Constant-Cost Batched Partial Decryption in Threshold Encryption

Sora Suegami[1] [0009−0008−7107−3676], Shinsaku Ashizawa[2] and Kyohei Shibano[3] [0000−0002−2150−8001]

[1] Privacy & Scaling Explorations, Ethereum Foundation `sorasuegami@pse.dev`
[2] Invers Inc, Tokyo, Japan
[3] Department of Technology Management for Innovation, School of Engineering, The University of Tokyo, Tokyo, Japan

Abstract. Threshold public key encryption schemes distribute secret keys among multiple parties, known as the committee, to reduce reliance on a single trusted entity. However, existing schemes face inefficiencies as the committee should perform computation and communication for decryption of each individual ciphertext. As the number of ciphertexts being decrypted per unit of time increases, this can limit the number of committee parties and their decentralization due to increased hardware requirements, heightening the risk of adversarial collusion. To address this, we introduce tag-based batched threshold encryption (TBTE), which ensures constant computational and communication costs per committee member, independent of the number of ciphertexts being decrypted in batch under distinct decryption policies. The TBTE scheme is constructed over bilinear groups in the random oracle model and secure in the algebraic group model, assuming the hardness of the (q_1, q_2)-discrete logarithm problem and the EAV-security of the symmetric-key encryption scheme. Evaluation of our implementation demonstrates constant data size, specifically 48 bytes received and 56 bytes sent, and constant execution time for each committee party during decryption, even for various batch sizes up to 2^{20}.

Keywords: Threshold Encryption · Witness Encryption · One-time programs.

1 Introduction

Many modern cryptographic schemes ensure security not by hiding the algorithms but by using secret keys that are held only by trusted parties [49]. However, this approach introduces a dependency on the party holding the secret key, creating a potential single point of failure in applications using cryptographic schemes. Threshold cryptosystems [8, 27, 28, 53] address this issue by splitting the secret key among multiple parties, thus eliminating the single point of failure. For example, threshold cryptosystems have been used in networks where a trusted party is absent, e.g., peer-to-peer network [52], Ad Hod network [63], and blockchain [20, 50, 61]. Given this context, the efficiency for threshold cryptosystems has been improved for over 40 years [53].

In threshold public key encryption (threshold PKE) [27, 28], a master secret key, corresponding to a public key, is distributed among n parties. At least $t + 1 \leq n$ parties must collaborate to decrypt a ciphertext encrypted under the public key. Specifically, each party, upon receiving the ciphertext, locally generates a partial decryption. With $t+1$ or more partial decryptions, anyone can recover the message from the ciphertext. Importantly, the security of threshold PKE guarantees that any coalition of t or fewer parties cannot decrypt the ciphertext.

In many practical applications utilizing threshold PKE, a common group of parties holding secret key shares, often referred to as a committee, is employed. Examples of such committees include those that escrow secrets under some decryption policies [17, 25, 62], audit a data integrity in cloud service providers [19], and decrypt encryptions of transactions on blockchains to prevent exploitation called "Miner Extractable Value [24]" (MEV) [20, 50].

However, traditional threshold PKE schemes are not scaled efficiently because the computational and communication costs for the committee increase proportionally with the number of ciphertexts being decrypted. Thus, to increase the number of ciphertexts decrypted per unit of time, a committee party would need to use hardware with greater computational and communication capabilities. Consequently, the cost of participating in the committee increases, which could ultimately limit the number of parties and their decentralization. This increases the likelihood of an adversary colluding with more than t parties. These challenges motivate us to focus on minimizing the computational and communication costs for the committee in relation to the number of ciphertexts being decrypted.

Signature-based witness encryption (SWE) proposed in [29], transforming a digital signature scheme into an encryption scheme, maintains those costs constant, but only for ciphertexts under the same decryption policy. Specifically, a ciphertext in the SWE scheme is associated with a public key and a tag to be signed. It can be decrypted if signatures for that tag are provided using secret key shares such that the number of these shares exceeds a certain threshold, which can be treated as partial decryptions in the context of threshold PKE[4]. However, assuming that each tag represents a different decryption policy and that the committee only signs tags corresponding to fulfilled decryption policies, the costs for the committee are still proportional to the number of decryption policies. This poses a practical problem because many applications of threshold PKE, such as those mentioned above, require managing multiple decryption policies.

[4] The original construction in [29] encrypts a message under distinct public keys, each independently generated by the corresponding committee party. However, we can observe that it can also be done using a single public key, with secret key shares distributed among the committee parties.

Table 1. Comparison of our scheme against existing threshold PKE schemes. The second, third, and fourth columns represent the reception, transmission, and computational costs per committee party for generating partial decryptions for ciphertexts under B distinct decryption policies, respectively. The fifth column indicates the computational complexity of the encryption algorithm and the ciphertext size with respect to the committee size n. The sixth column lists the processes required for setting up a public key without relying on a trusted authority in each scheme.

Scheme	Reception	Transmission	Computational	Encryption	Setup
BBH06 [10]	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(1)$	DKG
DHM+23 [29]	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(n)$	Non-Interactive
GKP+24 [37]	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(B)$	$\mathcal{O}(1)$	Non-Interactive
CGP+24 [20]	$\mathcal{O}(B)$	$\mathcal{O}(1)$	$\mathcal{O}(B)$	$\mathcal{O}(1)$	MPