputeKey}$ are identically distributed to those computed using MK, the adversary's view remains invariant. Consequently, the advantage of $\mathcal{A}$ remains unchanged:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}.$$

Game $\mathbf{G}_3$. This game is identical to $\mathbf{G}_2$ except for the generation of the challenge capsule key CK_{μ}^*. In the challenge round, the simulator provides $\mathcal{A}$ with a challenge ciphertext where CK_{μ}^* is replaced by a value chosen uniformly at random from the key space, regardless of the bit $\mu \in \{0,1\}$. The indistinguishability between $\mathbf{G}_2$ and $\mathbf{G}_3$ is reduced to the SE-IND-CPA security of the underlying BIBE-KEM scheme, which will be formally established in Lemma 3.6. Accordingly, we have

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}\right| \leq \mathsf{Adv}_{\mathit{BIBE},\mathcal{B}}^{\mathsf{SE\text{-}IND\text{-}CPA}}(\lambda).$$

Since the challenge capsule key is now purely random and independent of μ, the adversary $\mathcal{A}$ obtains no information regarding the challenge bit. Consequently, the advantage of $\mathcal{A}$ in $\mathbf{G}_3$ is exactly zero: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} = 0$.

By applying the advantage differences established in Lemmas 3.4 and 3.5, we obtain the following upper bound on the total advantage:

$$\begin{aligned}\mathsf{Adv}_{\mathit{O\text{-}BIBE},\mathcal{A}}^{\mathsf{SE\text{-}IND\text{-}CPA}}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} &\leq q \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} \leq q\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}\right| + q \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} \\ &\leq q \cdot \mathsf{Adv}_{\mathit{BIBE},\mathcal{B}}^{\mathsf{SE\text{-}IND\text{-}CPA}}(\lambda).\end{aligned}$$

This completes the proof. □

Lemma 3.4. *The games $\boldsymbol{G}_0$ and $\boldsymbol{G}_1$ are statistically indistinguishable conditioned on the event that the simulation does not abort. The probability that the simulation completes without an abort is at least $1/q$ where q denotes the maximum number of $\mathsf{OCreateDK}$ queries.*

Proof. In the game $\mathbf{G}_1$, we refine the oracle simulation logic to incorporate index-guessing and lazy sampling strategies. Specifically, for queries where $\mathsf{BL} \neq \mathsf{BL}^*$, the simulator $\mathcal{B}$ proceeds identically to the previous game. However, for queries satisfying $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S$, $\mathcal{B}$ employs an index-guessing method combined with lazy sampling to defer key generation until a valid reveal query is issued. Furthermore, for cases where $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \in S$, the simulator utilizes random sampling to provide functionally consistent but simulated responses, ensuring the simulation continues without access to the actual secret keys.

The detailed procedures for each oracle in $\mathbf{G}_1$ are given as follows:

- $\mathsf{OInit}()$: It initializes the simulation environment as follows:
  1. It chooses a random index $k^* \in [q]$ where q is the maximum number of $\mathsf{OCreateDK}$ queries.
  2. It sets the state variables $\mathsf{gss\text{-}ctr} \leftarrow k^*$ and $\mathsf{key\text{-}ctr} \leftarrow 0$.
  3. It initializes two empty lists $L_{\text{key}} \leftarrow \emptyset$ and $L_{\text{del}} \leftarrow \emptyset$.
- $\mathsf{OCreateDK}(S, \mathsf{BL})$: If there exists an entry $(S, \mathsf{BL}, \cdot, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. If $\mathsf{BL} \neq \mathsf{BL}^*$: It derives $\mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S)$ and computes $\mathsf{DK} \leftarrow \mathsf{ComputeKey}(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL})$.
  2. Else if $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S$: It increments $\mathsf{key\text{-}ctr} \leftarrow \mathsf{key\text{-}ctr} + 1$. If $\mathsf{key\text{-}ctr} = \mathsf{gss\text{-}ctr}$: It derives $\mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S)$ and computes $\mathsf{DK} \leftarrow \mathsf{ComputeKey}(\mathsf{MK}, \mathsf{DIG}, \mathsf{BL})$. Otherwise ($\mathsf{key\text{-}ctr} \neq \mathsf{gss\text{-}ctr}$): It sets $\mathsf{DK} = \perp$.
  3. Otherwise ($\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \in S$): It sets $\mathsf{DK} = \perp$.
  4. It picks a unique key handle $\mathsf{hk} \in \{0,1\}^\lambda$, updates $L_{\text{key}} \leftarrow L_{\text{key}} \cup \{(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK})\}$, and responds with hk.
- $\mathsf{ORevealDK}(\mathsf{hk})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK})$ from L_{key}.
  2. If $\mathsf{DK} \neq \perp$: It responds with DK. Otherwise ($\mathsf{DK} = \perp$): The simulator **aborts**.
- $\mathsf{ODelegate}(\mathsf{hk}, (\mathsf{CH}_i)_{i \in U})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DK})$ from L_{key}

2. If $\mathsf{DK} \neq \perp$: It computes $(\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i \in U}, \mathsf{BL})$.
3. Otherwise ($\mathsf{DK} = \perp$): It samples $(V_1, V_2, y_n) \in \hat{\mathbb{G}}^2 \times \mathbb{Z}_p^*$ uniformly at random. It sets $\mathsf{RK} = \perp$ and $\mathsf{TK} = (V_1, V_2, y_n)$. It derives $(\mathsf{WT}_i)_{i \in U}$ from $(\mathsf{CH}_i)_{i \in U}$ accordingly.
4. It picks a unique delegation handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\mathrm{del}} \leftarrow L_{\mathrm{del}} \cup \{(\mathsf{hd}, \mathsf{RK})\}$ and responds with $(\mathsf{TK}, (\mathsf{WT}_i)_{i \in U})$.

- $\mathsf{ORevealRK}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\mathrm{del}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(\mathsf{hd}, \mathsf{RK})$ from L_{del}.
  2. If $\mathsf{RK} \neq \perp$: It responds with RK. Otherwise: The simulator **aborts**.

To establish the statistical indistinguishability between $\mathbf{G}_0$ and $\mathbf{G}_1$, we first consider the probability that the simulator $\mathcal{B}$ completes the game without an abort. Since $\mathcal{B}$ chooses the target index $k^* \in [q]$ uniformly at random, and the simulation only proceeds if the adversary's reveal query corresponds to this guessed index, the probability of the non-abort event is at least $1/q$. Conditioned on the event that $\mathcal{B}$ does not abort, the view of the adversary $\mathcal{A}$ remains statistically identical to the real game. Through $\mathsf{ODelegate}$ queries, $\mathcal{A}$ obtains the transformation key $\mathsf{TK} = (V_1, V_2, y_n)$ and the associated work tasks. The distribution of this view is characterized by the linear mapping from the internal randomness $\mathbf{v}_{in} = (y_1, z_1, z_2)^\top$ to the observable components $\mathbf{v}_{out} = (\log_{\hat{g}}(V_1), \log_{\hat{g}}(V_2), y_n)^\top$:

$$\begin{pmatrix} \log_{\hat{g}}(V_1) \\ \log_{\hat{g}}(V_2) \\ y_n \end{pmatrix} = \begin{pmatrix} \log_{\hat{g}}(K_1) & 1 & \log_{\hat{g}}(\hat{u}^{\mathsf{BL}}\hat{h}) \\ \log_{\hat{g}}(K_2) & 0 & 1 \\ y & 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ z_1 \\ z_2 \end{pmatrix}.$$

Since the determinant of the transformation matrix $\mathbf{A}$ is $\det(\mathbf{A}) = -y \pmod p$, and $y \in \mathbb{Z}_p^*$, the mapping $\mathbf{A}$ constitutes a bijection. In $\mathbf{G}_1$, although the simulator $\mathcal{B}$ performs direct sampling of the components (V_1, V_2, y_n) only for specific cases within the $\mathsf{ODelegate}$ simulation—namely when $\mathsf{BL} = \mathsf{BL}^*$—the overall statistical distribution of the adversary's view remains invariant. Due to the one-to-one correspondence established by the linear mapping $\mathbf{A}$, sampling the observable components (V_1, V_2, y_n) directly from their respective groups is statistically equivalent to sampling the internal randomness (y_1, z_1, z_2) and computing the responses. Consequently, even with this selective sampling strategy, the distribution of the adversary's view in $\mathbf{G}_1$ is identical to that in $\mathbf{G}_0$, provided the simulation does not abort. Consequently, by accounting for the success probability of the index-guessing, the advantages of the adversary in these games satisfy the following relationship: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} \geq \frac{1}{q} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}$. This completes the proof. □

Lemma 3.5. *The games $\mathbf{G}_1$ and $\mathbf{G}_2$ are identical.*

Proof. The only difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ lies in the method of generating the decryption keys (DK). In $\mathbf{G}_1$, $\mathcal{B}$ computes DK directly using MK, whereas in $\mathbf{G}_2$, it invokes the key computation oracle $\mathsf{OComputeKey}$ of the BIBE-KEM scheme. According to the correctness of the BIBE-KEM key generation, the output of $\mathsf{OComputeKey}$ for a given (S, BL) is equivalent to the output of the $\mathsf{ComputeKey}$ algorithm using MK. Consequently, the distribution of all oracle responses remains invariant between the two games, making them perfectly indistinguishable from the adversary's perspective. □

Lemma 3.6. *The games $\mathbf{G}_2$ and $\mathbf{G}_3$ are computationally indistinguishable, assuming that the underlying BIBE-KEM scheme is SE-IND-CPA secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the security of the SE-IND-CPA game of O-BIBE with a non-negligible advantage. We construct a simulator $\mathcal{B}$ that breaks the underlying BIBE-KEM scheme by invoking $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ commits a challenge identity-label pair $(\mathsf{ID}^*, \mathsf{BL}^*)$. $\mathcal{B}$ forwards $(\mathsf{ID}^*, \mathsf{BL}^*)$ to the BIBE-KEM challenger and receives $(\mathsf{EK}_{BIBE}, \mathsf{PK}_{BIBE})$.

Setup: The simulator $\mathcal{B}$ initializes the simulation environment by invoking the $\mathsf{OInit}(\cdot)$ oracle as defined in $\mathbf{G}_2$. This procedure establishes the initial state, including the selection of the target index $k^* \in [q]$ and the initialization of the lists L_{key} and L_{del} to empty sets. Furthermore, $\mathcal{B}$ sets the O-BIBE public parameters using the underlying BIBE-KEM instance by setting $\mathsf{EK} \leftarrow \mathsf{EK}_{\mathsf{BIBE}}$ and $\mathsf{PK} \leftarrow \mathsf{PK}_{\mathsf{BIBE}}$. Finally, $\mathcal{B}$ provides $(\mathsf{EK}, \mathsf{PK})$ to the adversary $\mathcal{A}$.

Pre-challenge Queries: The simulator $\mathcal{B}$ responds to the adversary's queries by following the exact same procedures defined in $\mathbf{G}_2$. Specifically, for all queries requiring the computation of decryption keys (DK) in the $\mathsf{OCreateDK}$ simulation in $\mathbf{G}_2$, $\mathcal{B}$ does not utilize the master key MK (which it does not possess). Instead, it obtains the required decryption keys by issuing key computation queries to the underlying BIBE-KEM $\mathsf{OComputeKey}$ oracle. Since the distribution of the oracle responses is identical to that in the previous game, the simulation remains consistent and indistinguishable from the perspective of $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives the challenge ciphertext $\mathsf{CH}^*_{BIBE}, \mathsf{CK}^*_{BIBE}$ from the BIBE-KEM challenger. It sets $\mathsf{CH}^* = \mathsf{CH}^*_{BIBE}$ and $\mathsf{CK}^* = \mathsf{CK}^*_{BIBE}$, and gives the pair $(\mathsf{CH}^*, \mathsf{CK}^*)$ to $\mathcal{A}$

Post-challenge Queries: $\mathcal{B}$ responds to these queries in the same manner as the pre-challenge queries, subject to the standard security game restrictions.

Output: $\mathcal{A}$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'.

The advantage of $\mathcal{B}$ in the BIBE-KEM security game is derived as follows: If the BIBE-KEM challenger's bit is $b = 0$, the adversary $\mathcal{A}$'s view is identical to $\mathbf{G}_2$. If the BIBE-KEM challenger's bit is $b = 1$, the adversary $\mathcal{A}$'s view is identical to $\mathbf{G}_3$. Therefore, $\mathcal{B}$'s advantage in winning the BIBE-KEM game is exactly equal to $\mathcal{A}$'s advantage in distinguishing $\mathbf{G}_2$ from $\mathbf{G}_3$. □

3.5 Efficiency Analysis

The efficiency of the basic algorithms in our O-BIBE scheme is consistent with the underlying GWWW-BIBE scheme. In the Setup algorithm, the public encryption key EK consists of $4|\mathbb{G}|$ and $1|\mathbb{G}_T|$, while the public key PK comprises $1|\mathbb{G}|$ and $(B+2)|\hat{\mathbb{G}}|$. The encapsulation algorithm (Encaps) requires 5 exponentiations in $\mathbb{G}$ and 1 exponentiation in $\mathbb{G}_T$, producing a ciphertext header of $3|\mathbb{G}|$ and a capsule key of $1|\mathbb{G}_T|$. The Digest algorithm involves $B\log B$ field multiplications via DFT and $(B+1)$ group exponentiations to output a digest of $1|\hat{\mathbb{G}}|$. For key management, the $\mathsf{ComputeKey}$ algorithm performs 5 exponentiations in $\hat{\mathbb{G}}$ to yield a decryption key $\mathsf{DK} \in \hat{\mathbb{G}}^2 \times \mathbb{Z}_p$. The Decaps algorithm for a single ciphertext header requires $B\log B$ field multiplications via DFT, B group exponentiations, and 3 pairing operations.

Regarding the outsourcing computation, the $\mathsf{Delegate}$ algorithm requires 6 exponentiations in $\hat{\mathbb{G}}$, outputting a recovery key $\mathsf{RK} \in \mathbb{Z}_p^2$ and a transformation key $\mathsf{TK} \in \hat{\mathbb{G}}^2 \times \mathbb{Z}_p$. To process a batch of B ciphertexts, the $\mathsf{Transform}$ algorithm on the server side executes approximately $B^2 \log B$ field multiplications, B^2 group exponentiations, and $4B$ pairings, resulting in transformed tasks of $2B|\mathbb{G}_T|$. Finally, the $\mathsf{Recover}$ algorithm on the client side only requires $2B$ exponentiations in $\mathbb{G}_T$ to output B capsule keys. The comparative analysis highlights a significant reduction in the client's computational overhead. For a batch of B ciphertexts, a standard local decryption would require $B^2 \log B$ multiplications, B^2 exponentiations, and $3B$ pairings. In contrast, by employing our outsourcing mechanism, the client's burden is reduced to only 6 exponentiations

in $\hat{\mathbb{G}}$ and $2B$ exponentiations in $\mathbb{G}_T$, with zero pairing operations. This effectively shifts the computational complexity from $O(B^2 \log B)$ to $O(B)$, demonstrating the practical scalability of our O-BIBE scheme for resource-constrained devices.

3.6 Discussions

Threshold O-BIBE Extension. While our proposed O-BIBE scheme utilizes a decryption key issued by a single centralized trusted server for batch decryption or delegation, it can be naturally extended to a threshold-based setting to distribute this trust. In this extended framework, denoted as outsourced threshold BIBE (O-TBIBE), batch decryption and delegation tasks are executed by combining decryption key shares provided by a threshold number of nodes within a decryption committee. We instantiate this O-TBIBE scheme by incorporating outsourcing functionalities into the GWWW-TBIBE scheme, which has been proven in the GGM. While the overall architecture of this O-TBIBE scheme closely mirrors that of our O-BIBE scheme, it features a more simplified decryption key structure due to its reliance on the GGM. A comprehensive presentation of the O-TBIBE syntax, security model, and concrete construction is provided in Appendix A.

4 Verifiable Outsourced Batched Encryption

In this section, we present the formal framework and construction of our VOBE scheme. We begin by defining the formal syntax and security models for VOBE, which captures the necessary requirements for both confidentiality and verifiability in an outsourced environment. Building upon the O-BIBE scheme of the previous section, we propose a concrete VOBE scheme that achieves efficient batching and secure delegation with verifiable results. We then provide a rigorous security analysis, demonstrating that the security of our VOBE scheme can be reduced to the underlying cryptographic primitives and the security of the O-BIBE scheme.

4.1 Definition

BE enables the simultaneous decryption of multiple ciphertexts rather than processing a single ciphertext at a time. We extend this BE paradigm to the concept of VOBE, which supports both the outsourcing of heavy decryption operations and the verification of the computed results.

In a VOBE scheme, a trusted central authority initializes the system by generating a master key, a public encryption key, and a general public key through a setup algorithm. Individual users then generate ciphertexts (CT) for messages associated with a specific batch label BL. A decryptor, possessed with the master key, can then locally decrypt a collection of ciphertexts sharing the same BL to recover the original messages simultaneously. To further alleviate the computational burden of the decryption process, a VOBE scheme provides a mechanism to offload the heavy decryption operations to a cloud server. By employing a delegation algorithm, the decryptor produces a recovery token (RT), a transformation key (TK), and a set of work tasks (WT). The TK and WT are then transmitted to the cloud server, which executes a transformation algorithm to convert the work tasks into transformed tasks (TT). These results are returned to the decryptor, who finally applies the RT to efficiently recover all original messages. This framework ensures that the decryptor can obtain the messages while verifying that the cloud server's computations were performed correctly.

Following this operational flow, the formal syntax of the VOBE scheme is defined as follows:

Definition 4.1 (VOBE)**.** A verifiable outsourced batched encryption (VOBE) scheme with a maximum batch size $B \in \mathbb{N}$, message space of $\mathcal{M}$, and batch label space of $\mathcal{L}$ consists of the following six algorithms. Let $\mathcal{M}_{\perp\!\!\!\perp} = \mathcal{M} \cup \{\perp, \perp\!\!\!\perp\}$ denote the extended message space, which includes the verification failure symbol $\perp$ (resulting from invalid input or dummy padding) and the transformation failure symbol $\perp\!\!\!\perp$.

$\mathsf{Setup}(1^\lambda, 1^B) \to (\mathsf{MK}, \mathsf{EK}, \mathsf{PK})$: The setup algorithm takes as input the security parameter λ and the maximum batch size B. It outputs a master key MK, a public encryption key EK, and a public key PK.

$\mathsf{Encrypt}(\mathsf{EK}, M, \mathsf{BL}) \to \mathsf{CT}$: The encryption algorithm takes as input the public encryption key EK, a message $M \in \mathcal{M}$, and a batch label $\mathsf{BL} \in \mathcal{L}$. It outputs a ciphertext CT.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \to (M_i)_{i\in[B]}$: The (deterministic) decryption algorithm takes as input the public key PK, the master key MK, ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$, and a batch label BL. It outputs the corresponding messages $(M_i)_{i\in[B]} \in \mathcal{M}_{\perp\!\!\!\perp}^B$.

$\mathsf{Delegate}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \to (\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$: The delegation algorithm takes as input the public key PK, the master key MK, ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$, and a batch label BL. It outputs a recovery token RT, a transformation key TK, and work tasks $(\mathsf{WT}_i)_{i\in U}$ where $U \subseteq [B]$.

$\mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{TT}_i)_{i\in U}$: The transformation algorithm takes as input the public key PK, the transformation key TK, the work tasks $(\mathsf{WT}_i)_{i\in U}$, and a batch label BL where $U \subseteq [B]$. It outputs transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

$\mathsf{Recover}(\mathsf{RT}, (\mathsf{TT}_i)_{i\in U}) \to (M_i)_{i\in[B]}$: The (deterministic) recovery algorithm takes as input the recovery token RT and the transformed tasks $(\mathsf{TT}_i)_{i\in U}$ where $U \subseteq [B]$. It outputs the recovered messages $(M_i)_{i\in[B]} \in \mathcal{M}_{\perp\!\!\!\perp}^B$.

Definition 4.2 (Correctness of VOBE)**.** A VOBE scheme is correct if for any security parameter λ and any batch size $B \le \mathrm{poly}(\lambda)$, the following two conditions hold:

- Correctness of Batch Decryption: For any messages $(M_i)_{i\in[B]} \in \mathcal{M}^B$ and any batch label $\mathsf{BL} \in \mathcal{L}$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Decrypt}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) & (\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B) \\ = (M_i)_{i\in[B]} & \forall i \in [B] : \mathsf{CT}_i \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M_i, \mathsf{BL}) \end{array}\right] = 1.$$

- Correctness of Outsourced Batch Decryption: For any messages $(M_i)_{i\in[B]} \in \mathcal{M}^B$ and any batch label $\mathsf{BL} \in \mathcal{L}$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Recover}(\mathsf{RT}, (\mathsf{TT}_i)_{i\in U}) & (\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B) \\ = (M_i)_{i\in[B]} & \forall i \in [B] : \mathsf{CT}_i \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M_i, \mathsf{BL}) \\ & (\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \\ & (\mathsf{TT}_i)_{i\in[B]} \leftarrow \mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, \mathsf{BL}) \end{array}\right] = 1.$$

We define the IND-CCA security model for our VOBE scheme by extending the foundational BE security model introduced by Agarwal et al. [1], incorporating additional outsourcing oracles to account for the delegated decryption environment. In the selective IND-CCA model, an adversary first commits to a target batch label BL^*, after which an challenger provides the public encryption key EK and the public key PK.

The adversary is then allowed to adaptively query the $\mathsf{ODecrypt}$ oracle for batch decryption services, as well as the $\mathsf{ODelegate}$ and $\mathsf{ORevealIRT}$ oracles to observe the delegation and recovery processes. In the challenge phase, the adversary submits two challenge messages M_0^* and M_1^*, and receives a challenge ciphertext CT^* that encrypts M_μ^* under the target label BL^* for a bit $\mu \in \{0,1\}$. Following the challenge, the adversary may continue to issue oracle queries. Finally, the adversary outputs a guess μ' for the challenge bit μ and wins if $\mu' = \mu$. To prevent trivial wins, standard restrictions are imposed on the adversary's queries: specifically, the adversary cannot query the $\mathsf{ODecrypt}$ oracle with a batch containing CT^*, nor can it query the $\mathsf{ORevealIRT}$ oracle for the recovery of the challenge ciphertext after the challenge phase has occurred.

A more rigorous definition of the selective IND-CCA security model for the VOBE scheme is given as follows:

Definition 4.3 (Selective IND-CCA Security of VOBE)**.** The Selective IND-CCA security (SE-IND-CCA) of VOBE is defined via the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}IND\text{-}CCA}}_{VOBE,\mathcal{A}}(\lambda, B)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: $\mathcal{A}$ commits the challenge batch label BL^*.
2. **Setup**: $\mathcal{C}$ runs $(\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B)$ and gives $(\mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.
3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{ODecrypt}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealIRT}$ queries. $\mathcal{C}$ handles these queries as follows:
   - $\mathsf{ODecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It runs $(M_i)_{i\in[B]} \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$ and responds with $(M_i)_{i\in[B]}$.
   - $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It computes $(\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$. It picks a fresh handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\mathrm{del}} \leftarrow L_{\mathrm{del}} \cup \{(\mathsf{hd}, \mathsf{RT})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.
   - $\mathsf{ORevealIRT}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\mathrm{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd}, \mathsf{RT}) \leftarrow L_{\mathrm{del}}$ and responds with RT.
4. **Challenge**: $\mathcal{A}$ submits two messages $M_0^*, M_1^* \in \mathcal{M}$. $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$, computes a challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M_\mu^*, \mathsf{BL}^*)$, and gives CT^* to $\mathcal{A}$.
5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. $\mathcal{C}$ handles these queries exactly as in the pre-challenge queries.
6. **Output**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. The experiment returns 1 if $\mu' = \mu$ and the following oracle constraints are satisfied:
   (a) **(Challenge Privacy)**: For the challenge label BL^* and the challenge ciphertext CT^*, the adversary $\mathcal{A}$ is restricted as follows:
      - $\mathcal{A}$ never queries $\mathsf{ODecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.
      - $\mathcal{A}$ never queries $\mathsf{ORevealIRT}(\mathsf{hd})$ for the handle hd returned by $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.

   (b) **(Label Distinctness)**: To ensure a consistent mapping between batch labels and ciphertext sets, the adversary $\mathcal{A}$ is restricted as follows:
      - $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODecrypt}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of decryption results for each unique label.

- $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODelegate}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of delegation tasks for each unique label.
- For any batch label BL, $\mathcal{A}$ must associate it with only one specific set of ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$ across both $\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$ queries. Any query involving a previously used label BL with a different set of ciphertexts $(\mathsf{CT}_j)_{j\in[B]}$ is strictly prohibited.

Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CCA}}_{VOBE,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{SE\text{-}IND\text{-}CCA}}_{VOBE,\mathcal{A}}(\lambda,B)=1]-\frac{1}{2}\right|$ where the probability is taken over all the randomness of the experiment. A VOBE scheme is said to be SE-IND-CCA secure if the advantage is negligible for all PPT adversaries.

Remark 4 (Label Distinctness and Blockchain). The label distinctness constraints imposed on the $\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$ oracles are strictly grounded in the operational logic of immutable ledgers. In a blockchain environment, a batch label BL functions as a unique identifier for a specific epoch, round, or transaction set. Once a label and its corresponding ciphertext set $(\mathsf{CT}_i)_{i\in[B]}$ are recorded on the ledger, they are finalized and persistently bound through the system's consensus mechanism. Consequently, any attempt to submit multiple distinct ciphertext sets for a single finalized label would be inherently rejected by the protocol's finality rules. Our security model accurately reflects this practical constraint by requiring that all batch labels submitted to the oracles be distinct. By ensuring that the adversary observes at most one set of decryption or delegation results for each unique label, we maintain logical consistency throughout the security game and bridge the gap between theoretical modeling and the immutability of real-world blockchain systems.

Remark 5 (Mempool Privacy). Our security model effectively captures the notion of mempool privacy by carefully calibrating oracle restrictions based on the inclusion of the challenge ciphertext CT^*. If CT^* is included in a ciphertext set $(\mathsf{CT}_i)_{i\in[B]}$ associated with the challenge label BL^*, direct decryption via $\mathsf{ODecrypt}$ is prohibited to prevent trivial wins. However, $\mathcal{A}$ is still permitted to issue $\mathsf{ODelegate}$ queries for the same batch, reflecting the realistic scenario where the challenge ciphertext is processed within a mempool alongside other ciphertexts. Furthermore, if CT^* is not part of the batch for BL^*, all oracles—including $\mathsf{ODecrypt}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRT}$—remain fully accessible to the adversary. This granular control ensures that the IND-CCA security of VOBE inherently guarantees mempool privacy, as the adversary cannot distinguish the challenge ciphertext even when it is batched with other arbitrary ciphertexts.

By adapting the IND-CCA security model of VOBE, we define the Verifiability (VER) model for the VOBE scheme, which ensures that it is computationally infeasible for a malicious outsourcing server to alter the outsourced results without detection. In the selective VER model, the adversary first commits to a target batch label BL^*, after which it receives the public encryption key EK and the public key PK. The adversary is granted access to the $\mathsf{ODecrypt}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRT}$ oracles, identical to the IND-CCA model. During the challenge phase, the challenger selects a random message M^*, generates a challenge ciphertext CT^* for M^* under the label BL^*, and provides CT^* to the adversary. The adversary may then continue to issue additional oracle queries. To conclude the game, the adversary outputs a target ciphertext set containing CT^*, a delegation handle obtained from the $\mathsf{ODelegate}$ oracle, and a set of forged transformed tasks $\widetilde{\mathsf{TT}}$. The adversary wins the game if the recovery process applied to the forged $\widetilde{\mathsf{TT}}$ yields a valid message $\widetilde{M}$ such that $\widetilde{M}\in\mathcal{M}$ and $\widetilde{M}\neq M^*$. To prevent trivial attacks, similar restrictions as in the IND-CCA model are imposed; specifically, the adversary is prohibited from querying the $\mathsf{ODecrypt}$ or $\mathsf{ORevealRT}$ oracles for the challenge ciphertext CT^* in a way that would expose the intended recovery.

A more rigorous definition of the selective VER security model for the VOBE scheme is as follows:

Definition 4.4 (Selective Verifiability of VOBE)**.** The selective verifiability (SE-VER) of a VOBE scheme is defined as the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}VER}}_{VOBE,\mathcal{A}}(\lambda, B)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: $\mathcal{A}$ commits the challenge batch label BL^*.
2. **Setup**: $\mathcal{C}$ runs $(\mathsf{MK}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B)$ and gives $(\mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.
3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{ODecrypt}, \mathsf{ODelegate}$, and $\mathsf{ORevealRT}$ queries. $\mathcal{C}$ handles these queries as follows:
   - $\mathsf{ODecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It runs $(M_i)_{i\in[B]} \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$ and responds with $(M_i)_{i\in[B]}$.
   - $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It computes $(\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$. It picks a fresh handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\mathrm{del}} \leftarrow L_{\mathrm{del}} \cup \{(\mathsf{hd}, \mathsf{RT})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.
   - $\mathsf{ORevealRT}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\mathrm{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd}, \mathsf{RT}) \leftarrow L_{\mathrm{del}}$ and responds with RT.
4. **Challenge**: $\mathcal{C}$ selects a random message $M^* \in \mathcal{M}$, computes a challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M^*, \mathsf{BL}^*)$, and gives CT^* to $\mathcal{A}$.
5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. $\mathcal{C}$ handles these queries exactly as in the pre-challenge queries.
6. **Output**: $\mathcal{A}$ outputs target ciphertexts $(\widetilde{\mathsf{CT}}_i)_{i\in[B]}$ such that $\widetilde{\mathsf{CT}}_k = \mathsf{CT}^*$ for an index $k \in [B]$, a target handle hd^* returned by $\mathsf{ODelegate}((\widetilde{\mathsf{CT}}_i)_{i\in[B]}, \mathsf{BL}^*)$, and target transformed tasks $(\widetilde{\mathsf{TT}}_i)_{i\in U}$. Next, $\mathcal{C}$ retrieves $(\mathsf{hd}^*, \mathsf{RT}^*) \leftarrow L_{\mathrm{del}}$ and recovers $(\widetilde{M}_i)_{i\in[B]} \leftarrow \mathsf{Recover}(\mathsf{RT}^*, (\widetilde{\mathsf{TT}}_i)_{i\in U})$.

   The experiment returns 1 if $(\widetilde{M}_k \in \mathcal{M} \wedge \widetilde{M}_k \neq M^*)$ for the index k and the following oracle constraints are satisfied:

   (a) **(Challenge Privacy)**: For the challenge label BL^* and the challenge ciphertext CT^*, the adversary $\mathcal{A}$ is restricted as follows:
   - $\mathcal{A}$ never queries $\mathsf{ODecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.
   - $\mathcal{A}$ never queries $\mathsf{ORevealRT}(\mathsf{hd})$ for the handle hd returned by $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.

   (b) **(Label Distinctness)**: To ensure a consistent mapping between batch labels and ciphertext sets, the adversary $\mathcal{A}$ is restricted as follows:
   - $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODecrypt}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of decryption results for each unique label.
   - $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODelegate}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of delegation tasks for each unique label.
   - For any batch label BL, $\mathcal{A}$ must associate it with only one specific set of ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$ across both $\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$ queries. Any query involving a previously used label BL with a different set of ciphertexts $(\mathsf{CT}_j)_{j\in[B]}$ is strictly prohibited.

Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}VER}}_{VOBE,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{SE\text{-}VER}}_{VOBE,\mathcal{A}}(\lambda, B) = 1]$ where the probability is taken over all the randomness of the game. A VOBE scheme is said to be SE-VER secure if the advantage is negligible for all PPT adversaries.

4.2 Construction

To construct our VOBE scheme, we transform the outsourcing-enabled O-BIBE scheme into a CCA-secure framework. For this CCA transformation, we employ the CHK conversion method [11], which utilizes an OTS scheme to ensure ciphertext integrity. By setting the verification key (VK) of the OTS as the identity (ID) for the underlying O-BIBE scheme, we establish a robust cryptographic link between the signature and the identity-based structure. This CHK approach is widely recognized for its efficacy in converting IBE into CCA-secure PKE and extending BIBE into CCA-secure BE. The core functionality of decryption outsourcing is inherited from the algorithms of the O-BIBE scheme. To achieve verifiability of the outsourced computation, we incorporate a PRF to embed a verification tag of the session key within the ciphertext. During the Recover phase, the client derives the capsule key from the transformed tasks provided by the cloud server and uses it as a PRF key to reconstruct the tag. The correctness of the cloud server's operation is then verified by comparing this reconstructed tag with the original tag stored in the ciphertext. Furthermore, we ensure the freshness and integrity of the session key by deriving both the symmetric encryption key and the verification tag from the PRF output, using the VK and the batch label BL as additional inputs. This design ensures that the ciphertext's validity depends not only on the O-BIBE session key but also on the specific VK and BL, thereby preventing key re-usage or substitution attacks.

The detailed specifications of our VOBE scheme are provided as follows:

VOBE.Setup$(1^\lambda, 1^B)$**:** It generates $(\mathsf{MK}_{OB}, \mathsf{EK}_{OB}, \mathsf{PK}_{OB}) \leftarrow \mathsf{O\text{-}BIBE.Setup}(1^\lambda, 1^B)$ and selects a hash function: $\mathsf{H}_s : \{0,1\}^* \to \mathcal{I}$ from $\mathcal{H}$. It outputs a master key $\mathsf{MK} = \mathsf{MK}_{OB}$, a public encryption key $\mathsf{EK} = (\mathsf{EK}_{OB}, \mathsf{H}_s)$, and a public key $\mathsf{PK} = \big(\mathsf{PK}_{OB}, \mathsf{H}_s\big)$.

VOBE.Encrypt$(\mathsf{EK}, M, \mathsf{BL})$**:** Let $\mathsf{EK} = (\mathsf{EK}_{OB}, \mathsf{H}_s)$.

1. It generates an OTS key pair $(\mathsf{SK}_{OTS}, \mathsf{VK}_{OTS}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$ and derives $\mathsf{ID} = \mathsf{H}_s(\mathsf{VK}_{OTS})$.
2. It computes a ciphertext header-key pair $(\mathsf{CH}_{OB}, \mathsf{CK}_{OB}) \leftarrow \mathsf{O\text{-}BIBE.Encaps}(\mathsf{EK}_{OB}, \mathsf{ID}, \mathsf{BL})$ and derives the key-tag value $(K \| \eta) = \mathsf{PRF}(\mathsf{CK}_{OB}, \mathsf{ID} \| \mathsf{BL})$.
3. Then, it encrypts the message as $\mathsf{CT}_{SKE} \leftarrow \mathsf{SKE.Encrypt}(K, M)$ and generates a signature $\sigma \leftarrow \mathsf{OTS.Sign}(\mathsf{SK}_{OTS}, \mathsf{BL} \| \mathsf{CH}_{OB} \| \mathsf{CT}_{SKE} \| \eta)$.
4. Finally, it outputs a ciphertext $\mathsf{CT} = \big(\mathsf{VK}_{OTS}, \mathsf{CH}_{OB}, \mathsf{CT}_{SKE}, \eta, \sigma\big)$.

VOBE.Decrypt$(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OB}, \mathsf{H}_s)$, $\mathsf{MK} = \mathsf{MK}_{OB}$ and $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for $i \in [B]$.

1. It initializes $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
   (a) It checks that $1 \stackrel{?}{=} \mathsf{OTS.Verify}(\mathsf{VK}_{i,OTS}, \mathsf{BL} \| \mathsf{CH}_{i,OB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$. If the check fails, it sets $M_i = \perp$ and continues to the next index.
   (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
2. It derives $\mathsf{DIG} \leftarrow \mathsf{O\text{-}BIBE.Digest}(\mathsf{PK}_{OB}, S)$ and computes $\mathsf{DK}_{OB} \leftarrow \mathsf{O\text{-}BIBE.ComputeKey}(\mathsf{PK}_{OB}, \mathsf{MK}_{OB}, \mathsf{DIG}, \mathsf{BL})$.

3. For each $i \in U$, it recovers $\mathsf{CK}_{i,OB} \leftarrow \mathsf{O\text{-}BIBE.Decaps}(\mathsf{PK}_{OB}, \mathsf{DK}_{OB}, S, \mathsf{CH}_{i,OB}, \mathsf{ID}_i, \mathsf{BL})$, from which it derives $(K_i \| \eta_i) = \mathsf{PRF}(\mathsf{CK}_{i,OB}, \mathsf{ID}_i \| \mathsf{BL})$ and decrypts the message as $M_i \leftarrow \mathsf{SKE.Decrypt}(K_i, \mathsf{CT}_{i,SKE})$.
4. Finally, it outputs the corresponding messages $(M_i)_{i\in[B]}$.

VOBE.Delegate$(\mathsf{PK}, \mathsf{MK}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OB}, \mathsf{H}_s)$, $\mathsf{MK} = \mathsf{MK}_{OB}$ and $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for $i \in [B]$.

1. It initializes $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
   (a) It checks that $1 \stackrel{?}{=} \mathsf{OTS.Verify}(\mathsf{VK}_{i,OTS}, \mathsf{BL} \| \mathsf{CH}_{i,OB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$. If the check fails, it continues to the next index.
   (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
2. It derives $\mathsf{DIG} \leftarrow \mathsf{O\text{-}BIBE.Digest}(\mathsf{PK}_{OB}, S)$ and computes $\mathsf{DK}_{OB} \leftarrow \mathsf{O\text{-}BIBE.ComputeKey}(\mathsf{PK}_{OB}, \mathsf{MK}_{OB}, \mathsf{DIG}, \mathsf{BL})$.
3. It generates $(\mathsf{RK}_{OB}, \mathsf{TK}_{OB}, (\mathsf{WT}_{i,OB})_{i\in U}) \leftarrow \mathsf{O\text{-}BIBE.Delegate}(\mathsf{PK}_{OB}, \mathsf{DK}_{OB}, (\mathsf{CH}_{i,OB})_{i\in U}, \mathsf{BL})$.
4. It sets a recovery token $\mathsf{RT} = \big(\mathsf{RK}_{OB}, (\mathsf{CT}_{i,SKE}, \eta_i)_{i\in U}, \mathsf{BL}\big)$, a transformation key $\mathsf{TK} = \mathsf{TK}_{OB}$, and a work task $\mathsf{WT}_i = (\mathsf{WT}_{i,OB}, \mathsf{VK}_{i,OTS})$ for each $i \in U$.
5. Finally, it outputs $(\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.

VOBE.Transform$(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OB}, \mathsf{H}_s)$, $\mathsf{TK} = \mathsf{TK}_{OB}$, and $\mathsf{WT}_i = (\mathsf{WT}_{i,OB}, \mathsf{VK}_{i,OTS})$ for $i \in U$.

1. It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ for all $i \in U$.
2. It obtains $(\mathsf{TT}_{i,OB})_{i\in U} \leftarrow \mathsf{O\text{-}BIBE.Transform}(\mathsf{PK}_{OB}, \mathsf{TK}_{OB}, (\mathsf{WT}_{i,OB})_{i\in U}, (\mathsf{ID}_i)_{i\in U}, \mathsf{BL})$ and sets $\mathsf{TT}_i = (\mathsf{TT}_{i,OB}, \mathsf{ID}_i)$ for $i \in U$.
3. Finally, it outputs the transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

VOBE.Recover$(\mathsf{RT}, (\mathsf{TT}_i)_{i\in U})$**:** Let $\mathsf{RT} = (\mathsf{RK}_{OB}, (\mathsf{CT}_{i,SKE}, \eta_i)_{i\in U}, \mathsf{BL})$, and $\mathsf{TT}_i = (\mathsf{TT}_{i,OB}, \mathsf{ID}_i)$ for $i \in U$.

1. It recovers capsule keys $(\mathsf{CK}_{i,OB})_{i\in U} \leftarrow \mathsf{O\text{-}BIBE.Recover}(\mathsf{RK}_{OB}, (\mathsf{TT}_{i,OB})_{i\in U})$.
2. For each $i \in [B]$, it proceeds as follows:
   (a) If $i \notin U$, it sets $M_i = \perp$ and continues to the next index.
   (b) It derives the key-tag value $(K'_i \| \eta'_i) = \mathsf{PRF}(\mathsf{CK}_{i,OB}, \mathsf{ID}_i \| \mathsf{BL})$.
   (c) If the verification $\eta_i \stackrel{?}{=} \eta'_i$ holds, it decrypts the message $M_i \leftarrow \mathsf{SKE.Decrypt}(K'_i, \mathsf{CT}_{i,SKE})$; otherwise, it sets $M_i = \perp\!\!\!\perp$.
3. Finally, it outputs the recovered messages $(M_i)_{i\in[B]} \in \mathcal{M}^B_{\perp\!\!\!\perp}$.

4.3 Correctness

Theorem 4.1. *The proposed VOBE scheme satisfies correctness, provided that the underlying O-BIBE scheme, the OTS scheme, and the SKE scheme satisfy their respective correctness properties.*

Proof. The correctness of the VOBE scheme is established by demonstrating the validity of both batch decryption and outsourced batch decryption.

By the correctness of the OTS scheme, the identity set $S = \{\mathsf{ID}_i\}_{i\in[B]}$ derived from the verification keys $\{\mathsf{VK}_{i,OTS}\}$ is consistent with the identities associated with the batch label BL. Since $\mathsf{ID}_i \in S$, the batch decryption correctness of the underlying O-BIBE scheme guarantees the recovery of the capsule keys $\mathsf{CK}_{i,OB}$. Because PRF is a deterministic function, the derived SKE key K_i matches the value generated during SKE encryption. Consequently, the original messages M_i are correctly restored through SKE decryption by the correctness of the SKE scheme.

In the outsourced setting, the correctness of the OTS scheme ensures the consistency of the identity set S and the batch label BL during delegation and transformation. The $\mathsf{O\text{-}BIBE.Recover}$ algorithm effectively derives the valid capsule keys $\mathsf{CK}_{i,OB}$ from the transformed tasks. In the $\mathsf{VOBE.Recover}$ algorithm, the computed tag η_i' from PRF remains identical to the η_i stored in the state, validating the integrity of the outsourced computation. Finally, the deterministic nature of PRF ensures that the recovered symmetric key K_i' is identical to the symmetric key, allowing for the perfect restoration of the original messages $(M_i)_{i\in[B]}$ by the correctness of the SKE scheme. □

4.4 Security Analysis

4.4.1 IND-CCA Security

Theorem 4.2. *The proposed VOBE scheme satisfies SE-IND-CCA security for any polynomial batch size B, assuming that the OTS scheme is SUF secure, and the hash function is collision-resistant, the underlying O-BIBE scheme is SE-IND-CPA secure, the PRF is pseudo-random, and the SKE scheme is IND-OT secure.*

Proof. To prove the SE-IND-CCA security of the VOBE scheme, we employ the hybrid game technique. We define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \cdots, \mathbf{G}_6$ where the first game will be the original SE-IND-CCA security game and the last one will be a game in which an adversary has no advantage. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. The hybrid games are defined as follows:

Game $\mathbf{G}_0$. This game corresponds to the original SE-IND-CCA security experiment in Definition 4.3. The simulator of this game uses the master key $\mathsf{MK} = \mathsf{MK}_{OB}$ to handle all queries of $\mathcal{A}$. By the definition, it follows that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} = \mathsf{Adv}_{VOBE,\mathcal{A}}^{\mathsf{SE\text{-}IND\text{-}CCA}}(\lambda).$$

Game $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$ except for the handling of $\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$ queries. The simulator $\mathcal{B}$ picks a challenge OTS key pair $(\mathsf{SK}_{OTS}^*, \mathsf{VK}_{OTS}^*)$ for the challenge ciphertext at the beginning of the game. Let BAD_1 be the event that the adversary $\mathcal{A}$ submits a query containing a ciphertext $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for some $i \in [B]$ such that:

$$\begin{aligned}
&\mathsf{VK}_{i,OTS} = \mathsf{VK}_{OTS}^* \wedge \\
&(\mathsf{BL}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i) \neq (\mathsf{BL}^*, \mathsf{CH}_{OB}^*, \mathsf{CT}_{SKE}^*, \eta^*) \wedge \\
&\mathsf{OTS.Verify}(\mathsf{VK}_{OTS}^*, (\mathsf{BL} \| \mathsf{CH}_{i,OB} \| \mathsf{CT}_{i,SKE} \| \eta_i), \sigma_i) = 1
\end{aligned}$$

where $\mathsf{CT}^* = (\mathsf{VK}_{OTS}^*, \mathsf{CH}_{OB}^*, \mathsf{CT}_{SKE}^*, \eta^*, \sigma^*)$ is the challenge ciphertext. If BAD_1 occurs, it sets $\mathsf{BAD}_1 = 1$ and aborts the game. Since the probability of aborting in this game is bounded by the strong unforgeability (SUF) of the OTS scheme, we have

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}\right| \leq \Pr[\mathsf{BAD}_1] \leq \mathsf{Adv}_{OTS,\mathcal{B}}^{\mathsf{SUF}}(\lambda).$$

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$ except that the simulator additionally aborts in $\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$ queries. Let BAD_2 be the event that the adversary $\mathcal{A}$ submits a query containing $\mathsf{VK}_{i,OTS}$ such that:

$$\mathsf{VK}_{i,OTS} \neq \mathsf{VK}^*_{OTS} \wedge \mathsf{H}_s(\mathsf{VK}_{i,OTS}) = \mathsf{H}_s(\mathsf{VK}^*_{OTS}).$$

If BAD_2 occurs, it sets $\mathsf{BAD}_2 = 1$ and aborts the game. Since the probability of this aborting in this game is bounded by the collision resistance (CR) of the hash function, we have

$$\left|\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}\right| \leq \Pr[\mathsf{BAD}_2] \leq \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_3$. In this game, the simulator no longer uses the master key $\mathsf{MK} = \mathsf{MK}_{OB}$. Instead, it simulates the VOBE oracles by invoking the O-BIBE oracles $\mathsf{OCreateDK}$, $\mathsf{ORevealDK}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRK}$. For a query on $((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$, the simulator extracts $S = \{\mathsf{ID}_i\}$ and requests the corresponding DK_{OB} on (S, BL) from the O-BIBE challenger.

Specifically, for queries under the challenge label BL^* where $\mathsf{CT}^* \notin (\mathsf{CT}_i)_{i\in[B]}$, the condition $\mathsf{ID}^* \notin S$ is strictly guaranteed because the events BAD_1 (reuse of the challenge verification key) and BAD_2 (hash collision) did not occur in the previous games. This ensures that all such queries remain valid within the O-BIBE security model and can be perfectly simulated by the O-BIBE challenger. Due to the correctness of the underlying O-BIBE scheme, the distribution of responses is identical to that in $\mathbf{G}_2$. Thus, we have

$$\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}} = \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}.$$

Game $\mathbf{G}_4$. This game is identical to $\mathbf{G}_3$ except that the capsule key CK^*_{OB} of the challenge ciphertext CT^* is replaced by a random capsule key $\widetilde{CK} \in \mathbb{G}_T$. Since this change relies on the SE-IND-CPA secuirty of the O-BIBE scheme from Lemma 4.5, we have

$$\left|\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{O\text{-}BIBE,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_5$. In the challenge ciphertext CT^*, the simulator replaces the PRF outputs $(K^*\|\eta^*) = \mathsf{PRF}(\widetilde{CK}, \mathsf{ID}^*\|\mathsf{BL}^*)$ with a random string $(\widetilde{K}\|\widetilde{\eta})$ chosen uniformly at random. Since this transition relies on the pseudo-randomness of PRF, we have

$$\left|\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_6$. In the final game, the simulator replaces the challenge SKE ciphertext $\mathsf{CT}^*_{SKE} \leftarrow \mathsf{SKE.Encrypt}(\widetilde{K}, M^*_\mu)$ with an encryption of a constant zero-message $\mathsf{CT}^*_{SKE} \leftarrow \mathsf{SKE.Encrypt}(\widetilde{K}, 0^{|M|})$. Since $\widetilde{K}$ is a fresh random key unknown to $\mathcal{A}$, this transition is bounded by the IND-OT security of the SKE scheme.

$$\left|\mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{B}}(\lambda).$$

Since the challenge ciphertext CT^* in the final game is not related to the random coin μ, we have $\mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}} = 0$.

By applying the advantage differences established in Lemmas 4.3, 4.4, 4.5, 4.6, and 4.7, we obtain the following upper bound on the total advantage:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CCA}}_{VOBE,\mathcal{A}}(\lambda) \\
&\leq \sum_{j=1}^{6} \left|\mathsf{Adv}^{\mathbf{G}_{j-1}}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_j}_{\mathcal{A}}\right| + \mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}} \\
&\leq \mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{O\text{-}BIBE,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{B}}(\lambda).
\end{aligned}$$

This completes the proof. □

Lemma 4.3. *The games $\boldsymbol{G}_0$ and $\boldsymbol{G}_1$ are computationally indistinguishable, assuming that the OTS scheme is SUF secure.*

Proof. The only difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ is the occurrence of the event BAD_1. Since the two games are identical unless BAD_1 occurs, we have $|\mathsf{Adv}^{\mathbf{G}_0}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}| \leq \Pr[\mathsf{BAD}_1]$ by the Difference Lemma.

We now construct a simulator $\mathcal{B}$ that breaks the SUF security of the OTS scheme using the occurrence of BAD_1. Initially, $\mathcal{B}$ receives a target verification key VK^*_{OTS} from the OTS challenger. $\mathcal{B}$ generates the master key MK by running $\mathsf{VOBE.Setup}(1^\lambda, 1^B)$ and perfectly simulates oracle queries ($\mathsf{ODecrypt}$ and $\mathsf{ODelegate}$) from the adversary $\mathcal{A}$ using MK. In the challenge step, $\mathcal{B}$ constructs the challenge ciphertext $\mathsf{CT}^* = (\mathsf{VK}^*_{OTS}, \mathsf{CH}^*_{OB}, \mathsf{CT}^*_{SKE}, \eta^*, \sigma^*)$ by using an OTS signature σ^* received from the OTS oracle on the message $(\mathsf{BL}^* \| \mathsf{CH}^*_{OB} \| \mathsf{CT}^*_{SKE} \| \eta^*)$.

If the event BAD_1 occurred during the query phase, it implies that $\mathcal{A}$ has submitted a ciphertext $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for some $i \in [B]$ such that $\mathsf{VK}_{i,OTS} = \mathsf{VK}^*_{OTS}$ but $\mathsf{CT}_i \neq \mathsf{CT}^*$. Since CT^* is the only ciphertext signed under VK^*_{OTS} by the simulator, such a CT_i necessarily contains a valid new OTS signature for VK^*_{OTS}. Thus, $\mathcal{B}$ can output $(\mathsf{BL}_i \| \mathsf{CH}_{i,OB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$ as the successful forgery of the OTS scheme, which implies $\Pr[\mathsf{BAD}_1] \leq \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{OTS},\mathcal{B}}(\lambda)$. This completes the proof. □

Lemma 4.4. *The games $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are computationally indistinguishable, assuming that the hash function is collision-resistant.*

Proof. The games $\mathbf{G}_1$ and $\mathbf{G}_2$ are identical unless the event BAD_2 occurs. By the Difference Lemma, we have $|\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}| \leq \Pr[\mathsf{BAD}_2]$.

We construct a simulator $\mathcal{B}$ that finds a collision in the hash function $\mathcal{H}$ upon the occurrence of BAD_2. $\mathcal{B}$ receives a seed s for the hash function H_s from the CR challenger. $\mathcal{B}$ runs $\mathsf{VOBE.Setup}(1^\lambda, 1^B)$ to obtain the master key MK and perfectly simulates all oracle queries from the adversary $\mathcal{A}$ using MK. $\mathcal{B}$ also generates the target OTS key-pair $(\mathsf{SK}^*_{OTS}, \mathsf{VK}^*_{OTS}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$ for the challenge ciphertext and computes the target identity $\mathsf{ID}^* = \mathsf{H}_s(\mathsf{VK}^*_{OTS})$.

If $\mathcal{A}$ triggers the event BAD_2 during its oracle queries, it implies there exists a ciphertext CT_i such that $\mathsf{ID}_i = \mathsf{ID}^*$ but $\mathsf{VK}_{i,OTS} \neq \mathsf{VK}^*_{OTS}$. Since $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and $\mathsf{ID}^* = \mathsf{H}_s(\mathsf{VK}^*_{OTS})$, the condition $\mathsf{ID}_i = \mathsf{ID}^*$ directly implies $\mathsf{H}_s(\mathsf{VK}_{i,OTS}) = \mathsf{H}_s(\mathsf{VK}^*_{OTS})$. Given that $\mathsf{VK}_{i,OTS} \neq \mathsf{VK}^*_{OTS}$, $\mathcal{B}$ can output $(\mathsf{VK}_{i,OTS}, \mathsf{VK}^*_{OTS})$ as a valid collision for H_s. Therefore, we have $\Pr[\mathsf{BAD}_2] \leq \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda)$, which completes the proof. □

Lemma 4.5. *The games $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$ are computationally indistinguishable, assuming that the underlying O-BIBE scheme is SE-IND-CPA secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the security of the SE-IND-CCA game with a non-negligible advantage. Let $\mathcal{C}$ be the challenger of the O-BIBE scheme. The simulator $\mathcal{B}$ that breaks the O-BIBE scheme by using $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ commits to a challenge batch label BL^*. $\mathcal{B}$ generates $(\mathsf{SK}^*_{OTS}, \mathsf{VK}^*_{OTS}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$ for the challenge ciphertext and computes $\mathsf{ID}^* = \mathsf{H}_s(\mathsf{VK}^*_{OTS})$. It commits to a challenge identity-label pair $(\mathsf{ID}^*, \mathsf{BL}^*)$ to the O-BIBE challenger $\mathcal{C}$ and receives $(\mathsf{EK}_{OB}, \mathsf{PK}_{OB})$.

Setup: It selects a hash function H_s and sets $\mathsf{EK} = (\mathsf{EK}_{OB}, \mathsf{H}_s)$ and $\mathsf{PK} = (\mathsf{PK}_{OB}, \mathsf{H}_s)$. It initializes two empty lists $L_{\text{map}} = \emptyset$ and $L_{\text{del}} = \emptyset$, which are used to track handle mapping and delegation queries. It gives $(\mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.

Pre-challenge Queries: $\mathcal{B}$ handles the queries as follows:

- $\mathsf{ODecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: Let $\mathsf{CTS} = (\mathsf{CT}_i)_{i\in B}$ where $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$. It proceeds as follows:
  1. It initialize $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
     (a) If the OTS verification of σ_i is invalid, it sets $M_i = \perp$ and continues to the next index.
     (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
  2. If $(\mathsf{CTS}, \mathsf{BL}, \cdot) \in L_{\text{map}}$, it retrieves $(\mathsf{CTS}, \mathsf{BL}, \mathsf{hk}_{OB}) \leftarrow L_{\text{map}}$. Otherwise, it acquires hk_{OB} from $\mathcal{C}$ by querying $\mathsf{OCreateDK}(S, \mathsf{BL})$ and updates $L_{\text{map}} \leftarrow L_{\text{map}} \cup \{(\mathsf{CTS}, \mathsf{BL}, \mathsf{hk}_{OB})\}$.
  3. It acquires DK_{OB} from $\mathcal{C}$ by querying $\mathsf{ORevealDK}(\mathsf{hk}_{OB})$.
  4. For each $i \in U$, it proceeds as follows:
     (a) It recovers $\mathsf{CK}_{i,OB} \leftarrow \mathsf{O\text{-}BIBE.Decaps}(\mathsf{PK}_{OB}, \mathsf{DK}_{OB}, S, \mathsf{CH}_{i,OB}, \mathsf{ID}_i, \mathsf{BL})$.
     (b) It derives the key-tag pair $(K_i \| \eta_i) = \mathsf{PRF}(\mathsf{CK}_{i,OB}, \mathsf{ID}_i \| \mathsf{BL})$ and decrypts $M_i \leftarrow \mathsf{SKE.Decrypt}(K_i, \mathsf{CT}_{i,SKE})$.
  5. It responds with $(M_i)_{i\in[B]}$.
- $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: Let $\mathsf{CTS} = (\mathsf{CT}_i)_{i\in B}$ where $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$. It proceeds as follows:
  1. It initialize $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
     (a) If the OTS verification of σ_i is invalid, it sets $M_i = \perp$ and continues to the next index.
     (b) Otherwise, it computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
  2. If $(\mathsf{CTS}, \mathsf{BL}, \cdot) \in L_{\text{map}}$, it retrieves $(\mathsf{CTS}, \mathsf{BL}, \mathsf{hk}_{OB}) \leftarrow L_{\text{map}}$. Otherwise, it acquires hk_{OB} from $\mathcal{C}$ by querying $\mathsf{OCreateDK}(S, \mathsf{BL})$ and updates $L_{\text{map}} \leftarrow L_{\text{map}} \cup \{(\mathsf{CTS}, \mathsf{BL}, \mathsf{hk}_{OB})\}$.
  3. It acquires $(\mathsf{hd}_{OB}, \mathsf{TK}_{OB}, (\mathsf{WT}_{i,OB})_{i\in U})$ from $\mathcal{C}$ by querying $\mathsf{ODelegate}(\mathsf{hk}_{OB}, (\mathsf{CH}_{i,OB})_{i\in U})$.
  4. If $\mathsf{BL} = \mathsf{BL}^*$ and $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$, it sets $\mathsf{RT} = \perp$. Otherwise, it acquires RK_{OB} from $\mathcal{C}$ by querying $\mathsf{ORevealRK}(\mathsf{hd}_{OB})$, and sets $\mathsf{RT} = (\mathsf{RK}_{OB}, (\mathsf{CT}_{i,SKE}, \eta_i)_{i\in U}, \mathsf{BL})$.
  5. It sets $\mathsf{TK} = \mathsf{TK}_{OB}$ and $\mathsf{WT}_i = (\mathsf{WT}_{i,OB}, \mathsf{VK}_{i,OTS})$ for all $i \in U$.
  6. It picks a fresh handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\text{del}} \leftarrow L_{\text{del}} \cup \{(\mathsf{hd}, \mathsf{RT})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.
- $\mathsf{ORevealRT}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\text{del}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(\mathsf{hd}, \mathsf{RT}) \leftarrow L_{\text{del}}$ and responds with RT. Note that the case $(\mathsf{RT} = \perp)$ cannot occur due to the challenge privacy constraint.

Challenge: $\mathcal{A}$ submits two messages M^*_0, M^*_1, $\mathcal{B}$ proceeds as follows:

1. It acquires the challenge O-BIBE ciphertext pair $(\mathsf{CH}^*_{OB}, \mathsf{CK}^*_{OB})$ from $\mathcal{C}$.

2. It derives $(K^* \| \eta^*) \leftarrow \mathsf{PRF}(\mathsf{CK}^*_{OB}, \mathsf{ID}^* \| \mathsf{BL}^*)$. It flips a random coin $\mu \in \{0,1\}$ and computes $\mathsf{CT}^*_{SKE} \leftarrow \mathsf{SKE.Encrypt}(K^*, M^*_\mu)$. It generates $\sigma^* \leftarrow \mathsf{OTS.Sign}(\mathsf{SK}^*_{OTS}, \mathsf{BL}^* \| \mathsf{CH}^*_{OB} \| \mathsf{CT}^*_{SKE} \| \eta^*)$.

3. It sets $\mathsf{CT}^* = (\mathsf{VK}^*_{OTS}, \mathsf{CH}^*_{OB}, \mathsf{CT}^*_{SKE}, \eta^*, \sigma^*)$ and gives CT^* to $\mathcal{A}$.

Post-challenge Queries: $\mathcal{B}$ responds to these queries in the same manner as the pre-challenge queries, subject to the restrictions of the security game.

Output: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

Analysis of Simulation: The validity of the simulator $\mathcal{B}$ is grounded in the following points:

- Oracle Compatibility: For any $\mathsf{ODecrypt}$ or $\mathsf{ODelegate}$ query that does not involve CT^*, the absence of BAD_1 and BAD_2 guarantees that $\mathsf{ID}^* \notin S$. This allows $\mathcal{B}$ to freely request DK_{OB} from the O-BIBE challenger $\mathcal{C}$ without violating the O-BIBE security model.

- Handling Challenge Context: When the adversary $\mathcal{A}$ requests an $\mathsf{ODelegate}$ query on BL^* for a ciphertext set containing CT^*, $\mathcal{B}$ invokes the O-BIBE $\mathsf{ODelegate}$ oracle to provide a consistent response. While $\mathcal{B}$ sets $\mathsf{RT} = \perp$ in this case, the simulation remains perfectly valid because the VOBE security game strictly prohibits $\mathcal{A}$ from issuing an $\mathsf{OReveal RT}$ query on the corresponding handle.

- Probability Reduction: If $\mathcal{C}$ provides a real capsule key CK^*_{OB}, $\mathcal{A}$'s view is identical to $\mathbf{G}_3$. If $\mathcal{C}$ provides a truly random string, $\mathcal{A}$'s view corresponds to $\mathbf{G}_4$. Consequently, the advantage of $\mathcal{A}$ in distinguishing these two games is bounded by the advantage of $\mathcal{B}$ in breaking the SE-IND-CPA security of the underlying O-BIBE scheme:

$$|\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}| \leq \mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{\mathit{O\text{-}BIBE},\mathcal{B}}(\lambda).$$

This completes the proof. □

Lemma 4.6. *The games $\mathbf{G}_4$ and $\mathbf{G}_5$ are computationally indistinguishable, assuming that the PRF is pseudorandom.*

Proof. The only difference between $\mathbf{G}_4$ and $\mathbf{G}_5$ lies in the generation of $(K^* \| \eta^*)$ for the challenge ciphertext CT^*. In $\mathbf{G}_4$, these values are derived via $\mathsf{PRF}(\mathsf{CK}^*_{OB}, \mathsf{ID}^* \| \mathsf{BL}^*)$, whereas in $\mathbf{G}_5$, they are replaced by a truly random string of the same length.

We construct a simulator $\mathcal{B}$ that breaks the pseudo-randomness of the PRF. $\mathcal{B}$ is given access to an oracle $\mathsf{O}(\cdot)$, which is either a real PRF $\mathsf{F}(k,\cdot)$ for a hidden key k or a truly random function $f(\cdot)$. $\mathcal{B}$ simulates all oracles for $\mathcal{A}$ as in $\mathbf{G}_4$. Note that due to the transition in $\mathbf{G}_3$, the capsule key CK^*_{OB} is now a uniform random string, which can perfectly play the role of the hidden PRF key k. In the challenge step, $\mathcal{B}$ does not compute the PRF locally. Instead, it queries its oracle $\mathsf{O}(\cdot)$ with the input $(\mathsf{ID}^* \| \mathsf{BL}^*)$ and sets the output as $(K^* \| \eta^*)$. If $\mathsf{O}(\cdot) = \mathsf{F}(k,\cdot)$, the view of $\mathcal{A}$ is identical to $\mathbf{G}_4$. If $\mathsf{O}(\cdot) = f(\cdot)$, the view of $\mathcal{A}$ is identical to $\mathbf{G}_5$. Because BAD_1 and BAD_2 are excluded and the underlying O-BIBE security ensures CK^*_{OB} remains unknown to $\mathcal{A}$, the specific PRF value is never leaked through other oracle queries. Thus, the advantage of $\mathcal{A}$ in distinguishing these games is exactly the advantage of $\mathcal{B}$ in breaking the PRF:

$$|\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}| \leq \mathsf{Adv}^{\mathsf{PR}}_{\mathit{PRF},\mathcal{B}}(\lambda).$$

This completes the proof. □

Lemma 4.7. *The games $\mathbf{G}_5$ and $\mathbf{G}_6$ are computationally indistinguishable, assuming that the SKE scheme is IND-OT secure.*

Proof. The difference between $\mathbf{G}_5$ and $\mathbf{G}_6$ lies in the SKE component CT^*_{SKE} of the challenge ciphertext. In $\mathbf{G}_5$, CT^*_{SKE} is an encryption of M^*_μ, whereas in $\mathbf{G}_6$, it is replaced by an encryption of a fixed string $0^{|M^*|}$.

We construct a simulator $\mathcal{B}$ that breaks the IND-OT security of the SKE scheme. $\mathcal{B}$ interacts with the SKE challenger. Throughout the game, $\mathcal{B}$ simulates all oracles for $\mathcal{A}$ as in $\mathbf{G}_5$. In the challenge step, $\mathcal{B}$ proceeds as follows:

1. It picks a random bit $\mu \in \{0,1\}$ internally.
2. It submits two challenge messages $m_0 = M^*_\mu$ and $m_1 = 0^{|M^*|}$ to the SKE challenger.
3. Upon receiving the challenge ciphertext from the SKE challenger, $\mathcal{B}$ sets it as CT^*_{SKE} and constructs the final VOBE challenge ciphertext CT^*, which is then given to $\mathcal{A}$.

Since the symmetric key K^* in $\mathbf{G}_5$ is a truly random string and is never used in any other part of the simulation (due to the exclusion of $\mathsf{BAD}_1, \mathsf{BAD}_2$ and the PRF transition), it perfectly functions as the hidden one-time key in the IND-OT experiment.

- If the SKE challenger encrypts m_0, the view of $\mathcal{A}$ is identical to $\mathbf{G}_5$.
- If the SKE challenger encrypts m_1, the view of $\mathcal{A}$ is identical to $\mathbf{G}_6$.

In $\mathbf{G}_6$, the challenge ciphertext is completely independent of the bit μ, meaning $\mathcal{A}$ has zero advantage in guessing μ. Consequently, the advantage of $\mathcal{A}$ in distinguishing these games is bounded by the IND-OT advantage of $\mathcal{B}$:

$$|\mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}}| \leq \mathsf{Adv}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{B}}(\lambda).$$

This completes the proof. □

4.4.2 Verifiability

Theorem 4.8. *The proposed VOBE scheme satisfies SE-VER security for any polynomial batch size B, assuming that the OTS scheme is SUF secure, and the hash function is collision-resistant, underlying O-BIBE scheme is SE-IND-CPA secure, and the PRF is pseudo-random.*

Proof. To prove the SE-VER security of the VOBE scheme, we employ the hybrid game technique. We define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \cdots, \mathbf{G}_5$ where the first game will be the original SE-VER security game and the last one will be a game in which an adversary has negligible advantage. Let $\mathsf{Adv}^{\mathbf{G}_j}_{\mathcal{A}}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. The hybrid games are defined as follows:

Game $\mathbf{G}_0$. This game corresponds to the original SE-VER security experiment in Definition 4.4. Since the simulator of this game follows the honest algorithms, it follows by definition that

$$\mathsf{Adv}^{\mathbf{G}_0}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{SE\text{-}VER}}_{VOBE,\mathcal{A}}(\lambda).$$

Game $\mathbf{G}_1$. This game is defined identically to $\mathbf{G}_1$ in the proof of SE-IND-CCA security (see Theorem 4.2). It introduces an abort condition BAD_1 that occurs if the adversary forges a valid signature σ_i under the challenge verification key VK^*_{OTS}. As established in the previous proof, the probability of BAD_1 is bounded by the strong unforgeability of the OTS scheme:

$$\left|\mathsf{Adv}^{\mathbf{G}_0}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}\right| \leq \Pr[\mathsf{BAD}_1] \leq \mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_2$. This game is defined identically to $\mathbf{G}_2$ in the SE-IND-CCA proof (see Theorem 4.2). It also introduces an abort condition BAD_2 that accounts for potential hash collisions between $\mathsf{VK}_{i,OTS}$ and VK^*_{OTS}, ensuring that $\mathsf{ID}_i \neq \mathsf{ID}^*$ for all queries. The transition probability is bounded by the collision resistance (CR) of the hash function H_s:

$$\left|\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}\right| \leq \Pr[\mathsf{BAD}_2] \leq \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_3$. This game is defined identically to $\mathbf{G}_3$ in the SE-IND-CCA proof (see Theorem 4.2). The simulator transitions from using the master key MK to invoking the O-BIBE oracles. Since the conditions $\mathsf{ID}^* \notin S$ and $\mathsf{VK}_{i,OTS} \neq \mathsf{VK}^*_{OTS}$ are guaranteed by the previous games, the simulation is perfect due to the correctness of the underlying O-BIBE scheme. Consequently,

$$\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}} = \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}.$$

Game $\mathbf{G}_4$. This game follows the same construction as $\mathbf{G}_4$ in the SE-IND-CCA proof (see Theorem 4.2). The challenge capsule key CK^*_{OB} is replaced with a uniform random value $\widetilde{CK} \in \mathbb{G}_T$. The indistinguishability between $\mathbf{G}_3$ and $\mathbf{G}_4$ is directly reduced to the SE-IND-CPA security of the O-BIBE scheme:

$$\left|\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{O\text{-}BIBE,\mathcal{B}}(\lambda).$$

Game $\mathbf{G}_5$. In the challenge ciphertext CT^*, the simulator replaces the PRF outputs $(K^* \| \eta^*) = \mathsf{PRF}(\widetilde{CK}, \mathsf{ID}^* \| \mathsf{BL}^*)$ with a random string $(\widetilde{K} \| \widetilde{\eta})$ chosen uniformly at random. Since this transition relies on the pseudo-randomness of PRF, we have

$$\left|\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda).$$

In this final game $\mathbf{G}_5$, the tag η^* (and any tag computed during the simulation) is no longer the output of a PRF but a purely random string sampled uniformly from $\{0,1\}^\lambda$. This random string is statistically independent of any information available to $\mathcal{A}$ in its view. For $\mathcal{A}$ to successfully forge a valid ciphertext, it must output a tag that matches the randomly chosen string by the simulator. Since the choice of the tag is independent of the adversary's capabilities, the probability of a successful forgery is exactly the probability of guessing a λ-bit string:

$$\mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}} = \frac{1}{2^\lambda}.$$

By applying the advantage differences established in Lemmas 4.9, 4.10, 4.11, and 4.12, we obtain the following upper bound on the total advantage:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{SE\text{-}VER}}_{VOBE,\mathcal{A}}(\lambda) \\
&\leq \sum_{j=1}^{5} \left|\mathsf{Adv}^{\mathbf{G}_{j-1}}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_j}_{\mathcal{A}}\right| + \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}} \\
&\leq \mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{SE\text{-}IND\text{-}CPA}}_{O\text{-}BIBE,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda) + \frac{1}{2^\lambda}.
\end{aligned}$$

This completes the proof. □

Lemma 4.9. *The games $\boldsymbol{G}_0$ and $\boldsymbol{G}_1$ are computationally indistinguishable, assuming that the OTS scheme is SUF secure.*

Lemma 4.10. *The games $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are computationally indistinguishable, assuming that the hash function is collision-resistant.*

Lemma 4.11. *The games $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$ are computationally indistinguishable, assuming that the underlying O-BIBE scheme is SE-IND-CPA secure.*

Lemma 4.12. *The games $\boldsymbol{G}_4$ and $\boldsymbol{G}_5$ are computationally indistinguishable, assuming that the PRF is pseudo-random.*

The proofs of Lemmas 4.9–4.12 are structurally identical to the corresponding reductions provided in the proof of Theorem 4.2. For brevity, we omit the individual proof for each transition and adopt the same probability bounds as established in the SE-IND-CCA analysis. We refer the reader to Theorem 4.2 for the complete details of these reduction arguments.

4.5 Discussions

Threshold VOBE Extension. While our proposed VOBE scheme supports efficient batch decryption via a single designated server, it can be further extended to a threshold-based setting to eliminate the reliance on a single point of trust. In this extended system, referred to as verifiable outsourced BTE (VOBTE), the batch decryption process is collectively performed by a decryption committee. A valid restoration of the batch plaintexts is achieved only when a threshold number of authorized nodes provide their respective decryption key shares. By leveraging the O-TBIBE scheme in Appendix A, which is the threshold extension of the O-BIBE scheme, we demonstrate that designing a secure VOBTE scheme is indeed feasible. The VOBTE scheme largely mirrors the architecture of our VOBE scheme, with the primary distinction being that the decryption and delegation algorithms take key shares from committee nodes as input to derive the original decryption keys, rather than relying on a single master secret key. For a more comprehensive treatment of the VOBTE syntax, security model, and concrete construction, we refer the reader to Appendix B.

5 Conclusion

In this paper, we proposed a novel verifiable outsourcing framework for batched encryption to address the inherent computational efficiency bottlenecks of cryptographic systems designed to mitigate MEV attacks in blockchains. First, we introduced the O-BIBE paradigm by integrating a decryption key re-randomization mechanism into the conventional BIBE framework, allowing secure delegation of batch decryption operations to an untrusted cloud server. Building upon this foundation, we strategically combined O-BIBE with complementary cryptographic primitives to instantiate a secure and efficient VOBE scheme, simultaneously achieving both public verifiability and robust confidentiality. Furthermore, to accommodate decentralized trust in distributed environments, we extended our core architectures to the threshold setting, formalizing the O-TBIBE framework and subsequently constructing the fully featured VOBTE system. Notably, both our proposed VOBE and VOBTE schemes successfully offload the heavy algebraic workloads of batch processing onto the cloud, effectively reducing the client-side computational complexity for decrypting B ciphertexts from the conventional $O(B^2)$ or $O(B\log B)$ overhead down to a strict $O(B)$ linear complexity. Coupled with a seamless, unforgeable verification mechanism that cross-checks the server's returned results, our framework presents a highly scalable, robust, and mathematically sound solution for secure transaction batching in modern decentralized systems.

References

[1] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025, Part III*, Lecture Notes in Computer Science, pages 69–100. Springer, 2025.

[2] Leemon Baird, Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Threshold signatures in the multiverse. In *IEEE Symposium on Security and Privacy, SP 2023*, pages 1454–1470. IEEE Computer Society, 2023.

[3] Joseph Bebel and Dev Ojha. Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Paper 2022/898, 2022.

[4] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 1998.

[5] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[6] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

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

[8] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

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

[10] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: epochless approach to batched threshold encryption for MEV prevention. In Lujo Bauer and Giancarlo Pellegrino, editors, *34th USENIX Security Symposium, USENIX Security 2025*, pages 3457–3476. USENIX Association, 2025.

[11] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

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

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

[14] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *IEEE Symposium on Security and Privacy - SP 2020*, pages 910–927. IEEE, 2020.

[15] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, Lecture Notes in Computer Science, pages 307–315. Springer, 1989.

[16] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: Verifiable encryption to the future made practical. In Foteini Baldimtsi and Christian Cachin, editors, *Financial Cryptography and Data Security - FC 2023, Part I*, Lecture Notes in Computer Science, pages 252–269. Springer, 2023.

[17] Jesko Dujmovic, Rachit Garg, and Giulio Malavolta. Time-lock puzzles with efficient batch solving. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024, Part II*, Lecture Notes in Computer Science, pages 311–341. Springer, 2024.

[18] Stefan Dziembowski, Sebastian Faust, and Jannik Luhn. Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Paper 2024/1981, 2024.

[19] Dankrad Feist and Dmitry Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Paper 2023/033, 2023.

[20] Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024, Part X*, Lecture Notes in Computer Science, pages 449–487. Springer, 2024.

[21] Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable snarks and applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025, Part III*, Lecture Notes in Computer Science, pages 504–539. Springer, 2025.

[22] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. hinTS: Threshold signatures with silent setup. In *IEEE Symposium on Security and Privacy, SP 2024*, pages 3034–3052. IEEE, 2024.

[23] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024, Part VII*, Lecture Notes in Computer Science, pages 352–386. Springer, 2024.

[24] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, Lecture Notes in Computer Science, pages 354–371. Springer, 1996.

[25] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.

[26] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology - EUROCRYPT 2026, Part V*, Lecture Notes in Computer Science, pages 548–578. Springer, 2026.

[27] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of ABE ciphertexts. In *20th USENIX Security Symposium - USENIX 2011*. USENIX Association, 2011.

[28] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, Lecture Notes in Computer Science, pages 177–194. Springer, 2010.

[29] Peyman Momeni, Sergey Gorbunov, and Bohan Zhang. Fairblock: Preventing blockchain front-running with minimal overheads. In Fengjun Li, Kaitai Liang, Zhiqiang Lin, and Sokratis K. Katsikas, editors, *Security and Privacy in Communication Networks - SecureComm 2022*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 250–271. Springer, 2022.

[30] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, 1991.

[31] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[32] Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology - EUROCRYPT 2026, Part V*, Lecture Notes in Computer Science, pages 517–547. Springer, 2026.

A Outsourced Threshold BIBE

In this appendix, we extend our discussion to the threshold setting by presenting the syntax and security model for O-TBIBE. This framework incorporates a threshold decryption, providing enhanced fault tolerance and control in an outsourced environment. Building upon the existing GWWW-TBIBE scheme in the GGM, we propose a concrete O-TBIBE scheme that integrates secure delegation mechanisms while maintaining threshold properties. We then conduct a formal security analysis, demonstrating that the proposed O-TBIBE scheme achieves the required security goals against computationally bounded adversaries.

A.1 Definition

The O-TBIBE scheme is a decentralized extension of the O-BIBE scheme, where the generation of the decryption key is distributed across multiple nodes of a decryption committee. In this threshold-based framework, a decryptor obtains decryption key shares from individual committee nodes and can reconstruct the original decryption key only when a threshold number of shares are successfully gathered. In the O-TBIBE framework, a trusted central authority initializes the system by generating master key shares, public key shares, a public encryption key (EK), and an aggregated public key (PK). The master key shares are securely distributed to the nodes of the decryption committee. Similar to the O-BIBE scheme, individual users utilize the EK to generate ciphertexts associated with their identity (ID) and a batch label (BL), which are then uploaded to a ciphertext pool. To perform decryption, a decryptor selects a batch of B ciphertexts, generates a digest DIG for the set S of associated identities, and requests decryption key shares DK_j from the committee nodes for that specific DIG. Upon collecting a sufficient number of shares, the decryptor reconstructs the original decryption key DK, which is then used to derive the capsule keys for the ciphertexts in set S. The process for outsourcing decryption to a cloud server remains identical to the mechanisms established in the O-BIBE scheme, ensuring functional consistency.

Following this operational flow, the formal syntax of the O-TBIBE scheme is specified as follows:

Definition A.1 (O-TBIBE)**.** An outsourced threshold batched identity-based encryption (O-TBIBE) scheme with a maximum batch size $B \in \mathbb{N}$, identity space of $\mathcal{I}$, batch label space of $\mathcal{L}$, and capsule key space of $\mathcal{K}$ consists of the following eight algorithms:

$\mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T) \to ((\mathsf{MK}_j)_{j\in[N]}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK})$: The setup algorithm takes as input the security parameter λ, the maximum batch size B, the size of the decryption committee N, and the threshold $T \leq N$. It outputs master key shares $(\mathsf{MK}_j)_{j\in[N]}$, public key shares $(\mathsf{PK}_j)_{j\in[N]}$, a public encryption key EK, and a public key PK.

$\mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}, \mathsf{BL}) \to (\mathsf{CH}, \mathsf{CK})$: The encapsulation algorithm takes as input the public encryption key EK, an identity $\mathsf{ID} \in \mathcal{I}$, and a batch label $\mathsf{BL} \in \mathcal{L}$. It outputs a ciphertext header CH and a capsule key $\mathsf{CK} \in \mathcal{K}$.

$\mathsf{Digest}(\mathsf{PK}, S) \to \mathsf{DIG}$: The digest algorithm takes as input the public key PK and a set of identities $S \subseteq \mathcal{I}$. It outputs a digest DIG associated with S.

$\mathsf{ComputeKeyShare}(\mathsf{MK}_j, \mathsf{DIG}, \mathsf{BL}) \to \mathsf{DK}_j$: The key share computation algorithm takes as input the master key share MK_j, a digest DIG, and a batch label BL. It outputs a decryption key share DK_j.

$\mathsf{VerifyKeyShare}(\mathsf{PK}_j, \mathsf{DK}_j, \mathsf{DIG}, \mathsf{BL}) \to \{0,1\}$: The key share verification algorithm takes as input the public key share PK_j, the decryption key share DK_j, a digest DIG, and a batch label BL. It outputs 1 or 0 depending on the validity of the decryption key share.

$\mathsf{CombineKeyShare}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, \mathsf{DIG}, \mathsf{BL}) \to \mathsf{DK}$: The key share combining algorithm takes as input the public key PK, decryption key shares $(\mathsf{DK}_j)_{j\in A}$ for a subset $A \subseteq [N]$ where $|A| = T$, a digest DIG, and a batch label BL. It outputs a combined decryption key DK.

$\mathsf{VerifyCombKey}(\mathsf{PK}, \mathsf{DK}, \mathsf{DIG}, \mathsf{BL}) \to \{0,1\}$: The combined key verification algorithm takes as input the public key PK, the combined decryption key DK, a digest DIG, and a batch label BL. It outputs 1 or 0 depending on the validity of the combined decryption key.

$\mathsf{Decaps}(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, \mathsf{ID}, \mathsf{BL}) \to \mathsf{CK}$: The decapsulation algorithm takes as input the public key PK, a decryption key DK, a set of identities S, a ciphertext header CH, an identity ID, and a batch label BL. It outputs the corresponding capsule key $\mathsf{CK} \in \mathcal{K}$.

$\mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$: The delegation algorithm takes as input the public key PK, a decryption key DK, ciphertext headers $(\mathsf{CH}_i)_{i\in U}$ for a subset $U \subseteq [B]$, and a batch label BL. It outputs a recovery key RK, a transformation key TK, and work tasks $(\mathsf{WT}_i)_{i\in U}$.

$\mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, (\mathsf{ID}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{TT}_i)_{i\in U}$: The transformation algorithm takes as input the public key PK, a transformation key TK, work tasks $(\mathsf{WT}_i)_{i\in U}$, identities $(\mathsf{ID}_i)_{i\in U}$, and a batch label BL. It outputs transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

$\mathsf{Recover}(\mathsf{RK}, (\mathsf{TT}_i)_{i\in U}) \to (\mathsf{CK}_i)_{i\in U}$: The recovery algorithm takes as input a recovery key RK and transformed tasks $(\mathsf{TT}_i)_{i\in U}$ where $U \subseteq [B]$. It outputs a sequence of capsule keys $(\mathsf{CK}_i)_{i\in U} \in \mathcal{K}^{|U|}$.

Definition A.2. An O-TBIBE scheme is correct if for any security parameter λ and any batch size $B \leq \mathrm{poly}(\lambda)$, the following two conditions hold:

- Correctness of Decryption: For any size N, any threshold $T \leq N$, any identity $\mathsf{ID} \in \mathcal{I}$, any batch label $\mathsf{BL} \in \mathcal{L}$, and any set $S \subseteq \mathcal{I}$ such that $\mathsf{ID} \in S$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Decaps}(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, & ((\mathsf{MK}_j)_{j\in[N]}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T) \\ \mathsf{ID}, \mathsf{BL}) = \mathsf{CK} & (\mathsf{CH}, \mathsf{CK}) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}, \mathsf{BL}) \\ & \mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S) \\ & \forall j \in A : \mathsf{DK}_j \leftarrow \mathsf{ComputeKeyShare}(\mathsf{MK}_j, \mathsf{DIG}, \mathsf{BL}) \\ & \mathsf{DK} \leftarrow \mathsf{CombineKeyShare}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, \mathsf{DIG}, \mathsf{BL}) \end{array}\right] = 1.$$

- Correctness of Outsourced Batch Decryption: For any size N, any threshold $T \leq N$, any $S = \{\mathsf{ID}_i\}_{i\in[B]} \subseteq \mathcal{I}^B$, any subset $A \subseteq [N]$ such that $|A| = T$, and any batch label $\mathsf{BL} \in \mathcal{L}$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Recover}(\mathsf{RK}, (\mathsf{TT}_i)_{i\in[B]}) & ((\mathsf{MK}_j)_{j\in[N]}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T) \\ = (\mathsf{CK}_i)_{i\in[B]} & \forall i \in [B] : (\mathsf{CH}_i, \mathsf{CK}_i) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}_i, \mathsf{BL}) \\ & \mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S) \\ & \forall j \in A : \mathsf{DK}_j \leftarrow \mathsf{ComputeKeyShare}(\mathsf{MK}_j, \mathsf{DIG}, \mathsf{BL}) \\ & \mathsf{DK} \leftarrow \mathsf{CombineKeyShare}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, \mathsf{DIG}, \mathsf{BL}) \\ & (\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in[B]}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in[B]}, \mathsf{BL}) \\ & (\mathsf{TT}_i)_{i\in[B]} \leftarrow \mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in[B]}, (\mathsf{ID}_i)_{i\in[B]}, \mathsf{BL}) \end{array}\right] = 1.$$

We define the security model for our O-TBIBE scheme by extending the adaptive security model for TBIBE introduced by GWWW, incorporating additional outsourcing oracles. To formally account for the

exposure of master key shares from compromised committee nodes, we adopt a static corruption model where the set of corrupted nodes is designated at the beginning of the game. In this adaptive and static corruption CPA model, the adversary first commits to the total number of committee nodes N, the threshold T, and a set of corrupted nodes $C \subset [N]$ such that $|C| < T$. Following this commitment, the challenger provides the adversary with the master key shares for the corrupted nodes $\{\mathsf{MK}_j\}_{j\in C}$, all public key shares $\{\mathsf{PK}_j\}_{j\in[N]}$, the public encryption key EK, and the public key PK. The adversary is then permitted to adaptively issue queries to the $\mathsf{OCreateDKS}$ and $\mathsf{ORevealDKS}$ oracles to obtain decryption key shares from any non-corrupted nodes, as well as the $\mathsf{ODelegate}$ oracle to observe the outsourced ciphertext values. The challenge phase proceeds similarly to that of the O-BIBE scheme except that the adversary submits a target identity ID^* and a target batch label BL^*. After the challenge, the adversary may continue to perform additional oracle queries before finally submitting a guess μ' for the challenge bit μ. To prevent trivial winning conditions, standard restrictions are applied to the share-exposure oracles, the adversary cannot collect decryption key shares for ID^* associated with BL^*. Notably, however, no restrictions are imposed on the delegation oracle queries.

A more rigorous definition of the selective security with static corruption CPA model for the O-TBIBE scheme is as follows:

Definition A.3 (Selective CPA Security with Static Corruptions of O-TBIBE)**.** The selective CPA security with static corruptions (SE-SC-CPA) of O-TBIBE is defined via the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}CPA}}_{\textit{O-TBIBE},\mathcal{A}}(\lambda, B)$ between a challenger and a PPT adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: $\mathcal{A}$ commits to a challenge identity-label pair $(\mathsf{ID}^*, \mathsf{BL}^*)$, the size of the decryption committee N, the threshold $T \leq N$, and the indices of the corrupted nodes $C \subseteq [N]$ of the decryption committee where $|C| < T$.

2. **Setup**: The challenger runs $((\mathsf{MK}_j)_{j\in[N]}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T)$ and initializes the experiment by running the following oracle:

   - $\mathsf{OInit}()$: This oracle initializes two empty lists, $L_{\text{key}} \leftarrow \emptyset$ and $L_{\text{del}} \leftarrow \emptyset$, which are used to track key share exposure and delegation queries, respectively.

   Next, the challenger gives $((\mathsf{MK}_j)_{j\in C}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.

3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{OCreateDKS}$, $\mathsf{ORevealDKS}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRK}$ queries. The challenger handles these queries as follows:

   - $\mathsf{OCreateDKS}(S, \mathsf{BL})$: If there exists $(S, \mathsf{BL}, \cdot, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
     (a) It computes honest key shares $\mathsf{DKS} = (\mathsf{DK}_j)_{j\in[N]\setminus C}$ by using uncorrupted $(\mathsf{MK}_j)_{j\in[N]\setminus C}$.
     (b) Next, it picks a unique key handle $\mathsf{hk} \in \{0,1\}^\lambda$, updates $L_{\text{key}} = L_{\text{key}} \cup \{(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS})\}$, and responds with hk.
   - $\mathsf{ORevealDKS}(\mathsf{hk})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise, it retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS}) \leftarrow L_{\text{key}}$ and responds with DKS.
   - $\mathsf{ODelegate}(\mathsf{hk}, (\mathsf{CH}_i)_{i\in U})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
     (a) It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS})$ from L_{key} and computes the corrupted key shares $(\mathsf{DK}_j)_{j\in C}$ by using the corrupted $(\mathsf{MK}_j)_{j\in C}$.
     (b) It derives DK by combining the key shares $(\mathsf{DK}_j)_{j\in A}$ where $A \subseteq [N]$ is an arbitrary index set of size T. It computes $(\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in U}, \mathsf{BL})$.

(c) It picks a unique delegation handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\text{del}} \leftarrow L_{\text{del}} \cup \{(\mathsf{hd}, \mathsf{RK})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U})$.

- $\mathsf{ORevealRK}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\text{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd}, \mathsf{RK}) \leftarrow L_{\text{del}}$ and responds with RK.

4. **Challenge**: The challenger computes a ciphertext pair $(\mathsf{CH}^*, \mathsf{CK}^*) \leftarrow \mathsf{Encaps}(\mathsf{EK}, \mathsf{ID}^*, \mathsf{BL}^*)$. It sets $\mathsf{CK}_0^* = \mathsf{CK}^*$ and $\mathsf{CK}_1^* \xleftarrow{\$} \mathcal{K}$. It flips a random coin $\mu \in \{0,1\}$ and gives $(\mathsf{CH}^*, \mathsf{CK}_\mu^*)$ to $\mathcal{A}$.

5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. The challenger handles these queries exactly as in the pre-challenge queries.

6. **Output**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. The experiment returns 1 if $\mu' = \mu$ and the following oracle constraints are satisfied:

   (a) **(Challenge Privacy)**: For the challenge label BL^* and any set S such that $\mathsf{ID}^* \in S$, the adversary $\mathcal{A}$ is restricted as follows:

   - $\mathcal{A}$ never queries $\mathsf{ORevealDKS}(\mathsf{hk})$ for the key handle hk associated with (S, BL^*).
   - $\mathcal{A}$ never queries $\mathsf{ORevealRK}(\mathsf{hd})$ for any delegation handle hd that was generated from the key handle hk associated with (S, BL^*).

   (b) **(Label Distinctness)**: The adversary $\mathcal{A}$ is required to submit distinct batch labels BL for all $\mathsf{OCreateDKS}$ queries. This constraint ensures that for the challenge label BL^*, $\mathcal{A}$ must choose exactly one of the following scenarios at the time of the $\mathsf{OCreateDKS}(S, \mathsf{BL}^*)$ query:

   - Case ($\mathsf{ID}^* \notin S$): $\mathcal{A}$ is allowed to query $\mathsf{ORevealDKS}(\mathsf{hk})$, $\mathsf{ODelegate}(\mathsf{hk}, \cdot)$, and $\mathsf{ORevealRK}(\mathsf{hd})$ for this BL^*.
   - Case ($\mathsf{ID}^* \in S$): $\mathcal{A}$ is allowed to query $\mathsf{ODelegate}(\mathsf{hk}, \cdot)$, but is strictly prohibited from querying $\mathsf{ORevealDKS}(\mathsf{hk})$ or $\mathsf{ORevealRK}(\mathsf{hd})$ for this BL^*, as specified by the challenge privacy.

   Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CPA}}_{\textit{O-TBIBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}CPA}}_{\textit{O-TBIBE},\mathcal{A}}(\lambda, B) = 1] - \frac{1}{2} \right|$ where the probability is taken over all internal randomness of the experiment. An O-TBIBE scheme is said to be SE-SC-CPA secure if the advantage is negligible for all PPT adversaries.

A.2 Construction

The proposed O-TBIBE scheme is built upon the GWWW-TBIBE scheme [26], which utilizes the key structure of the BF-IBE scheme. In a manner analogous to our transformation of the BIBE scheme into O-BIBE, we apply several structural refinements to the O-TBIBE scheme to optimize its practical deployment. First, we consolidate the traditional Setup and KeyGen algorithms into a single, unified Setup algorithm. Second, we separate the master public parameters into a public encryption key (EK) and an aggregated public key (PK) to minimize the storage overhead on the encryptor. Finally, we transition from the standard PKE paradigm to a KEM framework to enhance the modularity of the O-TBIBE scheme.

The basic algorithms of our O-TBIBE scheme excluding the outsourcing algorithms are described as follows:

O-TBIBE.Setup$(1^\lambda, 1^B, 1^N, 1^T)$**:** It first samples an asymmetric bilinear group $\mathsf{GD} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g}) \leftarrow \mathcal{G}(1^\lambda)$. It selects random exponents $\tau, \alpha, w \in \mathbb{Z}_p$. Let $\mathbf{M} \in \mathbb{Z}_p^{N \times T}$ be the share-generation matrix for a T-out-of-N threshold policy over $\mathbb{Z}_p$. It samples random vectors $\mathbf{a}, \mathbf{w} \in \mathbb{Z}_p^N$ where $a_1 = \alpha$ and $w_1 = w$. For each $j \in [N]$, it sets $\alpha_j = \mathbf{m}_j^\top \mathbf{a}$ and $w_j = \mathbf{m}_j^\top \mathbf{w}$ where $\mathbf{m}_j^\top$ is the jth row of $\mathbf{M}$. It sets a hash function $\mathsf{H} : \{0,1\}^* \to \hat{\mathbb{G}}$ that is modeled as a random oracle. It outputs the master key shares $(\mathsf{MK}_j = (\mathsf{GD}, \alpha_j, w_j, \mathsf{H}))_{j \in [N]}$, the public key shares $(\mathsf{PK}_j = (\mathsf{GD}, g^{\alpha_j}, g^{w_j}, \mathsf{H}))_{j \in [N]}$, the public encryption key, and the public key as

$$\mathsf{EK} = \big(\mathsf{GD}, g^\alpha, g^w, g^{\tau w}, \mathsf{H}\big),\ \mathsf{PK} = \big(\mathsf{GD}, g^\tau, g^\alpha, g^w, \hat{g}^\tau, \hat{g}^{\tau^2}, \ldots, \hat{g}^{\tau^B}, \mathsf{H}\big).$$

O-TBIBE.Encaps$(\mathsf{EK}, \mathsf{ID}, \mathsf{BL})$**:** Let $\mathsf{EK} = (\mathsf{GD}, g^\alpha, g^w, g^{\tau w}, \mathsf{H})$. It samples a random exponent $t \in \mathbb{Z}_p$ and outputs a ciphertext header

$$\mathsf{CH} = \big(C_1 = g^t,\ C_2 = (g^{\tau w})^t (g^w)^{-\mathsf{ID}t} = g^{(\tau - \mathsf{ID})wt}\big)$$

and a capsule key $\mathsf{CK} = e(g^\alpha, \mathsf{H}(\mathsf{BL}))^t$.

O-TBIBE.Digest(PK, S)**:** Let $S = \{\mathsf{ID}_i\}_{i \in U}$ where $U \subseteq [B]$. It defines $F_S(x) = \prod_{\mathsf{ID}_i \in S} (x - \mathsf{ID}_i) = \sum_{k \in [0, |S|]} f_k x^k$ and outputs the digest

$$\mathsf{DIG} = \prod_{k \in [0, |S|]} \big(\hat{g}^{\tau^k}\big)^{f_k} = \hat{g}^{F_S(\tau)}.$$

O-TBIBE.ComputeKeyShare$(\mathsf{MK}_j, \mathsf{DIG}, \mathsf{BL})$**:** Let $\mathsf{MK}_j = (\mathsf{GD}, \alpha_j, w_j, \mathsf{H})$. It outputs a decryption key share

$$\mathsf{DK}_j = \big(K_j = \mathsf{H}(\mathsf{BL})^{\alpha_j} \mathsf{DIG}^{w_j}\big).$$

O-TBIBE.VerifyKeyShare$(\mathsf{PK}_j, \mathsf{DK}_j, \mathsf{DIG}, \mathsf{BL})$**:** Let $\mathsf{PK}_j = (\mathsf{GD}, g^{\alpha_j}, g^{w_j}, \mathsf{H})$ and $\mathsf{DK}_j = K_j$. It outputs 1 if the equation holds

$$e(g, K_j) \stackrel{?}{=} e(g^{\alpha_j}, \mathsf{H}(\mathsf{BL})) \cdot e(g^{w_j}, \mathsf{DIG})$$

and 0 otherwise.

O-TBIBE.CombineKeyShare$((\mathsf{DK}_j)_{j \in A}, \mathsf{DIG}, \mathsf{BL})$**:** Let $\mathsf{DK}_j = K_j$ for $j \in A$ where $|A| = T$. It computes the interpolation vector $\boldsymbol{\lambda}_A = (\lambda_{j,A})_{j \in [N]} \in \mathbb{Z}_p^N$ such that $\boldsymbol{\lambda}_A^\top \mathbf{M} = \mathbf{e}_1^\top$ and $\lambda_{j,A} = 0$ for all $j \notin A$ where $\mathbf{M}$ is the share-generation matrix for the T-out-of-N threshold policy. It outputs the combined decryption key

$$\mathsf{DK} = (K = \prod_{j \in A} K_j^{\lambda_{j,A}}).$$

O-TBIBE.VerifyCombKey$(\mathsf{PK}, \mathsf{DK}, \mathsf{DIG}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{GD}, g^\tau, g^\alpha, g^w, \cdots, \mathsf{H})$ and $\mathsf{DK} = K$. It outputs 1 if the equation holds

$$e(g, K) \stackrel{?}{=} e(g^\alpha, \mathsf{H}(\mathsf{BL})) \cdot e(g^w, \mathsf{DIG})$$

and 0 otherwise.

O-TBIBE.Decaps$(\mathsf{PK}, \mathsf{DK}, S, \mathsf{CH}, \mathsf{ID}, \mathsf{BL})$**:** Let $\mathsf{DK} = K$ and $\mathsf{CH} = (C_1, C_2)$ where DK is associated with (S, BL), CH is associated with $(\mathsf{ID}, \mathsf{BL})$, and $\mathsf{ID} \in S$. It defines $F_{S \setminus \{\mathsf{ID}\}}(x) = \prod_{\mathsf{ID}_k \in S \setminus \{\mathsf{ID}\}} (x - \mathsf{ID}_k) = \sum_{k \in [0, |S|-1]} \tilde{f}_k x^k$ and computes the witness

$$\pi = \prod_{k \in [0, |S|-1]} \big(\hat{g}^{\tau^k}\big)^{\tilde{f}_k} = \hat{g}^{F_{S \setminus \{\mathsf{ID}\}}(\tau)}.$$

It outputs the capsule key $\mathsf{CK} = e(C_1, K) \cdot e(C_2, \pi)^{-1}$.

To incorporate outsourcing capabilities into GWWW-TBIBE, we apply the same re-randomization mechanism developed for our O-BIBE scheme. A significant advantage of this approach lies in the structural simplicity of the decryption keys. Since a decryption key in the GWWW-TBIBE scheme consists of only a single group element, the re-randomization process is considerably more streamlined than in other identity-based variants. This efficiency allows the decryptor to generate transformation keys with minimal computational cost, even in a distributed environment. The detailed outsourcing algorithms of our O-TBIBE schemes are described as follows:

O-TBIBE.Delegate$(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i\in U}, \mathsf{BL})$**:** Let $\mathsf{DK} = K$ and $\mathsf{CH}_i = (C_{i,1}, C_{i,2})$ for $i \in U$ where $U \subseteq [B]$. It samples $z_1 \in \mathbb{Z}_p$ uniformly at random and sets a recover key $\mathsf{RK} = z_1$. Then, it constructs a transformation key

$$\mathsf{TK} = \big(V = K \cdot \hat{g}^{z_1}\big).$$

It sets a work task $\mathsf{WT}_i = \big(D_{i,1} = C_{i,1},\ D_{i,2} = C_{i,2}\big)$ for each $i \in U$. It outputs $(\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.

O-TBIBE.Transform$(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, (\mathsf{ID}_i)_{i\in U}, \mathsf{BL})$**:** Let $\mathsf{TK} = V$ and $\mathsf{WT}_i = (D_{i,1}, D_{i,2})$. It sets $S = \{\mathsf{ID}_i\}_{i\in U}$. For each $i \in U$, it defines $F_{S\setminus\{\mathsf{ID}_i\}}(x) = \prod_{\mathsf{ID}_k \in S\setminus\{\mathsf{ID}_i\}} (x - \mathsf{ID}_k) = \sum_{k\in[0,|S|-1]} \tilde{f}_{i,k} x^k$ and computes the witness

$$\pi_i = \prod_{k\in[0,|S|-1]} \big(\hat{g}^{\tau^k}\big)^{\tilde{f}_{i,k}} = \hat{g}^{F_{S\setminus\{\mathsf{ID}_i\}}(\tau)}.$$

For each $i \in U$, it constructs a transformed task

$$\mathsf{TT}_i = \big(E_{i,1} = e(D_{i,1}, V) \cdot e(D_{i,2}, \pi_i)^{-1},\ E_{i,2} = e(D_{i,1}, \hat{g})\big).$$

It outputs transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

O-TBIBE.Recover$(\mathsf{RK}, (\mathsf{TT}_i)_{i\in U})$**:** Let $\mathsf{RK} = z_1$ and $\mathsf{TT}_i = (E_{i,1}, E_{i,2})$ for $i \in U$. For each $i \in U$, it recovers $\mathsf{CK}_i = E_{i,1} \cdot E_{i,2}^{-z_1}$. It outputs capsule keys $(\mathsf{CK}_i)_{i\in U}$.

A.3 Correctness

Theorem A.1. *The proposed O-TBIBE scheme satisfies the correctness.*

Proof. The correctness of decryption is easily derived from the property of the underlying TBIBE scheme as follows:

$$\begin{aligned} & e(C_{i,1}, K) \cdot e(C_{i,2}, \pi_i)^{-1} \\ &= e(g^t, \mathsf{H}(\mathsf{BL})^\alpha \big(\hat{g}^{F_S(\tau)}\big)^w) \cdot e\big(g^{(\tau - \mathsf{ID}_i)wt}, \hat{g}^{F_{S\setminus\{\mathsf{ID}_i\}}(\tau)}\big)^{-1} \\ &= e(g^t, \mathsf{H}(\mathsf{BL})^\alpha) = \mathsf{CK}_i. \end{aligned}$$

Next, we show the correctness of outsourced batch decryption. For each $i \in [B]$, let $\mathsf{TT}_i = (E_{i,1}, E_{i,2})$ be the output of the O-TBIBE.Transform algorithm. By substituting the components of $\mathsf{TK} = V$ and $\mathsf{WT}_i = (D_{i,1}, D_{i,2})$, we obtain the following equation:

$$\begin{aligned} E_{i,1} &= e(D_{i,1}, V) \cdot e(D_{i,2}, \pi_i)^{-1} = e(C_{i,1}, K \cdot \hat{g}^{z_1}) \cdot e(C_{i,2}, \pi_i)^{-1} \\ &= \big(e(C_{i,1}, K) \cdot e(C_{i,2}, \pi_i)^{-1}\big) \cdot e(C_{i,1}, \hat{g})^{z_1} = \mathsf{CK}_i \cdot e(g, \hat{g})^{t \cdot z_1}. \end{aligned}$$

By using the recover key $\mathsf{RK} = z_1$, the O-TBIBE.Recover algorithm derives the following equation:

$$E_{i,1} \cdot E_{i,2}^{-z_1} = \left(\mathsf{CK}_i \cdot e(g,\hat{g})^{t \cdot z_1}\right) \cdot e(g,\hat{g})^{-t \cdot z_1} = \mathsf{CK}_i.$$

This completes the proof. □

A.4 Security Analysis

Theorem A.2. *Let TBIBE-KEM be the core scheme obtained by excluding the outsourcing algorithms (Delegate, Transform, Recover) from our proposed O-TBIBE scheme. The TBIBE-KEM scheme is SE-SC-CPA secure for any polynomial batch size B in the generic group model.*

Proof Sketch. The SE-SC-CPA security of the proposed TBIBE-KEM scheme is fundamentally rooted in the security properties of the GWWW-TBIBE scheme [26]. Although our scheme introduces three structural modifications, we argue that the simulation remains sound within the GGM.

First, the consolidation of Setup and KeyGen into a single Setup algorithm is a functional simplification that does not alter the distribution of the system parameters or the master secret key. Since the underlying algebraic relations between the generated elements remain identical to those in GWWW-TBIBE, this integration has no impact on the security reduction.

Second, our scheme partitions the original public key structure into an encryption key (EK) and a public key (PK). This separation is a structural refinement where elements specifically required for the encapsulation process are grouped into EK. Since these components are essentially a redistribution of the original public key elements from the GWWW-TBIBE scheme, this decomposition does not introduce any new algebraic dependencies or leak additional information about the master secret key. In the GGM, the adversary is provided with the same underlying group elements as in the original scheme; thus, the functional separation of EK for the Encaps algorithm does not alter the hardness of the underlying problems.

Third, the transition from PKE to a KEM is algebraically transparent. The encapsulation process in TBIBE-KEM utilizes the same bilinear map operations and randomization techniques as the original GWWW-TBIBE encryption. The capsule key CK^* is derived from the same masking principle used to encrypt messages in PKE. Therefore, the indistinguishability of CK^* from a random element in $\mathbb{G}_T$ follows directly from the hardness of the underlying problems in the GGM.

In conclusion, because the modified public key and the KEM-based structure can be perfectly simulated without introducing unintended algebraic dependencies, the SE-SC-CPA security of our TBIBE-KEM scheme is preserved. □

Theorem A.3. *The proposed O-TBIBE scheme satisfies SE-SC-CPA security for any polynomial batch size B if the underlying TBIBE-KEM scheme is SE-SC-CPA secure.*

Proof. To prove the SE-SC-CPA security of the O-TBIBE scheme, we employ a sequence of hybrid games similar to the SE-IND-CPA proof of the O-BIBE scheme in Theorem 3.3. We define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$, and $\mathbf{G}_3$, where $\mathbf{G}_0$ is the original experiment and $\mathbf{G}_3$ is a game in which the adversary has zero advantage. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_j}$ denote the advantage of $\mathcal{A}$ in $\mathbf{G}_j$.

Game $\mathbf{G}_0$. This game corresponds to the original SE-SC-CPA security experiment for O-TBIBE. The simulator follows the honest algorithms to handle all queries and generate the challenge ciphertext. By definition, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} = \mathsf{Adv}_{\mathit{O\text{-}TBIBE},\mathcal{A}}^{\mathsf{SE\text{-}SC\text{-}CPA}}(\lambda)$.

Game $\mathbf{G}_1$. This game refines the oracle simulation of $\mathbf{G}_0$ by incorporating a selective simulated sampling. The simulator $\mathcal{B}$ manages the queries as follows:

- In $\mathsf{OCreateDKS}$, $\mathcal{B}$ generates the key shares DKS using MK_j if $\mathsf{BL} \neq \mathsf{BL}^*$ or $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S$. Otherwise involving $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \in S$, $\mathcal{B}$ leaves the key shares entry empty ($\perp$) to facilitate lazy processing.
- In $\mathsf{ODelegate}$, $\mathcal{B}$ performs the original outsourcing algorithm if a valid DKS exists in the list. Otherwise, it generates a simulated transformation key via random sampling to maintain functional consistency.
- In $\mathsf{ORevealDKS}$ and $\mathsf{ORevealRK}$, $\mathcal{B}$ responds with the requested key only if it exists in the respective lists (L_{key} or L_{del}). By the challenge privacy constraint of the security model, the other case does not occur.

As established in Lemma A.4, the view of $\mathcal{A}$ is identical to $\mathbf{G}_0$. Thus, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}$.

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$ except that the simulator $\mathcal{B}$ no longer utilizes the master key shares MK_j to generate decryption key shares. Instead, $\mathcal{B}$ leverages the key share computation oracle $\mathsf{OComputeKeyShare}(\cdot)$ provided by the underlying TBIBE-KEM scheme. Specifically, in the $\mathsf{OCreateDKS}$ oracle, $\mathcal{B}$ obtains the required key shares DKS by issuing a query to $\mathsf{OComputeKeyShare}$ (S, BL) for the following cases: (i) when $\mathsf{BL} \neq \mathsf{BL}^*$, and (ii) when $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S$. Since the key shares obtained through $\mathsf{OComputeKeyShare}$ are identically distributed to those computed using MK_j, the adversary's view remains invariant. Consequently, the advantage of $\mathcal{A}$ remains unchanged: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}$.

Game $\mathbf{G}_3$. This game is identical to $\mathbf{G}_2$ except for the generation of the challenge capsule key CK^*. In the challenge phase, the simulator replaces CK^* with a value chosen uniformly at random from the key space $\mathbb{G}_T$, independent of the challenge bit μ. The indistinguishability between $\mathbf{G}_2$ and $\mathbf{G}_3$ is reduced to the SE-SC-CPA security of the underlying TBIBE-KEM scheme. Specifically, any adversary capable of distinguishing $\mathbf{G}_2$ from $\mathbf{G}_3$ can be transformed into a simulator $\mathcal{B}$ that breaks the TBIBE-KEM security. It follows that $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}\right| \leq \mathsf{Adv}_{TBIBE,\mathcal{B}}^{\mathsf{SE\text{-}SC\text{-}CPA}}(\lambda)$.

Since the challenge capsule key in $\mathbf{G}_3$ is purely random and carries no information regarding μ, the advantage of $\mathcal{A}$ is exactly zero: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} = 0$.

By summing the advantage differences across the games, we obtain the following upper bound:

$$\mathsf{Adv}_{O\text{-}TBIBE,\mathcal{A}}^{\mathsf{SE\text{-}SC\text{-}CPA}}(\lambda) \leq \sum_{j=1}^{3} \left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{j-1}} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_j}\right| + \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} \leq \mathsf{Adv}_{TBIBE,\mathcal{B}}^{\mathsf{SE\text{-}SC\text{-}CPA}}(\lambda).$$

This completes the proof. □

Lemma A.4. *The games $\boldsymbol{G}_0$ and $\boldsymbol{G}_1$ are statistically indistinguishable.*

Proof. In game $\mathbf{G}_1$, we refine the oracle simulation logic to handle queries involving the challenge identity and label more strategically. For all queries where the batch label BL differs from the challenge label BL^*, or where $\mathsf{BL} = \mathsf{BL}^*$ but the challenge identity ID^* is not included in the set S, the simulator $\mathcal{C}$ proceeds identically to $\mathbf{G}_0$ by computing honest key shares using the master secret keys. A key departure occurs specifically for queries satisfying $\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \in S$, where the simulator refrains from generating any actual key shares during the $\mathsf{OCreateDKS}$ query and instead sets the internal state $\mathsf{DKS} = \perp$. Consequently, if a subsequent $\mathsf{ODelegate}$ query is issued for a handle hk associated with this $\perp$ state, the simulator performs random sampling to generate a transformation key TK and its associated components, thereby providing

a functionally consistent response without requiring access to the underlying key shares. Due to the challenge privacy constraint, any queries that would expose these simulated states—namely $\mathsf{ORevealDKS}$ or $\mathsf{ORevealRK}$ for the challenge target—are strictly prohibited.

The detailed procedures for each oracle in $\mathbf{G}_1$ are given as follows:

- $\mathsf{OCreateDKS}(S, \mathsf{BL})$: If there exists an entry $(S, \mathsf{BL}, \cdot, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. If $(\mathsf{BL} \neq \mathsf{BL}^*)$ or $(\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \notin S)$: It derives $\mathsf{DIG} \leftarrow \mathsf{Digest}(\mathsf{PK}, S)$ and computes $\mathsf{DK}_j \leftarrow \mathsf{ComputeKeyShare}(\mathsf{MK}_j, \mathsf{DIG}, \mathsf{BL})$ for all $j \in [N] \setminus C$. It sets $\mathsf{DKS} = (\mathsf{DK}_j)_{j \in [N] \setminus C}$.
  2. Otherwise $(\mathsf{BL} = \mathsf{BL}^* \wedge \mathsf{ID}^* \in S)$: It sets $\mathsf{DKS} = \perp$.
  3. It picks a unique key handle $\mathsf{hk} \in \{0,1\}^\lambda$, updates $L_{\text{key}} \leftarrow L_{\text{key}} \cup \{(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS})\}$, and responds with hk.
- $\mathsf{ORevealDKS}(\mathsf{hk})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS})$ from L_{key} and responds with DKS. Note that the case $(\mathsf{DKS} = \perp)$ cannot occur due to the challenge privacy constraint.
- $\mathsf{ODelegate}(\mathsf{hk}, (\mathsf{CH}_i)_{i \in U})$: If there is no entry $(\cdot, \cdot, \mathsf{hk}, \cdot) \in L_{\text{key}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(S, \mathsf{BL}, \mathsf{hk}, \mathsf{DKS})$ from L_{key}.
  2. If $\mathsf{DKS} \neq \perp$: Let $\mathsf{DKS} = (\mathsf{DK}_j)_{j \in [N] \setminus C}$.
     (a) It computes corrupted key shares $(\mathsf{DK}_j)_{j \in C}$ by using the corrupted $(\mathsf{MK}_j)_{j \in C}$.
     (b) It derives DK by combining the key shares $(\mathsf{DK}_j)_{j \in A}$ where $A \subseteq [N]$ is an arbitrary index set of size T.
     (c) It computes $(\mathsf{RK}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, \mathsf{DK}, (\mathsf{CH}_i)_{i \in U}, \mathsf{BL})$.
  3. Otherwise $(\mathsf{DKS} = \perp)$: It samples $V \in \hat{\mathbb{G}}$ uniformly at random. It sets $\mathsf{RK} = \perp$ and $\mathsf{TK} = V$. It derives $(\mathsf{WT}_i)_{i \in U}$ from $(\mathsf{CH}_i)_{i \in U}$ accordingly.
  4. It picks a unique delegation handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\text{del}} \leftarrow L_{\text{del}} \cup \{(\mathsf{hd}, \mathsf{RK})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i \in U})$.
- $\mathsf{ORevealRK}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\text{del}}$, it responds with $\perp$. Otherwise:
  1. It retrieves $(\mathsf{hd}, \mathsf{RK})$ from L_{del} and responds with RK. Note that the case $(\mathsf{RK} = \perp)$ cannot occur due to the challenge privacy constraint.

To establish the statistical indistinguishability, we consider a computationally unbounded (statistical) adversary $\mathcal{A}$. From the observed transcripts of $\mathsf{ODelegate}$, $\mathcal{A}$ obtains the transformation key $\mathsf{TK} = V$ and the tasks $\mathsf{WT}_i = (D_{i,1}, D_{i,2})$. The adversary's view is entirely characterized by the value $\log_{\hat{g}}(V)$. The mapping from the internal random z_1 to the observable component $\log_{\hat{g}}(V)$ can be expressed as the following equation:

$$\log_{\hat{g}}(V) = \log_{\hat{g}}(K) + z_1.$$

This mapping constitutes a perfect bijection over the respective domain. In $\mathbf{G}_1$, the simulator performs direct sampling of V. Due to the one-to-one correspondence between the observable element and the random variable z_1, the distribution of the adversary's view is identical in both games. Consequently, $\mathsf{Adv}^{\mathbf{G}_0}_{\mathcal{A}} = \mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}$, completing the proof. □

Lemma A.5. *The games $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are identical.*

Proof. The only difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ lies in the method of generating the decryption key shares (DKS). In $\mathbf{G}_1$, $\mathcal{B}$ computes DK_j directly using MK_j, whereas in $\mathbf{G}_2$, it invokes the key share computation oracle $\mathsf{OComputeKeyShare}$ of the TBIBE-KEM scheme. According to the correctness of the TBIBE-KEM key share generation, the output of $\mathsf{OComputeKeyShare}$ for a given (S, BL) is equivalent to the outputs of the $\mathsf{ComputeKeyShare}$ algorithm using MK_j for all $j \in [N] \setminus C$. Consequently, the distribution of all oracle responses remains invariant between the two games, making them perfectly indistinguishable. □

Lemma A.6. *The games $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ are computationally indistinguishable, assuming that the underlying TBIBE-KEM scheme is SE-SC-CPA secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the security of the SE-SC-CPA game of O-TBIBE with a non-negligible advantage. We construct a simulator $\mathcal{B}$ that breaks the underlying TBIBE-KEM scheme by invoking $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ commits to a challenge identity-label pair $(\mathsf{ID}^*, \mathsf{BL}^*)$, the size of the decryption committee N, the threshold $T \leq N$, and the indices of the corrupted nodes $C \subseteq [N]$ where $|C| < T$. $\mathcal{B}$ forwards $(\mathsf{ID}^*, \mathsf{BL}^*, N, T, C)$ to the TBIBE-KEM challenger and receives $((\mathsf{MK}_{j,TB})_{j\in C}, (\mathsf{PK}_{j,TB})_{j\in[N]}, \mathsf{EK}_{TB}, \mathsf{PK}_{TB})$.

Setup: The simulator $\mathcal{B}$ initializes the simulation environment by invoking the $\mathsf{OInit}(\cdot)$ oracle as defined in $\mathbf{G}_2$. This procedure establishes the initial state, including the initialization of the lists L_{key} and L_{del} to empty sets. Furthermore, $\mathcal{B}$ sets the O-TBIBE public parameters using the underlying TBIBE-KEM instance by setting $(\mathsf{MK}_j = \mathsf{MK}_{j,TB})_{j\in C}, (\mathsf{PK}_j = \mathsf{PK}_{j,TB})_{j\in[N]}, \mathsf{EK} = \mathsf{EK}_{TB}$, and $\mathsf{PK} = \mathsf{PK}_{TB}$. Finally, $\mathcal{B}$ provides $((\mathsf{MK}_j)_{j\in C}, (\mathsf{PK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK})$ to the adversary $\mathcal{A}$.

Pre-challenge Queries: The simulator $\mathcal{B}$ responds to the adversary's queries by following the exact same procedures defined in $\mathbf{G}_2$. Specifically, for all queries requiring the computation of decryption key shares (DKS) in the $\mathsf{OCreateDKS}$ simulation in $\mathbf{G}_2$, $\mathcal{B}$ does not utilize the master key shares MK_j (which it does not possess). Instead, it obtains the required decryption key shares by issuing key share computation queries to the underlying TBIBE-KEM $\mathsf{OComputeKeyShare}$ oracle. Since the distribution of the oracle responses is identical to that in the previous game, the simulation remains consistent and indistinguishable from the perspective of $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives the challenge TBIBE ciphertext pair $(\mathsf{CH}^*_{TB}, \mathsf{CK}^*_{TB})$ from the TBIBE-KEM challenger. It sets $\mathsf{CH}^* = \mathsf{CH}^*_{TB}$ and $\mathsf{CK}^* = \mathsf{CK}^*_{TB}$, and gives the pair $(\mathsf{CH}^*, \mathsf{CK}^*)$ to $\mathcal{A}$

Post-challenge Queries: $\mathcal{B}$ responds to these queries in the same manner as the pre-challenge queries, subject to the standard security game restrictions.

Output: $\mathcal{A}$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'.

The advantage of $\mathcal{B}$ in the TBIBE-KEM security game is derived as follows: If the TBIBE-KEM challenger's bit is $b = 0$, the adversary $\mathcal{A}$'s view is identical to $\mathbf{G}_2$. If the TBIBE-KEM challenger's bit is $b = 1$, the adversary $\mathcal{A}$'s view is identical to $\mathbf{G}_3$. Therefore, $\mathcal{B}$'s advantage in winning the TBIBE-KEM game is exactly equal to $\mathcal{A}$'s advantage in distinguishing $\mathbf{G}_2$ from $\mathbf{G}_3$. □

B Verifiable Outsourced BTE

In this appendix, we introduce the formal syntax and security model for VOBTE, which supports threshold decryption of ciphertexts in an outsourced environment. We first establish a robust security framework that incorporates the requirements for verifiable batching alongside threshold-specific constraints. Leveraging

the O-TBIBE scheme introduced in the previous section, we design a concrete VOBTE scheme that enables secure delegation and verifiable decryption under decentralized trust. We then provide a rigorous security analysis to demonstrate that the security of our VOBTE system is reducible to the security of its underlying cryptographic primitives and the proven properties of the O-TBIBE scheme. This extension finalizes our framework by providing a comprehensive solution that achieves both high efficiency through batching and enhanced reliability through threshold mechanisms.

B.1 Definition

Definition B.1 (VOBTE)**.** A verifiable outsourced batched threshold encryption (VOBTE) scheme with a maximum batch size $B \in \mathbb{N}$, message space of $\mathcal{M}$, and batch label space of $\mathcal{L}$ consists of the following seven algorithms. Let $\mathcal{M}_{\perp\!\!\!\perp} = \mathcal{M} \cup \{\perp, \perp\!\!\!\perp\}$ denote the extended message space, which includes the verification failure symbol $\perp$ (resulting from invalid input or dummy padding) and the transformation failure symbol $\perp\!\!\!\perp$.

$\mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T) \to ((\mathsf{MK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK})$: The setup algorithm takes as input the security parameter λ, the maximum batch size B, the size of the decryption committee N, and the threshold T. It outputs master key shares $(\mathsf{MK}_j)_{j\in[N]}$, a public encryption key EK, and a public key PK.

$\mathsf{Encrypt}(\mathsf{EK}, M, \mathsf{BL}) \to \mathsf{CT}$: The encryption algorithm takes as input the public encryption key EK, a message $M \in \mathcal{M}$, and a batch label $\mathsf{BL} \in \mathcal{L}$. It outputs a ciphertext CT.

$\mathsf{PreDecrypt}(\mathsf{PK}, \mathsf{MK}_j, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \to \mathsf{DK}_j$: The pre-decryption algorithm takes as input the public key PK, the master key share MK_j, ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$, and a batch label BL. It outputs a decryption key share DK_j.

$\mathsf{Decrypt}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \to (M_i)_{i\in[B]} \cup \{\perp\}$: The (deterministic) decryption algorithm takes as input the public key PK, decryption key shares $(\mathsf{DK}_j)_{j\in A}$ for a subset $A \subseteq [N]$ where $|A| = T$, and ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$. It outputs the corresponding messages $(M_i)_{i\in[B]} \in \mathcal{M}_\perp^B$ or $\perp$ to indicate failure.

$\mathsf{Delegate}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \to (\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \cup \{\perp\}$: The delegation algorithm takes as input the public key PK, decryption key shares $(\mathsf{DK}_j)_{j\in A}$ for a subset $A \subseteq [N]$ where $|A| = T$, and ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$. It outputs a recovery token RT, a transformation key TK, and work tasks $(\mathsf{WT}_i)_{i\in U}$ where $U \subseteq [B]$, or $\perp$ to indicate failure.

$\mathsf{Transform}(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, \mathsf{BL}) \to (\mathsf{TT}_i)_{i\in U}$: The transformation algorithm takes as input the public key PK, the transformation key TK, the work tasks $(\mathsf{WT}_i)_{i\in U}$, and a batch label BL. It outputs transformed tasks $(\mathsf{TT}_i)_{i\in U}$.

$\mathsf{Recover}(\mathsf{RT}, (\mathsf{TT}_i)_{i\in U}) \to (M_i)_{i\in[B]}$: The (deterministic) recovery algorithm takes as input the recovery token RT, and the transformed tasks $(\mathsf{TT}_i)_{i\in U}$. It outputs the recovered messages $(M_i)_{i\in[B]} \in \mathcal{M}_{\perp\!\!\!\perp}^B$.

Definition B.2 (Correctness of VOBTE)**.** A VOBTE scheme is correct if for any security parameter λ and any batch size $B \le \mathrm{poly}(\lambda)$, the following two conditions hold:

- Correctness of Batch Decryption: For any N, any $T \le N$, any messages $(M_i)_{i\in[B]} \in \mathcal{M}^B$, any batch label $\mathsf{BL} \in \mathcal{L}$, and any subset $A \subseteq [N]$ such that $|A| = T$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Decrypt}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, & ((\mathsf{MK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T) \\ \mathsf{BL}) = (M_i)_{i\in[B]} & \forall i \in [B] : \mathsf{CT}_i \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M_i, \mathsf{BL}) \\ & \forall j \in A : \mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK}, \mathsf{MK}_j, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}) \end{array}\right] = 1.$$

- Correctness of Outsourced Batch Decryption: For any N, any $T \leq N$, any messages $(M_i)_{i\in[B]} \in \mathcal{M}^B$, any batch label $\mathsf{BL} \in \mathcal{L}$, and any subset $A \subseteq [N]$ such that $|A| = T$,

$$\Pr\left[\begin{array}{r|l} \mathsf{Recover}(\mathsf{RT},(\mathsf{TT}_i)_{i\in U}) & ((\mathsf{MK}_j)_{j\in[N]},\mathsf{EK},\mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda,1^B,1^N,1^T) \\ = (M_i)_{i\in[B]} & \forall i \in [B] : \mathsf{CT}_i \leftarrow \mathsf{Encrypt}(\mathsf{EK},M_i,\mathsf{BL}) \\ & \forall j \in A : \mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK},\mathsf{MK}_j,(\mathsf{CT}_i)_{i\in[B]},\mathsf{BL}) \\ & (\mathsf{RT},\mathsf{TK},(\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK},(\mathsf{DK}_j)_{j\in A},(\mathsf{CT}_i)_{i\in[B]},\mathsf{BL}) \\ & (\mathsf{TT}_i)_{i\in U} \leftarrow \mathsf{Transform}(\mathsf{PK},\mathsf{TK},(\mathsf{WT}_i)_{i\in U},\mathsf{BL}) \end{array}\right] = 1.$$

Definition B.3 (Selective IND-CCA Security with Static Corruptions of VOBTE)**.** The selective CCA security with static corruptions (SE-SC-CCA) of VOBTE is defined via the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}CCA}}_{VOBTE,\mathcal{A}}(\lambda,B)$ between a challenger and a PPT adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: $\mathcal{A}$ commits to the challenge batch label BL^*, the size of the decryption committee N, the threshold $T \leq N$, and the indices of the corrupted nodes $C \subseteq [N]$ of the decryption committee where $|C| < T$.

2. **Setup**: The challenger runs $((\mathsf{MK}_j)_{j\in[N]},\mathsf{EK},\mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda,1^B,1^N,1^T)$ and gives $((\mathsf{MK}_j)_{j\in C},\mathsf{EK},\mathsf{PK})$ to $\mathcal{A}$.

3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{OPreDecrypt}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRT}$ queries. The challenger handles these queries as follows:

   - $\mathsf{OPreDecrypt}((\mathsf{CT}_i)_{i\in[B]},\mathsf{BL})$: It runs $\mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK},\mathsf{MK}_j,(\mathsf{CT}_i)_{i\in[B]},\mathsf{BL})$ for all $j \in [N]\setminus C$, and responds with $(\mathsf{DK}_j)_{j\in[N]\setminus C}$.
   - $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]},\mathsf{BL})$: It proceeds as follows:
     (a) It computes $\mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK},\mathsf{MK}_j,(\mathsf{CT}_i)_{i\in[B]},\mathsf{BL})$ for each $j \in [N]$.
     (b) It computes $(\mathsf{RT},\mathsf{TK},(\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK},(\mathsf{DK}_j)_{j\in A},(\mathsf{CT}_i)_{i\in[B]},\mathsf{BL})$ where $A \subseteq [N]$ is an arbitrary index set of size T.
     (c) It picks a fresh handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\mathrm{del}} \leftarrow L_{\mathrm{del}} \cup \{(h,\mathsf{RT})\}$, and responds with $(\mathsf{hd},\mathsf{TK},(\mathsf{WT}_i)_{i\in U})$.
   - $\mathsf{ORevealRT}(\mathsf{hd})$: If there is no entry $(\mathsf{hd},\cdot) \in L_{\mathrm{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd},\mathsf{RT}) \leftarrow L_{\mathrm{del}}$ and responds with RT.

4. **Challenge**: $\mathcal{A}$ submits two messages $M_0^*, M_1^* \in \mathcal{M}$. The challenger flips a random coin $\mu \in \{0,1\}$, computes a challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{EK},M_\mu^*,\mathsf{BL}^*)$, and gives CT^* to $\mathcal{A}$.

5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. The challenger handles these queries exactly as in the pre-challenge queries.

6. **Output**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. The experiment returns 1 if $\mu' = \mu$ and the following oracle constraints are satisfied:

   (a) **(Challenge Privacy)**: For the challenge label BL^* and the challenge ciphertext CT^*, the adversary $\mathcal{A}$ is restricted as follows:

   - $\mathcal{A}$ never queries $\mathsf{OPreDecrypt}((\mathsf{CT}_i)_{i\in[B]},\mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.
   - $\mathcal{A}$ never queries $\mathsf{ORevealRT}(\mathsf{hd})$ for the handle hd returned by $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]},\mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.

(b) **(Label Distinctness)**: To ensure a consistent mapping between batch labels and ciphertext sets, the adversary $\mathcal{A}$ is restricted as follows:

- $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{OPreDecrypt}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of decryption key shares for each unique label.
- $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODelegate}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of delegation tasks for each unique label.
- For any batch label BL, $\mathcal{A}$ must associate it with only one specific set of ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$ across both $\mathsf{OPreDecrypt}$ and $\mathsf{ODelegate}$ queries. Any query involving a previously used label BL with a different set of ciphertexts $(\mathsf{CT}_j)_{j\in[B]}$ is strictly prohibited.

Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CCA}}_{VOBTE,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}CCA}}_{VOBTE,\mathcal{A}}(\lambda,B)=1] - \frac{1}{2}\right|$ where the probability is taken over all the randomness of the experiment. A VOBTE scheme is said to be SE-SC-CCA secure if the advantage is negligible for all PPT adversaries.

Definition B.4 (Selective Verifiability with Static Corruptions of VOBTE)**.** The selective verifiability with static corruptions (SE-SC-VER) of a VOBTE scheme is defined as the following experiment $\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}VER}}_{VOBTE,\mathcal{A}}(\lambda,B)$ between a challenger and an adversary $\mathcal{A}$ where λ and B are given as input:

1. **Init**: $\mathcal{A}$ commits to the challenge batch label BL^*, the size of the decryption committee N, the threshold $T \le N$, and the indices of the corrupted nodes $C \subseteq [N]$ of the decryption committee where $|C| < T$.

2. **Setup**: The challenger runs $((\mathsf{MK}_j)_{j\in[N]}, \mathsf{EK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^N, 1^T)$ and gives $((\mathsf{MK}_j)_{j\in C}, \mathsf{EK}, \mathsf{PK})$ to $\mathcal{A}$.

3. **Pre-challenge Queries**: $\mathcal{A}$ adaptively requests a polynomial number of $\mathsf{OPreDecrypt}$, $\mathsf{ODelegate}$, and $\mathsf{ORevealRT}$ queries. The challenger handles these queries as follows:

   - $\mathsf{OPreDecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It runs $\mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK}, \mathsf{MK}_j, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$ for all $j \in [N]\setminus C$, and responds with $(\mathsf{DK}_j)_{j\in[N]\setminus C}$.
   - $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$: It proceeds as follows:
     (a) It computes $\mathsf{DK}_j \leftarrow \mathsf{PreDecrypt}(\mathsf{PK}, \mathsf{MK}_j, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$ for each $j \in [N]$.
     (b) It computes $(\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}) \leftarrow \mathsf{Delegate}(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$ where $A \subseteq [N]$ is an arbitrary index set of size T.
     (c) It picks a fresh handle $\mathsf{hd} \in \{0,1\}^\lambda$, updates $L_{\mathrm{del}} \leftarrow L_{\mathrm{del}} \cup \{(h, \mathsf{RT})\}$, and responds with $(\mathsf{hd}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.
   - $\mathsf{ORevealRT}(\mathsf{hd})$: If there is no entry $(\mathsf{hd}, \cdot) \in L_{\mathrm{del}}$, it responds with $\perp$. Otherwise, it retrieves $(\mathsf{hd}, \mathsf{RT}) \leftarrow L_{\mathrm{del}}$ and responds with RT.

4. **Challenge**: The challenger selects a random message $M^* \in \mathcal{M}$, computes a challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{EK}, M^*, \mathsf{BL}^*)$, and gives CT^* to $\mathcal{A}$.

5. **Post-challenge Queries**: $\mathcal{A}$ continues to request oracle queries. The challenger handles these queries exactly as in the pre-challenge queries.

6. **Output**: $\mathcal{A}$ outputs target ciphertexts $(\widetilde{\mathsf{CT}}_i)_{i\in[B]}$ such that $\widetilde{\mathsf{CT}}_k = \mathsf{CT}^*$ for an index $k \in [B]$, a target handle hd^* returned by $\mathsf{ODelegate}((\widetilde{\mathsf{CT}}_i)_{i\in[B]}, \mathsf{BL}^*)$, and target transformed tasks $(\widetilde{\mathsf{TT}}_i)_{i\in U}$. Next, $\mathcal{C}$ retrieves $(\mathsf{hd}^*, \mathsf{RT}^*) \leftarrow L_{\mathrm{del}}$ and recovers $(\widetilde{M}_i)_{i\in[B]} \leftarrow \mathsf{Recover}(\mathsf{RT}^*, (\widetilde{\mathsf{TT}}_i)_{i\in U})$.

   The experiment returns 1 if $(\widetilde{M}_k \in \mathcal{M} \wedge \widetilde{M}_k \neq M^*)$ for the index k and the following oracle constraints are satisfied:

   (a) **(Challenge Privacy)**: For the challenge label BL^* and the challenge ciphertext CT^*, the adversary $\mathcal{A}$ is restricted as follows:
   - $\mathcal{A}$ never queries $\mathsf{OPreDecrypt}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.
   - $\mathcal{A}$ never queries $\mathsf{ORevealRT}(\mathsf{hd})$ for the handle hd returned by $\mathsf{ODelegate}((\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL}^*)$ such that $\mathsf{CT}_i = \mathsf{CT}^*$ for some $i \in [B]$.

   (b) **(Label Distinctness)**: To ensure a consistent mapping between batch labels and ciphertext sets, the adversary $\mathcal{A}$ is restricted as follows:
   - $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{OPreDecrypt}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of decryption key shares for each unique label.
   - $\mathcal{A}$ never submits the same batch label BL to multiple $\mathsf{ODelegate}$ queries. That is, $\mathcal{A}$ is allowed to observe at most one set of delegation tasks for each unique label.
   - For any batch label BL, $\mathcal{A}$ must associate it with only one specific set of ciphertexts $(\mathsf{CT}_i)_{i\in[B]}$ across both $\mathsf{OPreDecrypt}$ and $\mathsf{ODelegate}$ queries. Any query involving a previously used label BL with a different set of ciphertexts $(\mathsf{CT}_j)_{j\in[B]}$ is strictly prohibited.

   Otherwise, it returns 0.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}VER}}_{VOBTE,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{SE\text{-}SC\text{-}VER}}_{VOBTE,\mathcal{A}}(\lambda, B) = 1]$ where the probability is taken over all the randomness of the game. A VOBTE scheme is said to be SE-SC-VER secure if the advantage is negligible for all PPT adversaries.

B.2 Construction

The VOBTE scheme is constructed by integrating the decentralized key management of O-TBIBE into the VOBE framework. While the overarching design principles such as achieving CCA security via OTS and ensuring verifiability through PRF-based tagging remain consistent with the VOBE scheme, the VOBTE architecture introduces several critical modifications to support threshold operations. The primary architectural distinction in VOBTE is the addition of the PreDecrypt algorithm, which allows individual committee nodes to generate partial decryption key shares. Furthermore, the Decrypt and Delegate algorithms are refined to handle a set of decryption key shares rather than a single, unified decryption key. Specifically, these algorithms now incorporate an internal threshold reconstruction process that derives the decryption key once a sufficient number of shares are provided. This allows the VOBTE scheme to maintain the functional advantages of outsourced batch encryption while eliminating the reliance on a single trusted entity for key issuance.

The detailed specifications of the VOBTE scheme are provided as follows:

$\mathsf{VOBTE.Setup}(1^\lambda, 1^B, 1^N, 1^T)$: It generates $((\mathsf{MK}_{j,OTB})_{j\in[N]}, (\mathsf{PK}_{j,OTB})_{j\in[N]}, \mathsf{EK}_{OTB}, \mathsf{PK}_{OTB}) \leftarrow \mathsf{O\text{-}TBIBE.Setup}(1^\lambda, 1^B, 1^N, 1^T)$ and selects a hash function: $\mathsf{H}_s : \{0,1\}^* \to \mathcal{I}$ from $\mathcal{H}$. It outputs a master key $\mathsf{MK} = \mathsf{MK}_{OTB}$, a public encryption key $\mathsf{EK} = (\mathsf{EK}_{OTB}, \mathsf{H}_s)$, and a public key $\mathsf{PK} = \big(\mathsf{PK}_{OTB}, (\mathsf{PK}_{j,OTB})_{j\in[N]}, \mathsf{H}_s\big)$.

VOBTE.Encrypt$(\mathsf{EK}, M, \mathsf{BL})$**:** The procedure is identical to VOBE.Encrypt except for the substitution of O-TBIBE for O-BIBE as the base scheme.

VOBTE.PreDecrypt$(\mathsf{PK}, \mathsf{MK}_j, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OTB}, (\mathsf{PK}_{j,OTB})_{j\in[N]}, \mathsf{H}_s)$, $\mathsf{MK}_j = \mathsf{MK}_{j,OTB}$, and $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OTB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for $i \in [B]$.

1. It initializes $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, proceeds as follows:
   (a) It checks that $1 \stackrel{?}{=} \mathsf{OTS.Verify}(\mathsf{VK}_{i,OTS}, \mathsf{BL} \| \mathsf{CH}_{i,OTB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$. If the check fails, it sets $M_i = \perp$ and continues to the next index.
   (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$ and $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
2. It derives $\mathsf{DIG} \leftarrow \mathsf{O\text{-}TBIBE.Digest}(\mathsf{PK}_{OTB}, S)$ and computes a key share $\mathsf{DK}_{j,OTB} \leftarrow \mathsf{O\text{-}TBIBE.ComputeKeyShare}(\mathsf{PK}_{OTB}, \mathsf{MK}_{j,OTB}, \mathsf{DIG}, \mathsf{BL})$.
3. Finally, it outputs the decryption key share $\mathsf{DK}_j = \mathsf{DK}_{j,OTB}$.

VOBTE.Decrypt$(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OTB}, (\mathsf{PK}_{j,OTB})_{j\in[N]}, \mathsf{H}_s)$, $\mathsf{DK}_j = \mathsf{DK}_{j,OTB}$ for $j \in A$ where $|A| = T$, and $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OTB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for $i \in [B]$.

1. It initializes $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
   (a) It checks that $1 \stackrel{?}{=} \mathsf{OTS.Verify}(\mathsf{VK}_{i,OTS}, \mathsf{BL} \| \mathsf{CH}_{i,OTB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$. If the check fails, it sets $M_i = \perp$ and continues to the next index.
   (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
2. It derives $\mathsf{DIG} \leftarrow \mathsf{O\text{-}TBIBE.Digest}(\mathsf{PK}_{OTB}, S)$ and combines the key shares $\mathsf{DK}_{OTB} \leftarrow \mathsf{O\text{-}TBIBE.CombineKeyShare}((\mathsf{DK}_{j,OTB})_{j\in A}, \mathsf{DIG}, \mathsf{BL})$. If the verification $1 \stackrel{?}{=} \mathsf{O\text{-}TBIBE.VerifyCombKey}(\mathsf{PK}_{OTB}, \mathsf{DK}_{OTB}, \mathsf{DIG}, \mathsf{BL})$ holds, it proceeds; otherwise, it outputs $\perp$.
3. For each $i \in U$, it recovers $\mathsf{CK}_{i,OTB} \leftarrow \mathsf{O\text{-}TBIBE.Decaps}(\mathsf{PK}_{OTB}, \mathsf{DK}_{OTB}, S, \mathsf{CH}_{i,OTB}, \mathsf{ID}_i, \mathsf{BL})$, from which it derives $(K_i \| \eta_i) = \mathsf{PRF}(\mathsf{CK}_{i,OTB}, \mathsf{ID}_i \| \mathsf{BL})$ and decrypts the message $M_i \leftarrow \mathsf{SKE.Decrypt}(K_i, \mathsf{CT}_{i,SKE})$.
4. Finally, it outputs the corresponding messages $(M_i)_{i\in[B]}$.

VOBTE.Delegate$(\mathsf{PK}, (\mathsf{DK}_j)_{j\in A}, (\mathsf{CT}_i)_{i\in[B]}, \mathsf{BL})$**:** Let $\mathsf{PK} = (\mathsf{PK}_{OTB}, (\mathsf{PK}_{j,OTB})_{j\in[N]}, \mathsf{H}_s)$, $\mathsf{DK}_j = \mathsf{DK}_{j,OTB}$ for $j \in A$, and $\mathsf{CT}_i = (\mathsf{VK}_{i,OTS}, \mathsf{CH}_{i,OTB}, \mathsf{CT}_{i,SKE}, \eta_i, \sigma_i)$ for $i \in [B]$.

1. It initializes $U = \emptyset$ and $S = \emptyset$, and for each $i \in [B]$, it proceeds as follows:
   (a) It checks that $1 \stackrel{?}{=} \mathsf{OTS.Verify}(\mathsf{VK}_{i,OTS}, \mathsf{BL} \| \mathsf{CH}_{i,OTB} \| \mathsf{CT}_{i,SKE} \| \eta_i, \sigma_i)$. If the check fails, it continues to the next index.
   (b) It computes $\mathsf{ID}_i = \mathsf{H}_s(\mathsf{VK}_{i,OTS})$ and updates $U \leftarrow U \cup \{i\}$, $S \leftarrow S \cup \{\mathsf{ID}_i\}$.
2. It derives $\mathsf{DIG} \leftarrow \mathsf{O\text{-}TBIBE.Digest}(\mathsf{PK}_{OTB}, S)$ and combines the key shares $\mathsf{DK}_{OTB} \leftarrow \mathsf{O\text{-}TBIBE.CombineKeyShare}((\mathsf{DK}_{j,OTB})_{j\in A}, \mathsf{DIG}, \mathsf{BL})$. If the verification $1 \stackrel{?}{=} \mathsf{O\text{-}TBIBE.VerifyCombKey}(\mathsf{PK}_{OTB}, \mathsf{DK}_{OTB}, \mathsf{DIG}, \mathsf{BL})$ holds, it proceeds; otherwise, it outputs $\perp$.
3. It generates $(\mathsf{RK}_{OTB}, \mathsf{TK}_{OTB}, (\mathsf{WT}_{i,OTB})_{i\in U}) \leftarrow \mathsf{O\text{-}TBIBE.Delegate}(\mathsf{PK}_{OTB}, \mathsf{DK}_{OTB}, (\mathsf{CH}_{i,OTB})_{i\in U}, \mathsf{BL})$.
4. It sets a recovery token $\mathsf{RT} = \big(\mathsf{RK}_{OTB}, (\mathsf{CT}_{i,SKE}, \eta_i)_{i\in U}, \mathsf{BL}\big)$, a transformation key $\mathsf{TK} = \mathsf{TK}_{OTB}$, and a work task $\mathsf{WT}_i = (\mathsf{WT}_{i,OTB}, \mathsf{VK}_{i,OTS})$ for each $i \in U$.

5. Finally, it outputs $(\mathsf{RT}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U})$.

VOBTE.Transform$(\mathsf{PK}, \mathsf{TK}, (\mathsf{WT}_i)_{i\in U}, \mathsf{BL})$**:** The procedure is identical to VOBE.Transform except for the substitution of O-TBIBE for O-BIBE as the base scheme.

VOBTE.Recover$(\mathsf{RT}, (\mathsf{TT}_i)_{i\in U})$**:** The procedure is identical to VOBE.Recover except for the substitution of O-TBIBE for O-BIBE as the base scheme.

B.3 Security Analysis

Theorem B.1. *The proposed VOBTE scheme satisfies SE-SC-CCA security for any polynomial batch size B, assuming that the OTS scheme is SUF secure, and the hash function is collision-resistant, the underlying O-TBIBE scheme is SE-SC-CPA secure, the PRF is pseudo-random, and the SKE scheme is IND-OT secure.*

Proof. To prove the SE-SC-CCA security of the VOBTE scheme, we employ a sequence of hybrid games similar to the SE-IND-CCA proof of the VOBE scheme in Theorem 4.2. We define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \cdots, \mathbf{G}_6$. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$.

Game $\mathbf{G}_0$. This game corresponds to the original SE-SC-CCA security experiment. The simulator handles all queries using the master key MK. By definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} = \mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CCA}}_{VOBTE,\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$, except that the simulator aborts if the adversary $\mathcal{A}$ forges a valid OTS signature σ_i for $\mathsf{VK}_{i,OTS} = \mathsf{VK}^*_{OTS}$ (Event BAD_1). It follows that $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}\right| \leq \mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{B}}(\lambda)$.

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$, except the simulator aborts if a hash collision occurs such that $\mathsf{VK}_{i,OTS} \neq \mathsf{VK}^*_{OTS}$ but $\mathsf{H}_s(\mathsf{VK}_{i,OTS}) = \mathsf{H}_s(\mathsf{VK}^*_{OTS})$ (Event BAD_2). Thus, we have $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}\right| \leq \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}}(\lambda)$.

Game $\mathbf{G}_3$. In this game, the simulator $\mathcal{B}$ acts as an adversary against the underlying O-TBIBE scheme. At the Init step of the game, $\mathcal{B}$ receives the corruption set $C \subseteq [N]$ and the challenge label BL^* from $\mathcal{A}$ and forwards them to the O-TBIBE challenger to obtain the static corruption shares $(\mathsf{MK}_j)_{j\in C}$. For $\mathsf{OPreDecrypt}(\cdot,\cdot)$ queries, $\mathcal{B}$ invokes the O-TBIBE key-share oracle to obtain $(\mathsf{DK}_j)_{j\in[N]\setminus C}$. All $\mathsf{ODelegate}$ and $\mathsf{ORevealRT}$ queries are simulated by combining these shares. Due to the correctness of O-TBIBE and the elimination of BAD_1 and BAD_2, this simulation is perfect. Thus, $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}$.

Game $\mathbf{G}_4$. This game is identical to $\mathbf{G}_3$, except that the challenge capsule key CK^*_{OTB} is replaced by a random value $\widetilde{CK} \in \mathbb{G}_T$. This transition relies on the SE-SC-CPA security of the O-TBIBE scheme which is easily derived from the SE-SC-CPA security. It follows that $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4}\right| \leq \mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CPA}}_{O\text{-}TBIBE,\mathcal{B}}(\lambda)$.

Game $\mathbf{G}_5$. In this game, the simulator replaces the PRF outputs $(K^*\|\eta^*)$ with a random string $(\widetilde{K}\|\widetilde{\eta})$. This relies on the pseudo-randomness of the PRF. We have $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_5}\right| \leq \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda)$.

Game $\mathbf{G}_6$. In this game, the simulator replaces the challenge SKE ciphertext CT^*_{SKE} with an encryption of a zero-message. This transition is bounded by the IND-OT security of the SKE. We have $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_5} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_6}\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}OT}}_{SKE,\mathcal{B}}(\lambda)$. Since the challenge ciphertext in this final game $\mathbf{G}_6$ contains no information about μ, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_6} = 0$.

Combining the bounds from $\mathbf{G}_0$ to $\mathbf{G}_6$, the total advantage $\mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CCA}}_{VOBTE,\mathcal{A}}(\lambda)$ is negligible, which completes the proof. □

Theorem B.2. *The proposed VOBTE scheme satisfies SE-SC-VER security for any polynomial batch size B, assuming that the OTS scheme is SUF secure, and the hash function is collision-resistant, underlying O-TBIBE scheme is SE-SC-CPA secure, and the PRF is pseudo-random.*

Proof. The proof for SE-SC-VER security follows a sequence of hybrid games similar to the SE-SC-CCA proof in Theorem B.1. We focus on the transition from the ciphertext's algebraic structure to its statistical independence.

Games $\mathbf{G}_0 \sim \mathbf{G}_4$. These games are defined identically to those in the proof of Theorem B.1. Specifically, $\mathbf{G}_1$ and $\mathbf{G}_2$ handle OTS unforgeability and hash collisions. In $\mathbf{G}_3$, the simulator transitions to the O-TBIBE security model, handling static corruption and oracles via the O-TBIBE challenger. In $\mathbf{G}_4$, the challenge capsule key CK^*_{OTB} is replaced by a random value $\widetilde{CK} \in \mathbb{G}_T$. The advantage difference up to this point is:

$$\left|\mathsf{Adv}^{\mathbf{G}_0}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{SUF}}_{OTS,\mathcal{B}} + \mathsf{Adv}^{\mathsf{CR}}_{HF,\mathcal{B}} + \mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}CPA}}_{O\text{-}TBIBE,\mathcal{B}}.$$

Game $\mathbf{G}_5$. In this game, the simulator replaces the PRF output $(K^*\|\eta^*)$ with a purely random string $(\widetilde{K}\|\widetilde{\eta}) \xleftarrow{\$} \{0,1\}^*$. Under the pseudo-randomness of the PRF, this change is computationally indistinguishable. It follows that $\left|\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{PR}}_{PRF,\mathcal{B}}(\lambda)$.

In this final game $\mathbf{G}_5$, the tag η^* is a purely random string, which is statistically independent of the adversary $\mathcal{A}$'s view. For $\mathcal{A}$ to win the SE-SC-VER game, it must output a forged transformed task $(\widetilde{\mathsf{TT}}_i)$ such that Recover yields $\widetilde{M}_k \neq M^*$ while the verification check involving η^* passes. Since η^* is chosen uniformly at random and is independent of the adversary's capabilities, the probability of $\mathcal{A}$ successfully forging a tag that matches the simulator's random choice is exactly $\mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}} = \frac{1}{2^\lambda}$.

Combining the bounds from $\mathbf{G}_0$ to $\mathbf{G}_5$, the total advantage $\mathsf{Adv}^{\mathsf{SE\text{-}SC\text{-}VER}}_{VOBTE,\mathcal{A}}(\lambda)$ is negligible, which completes the proof. □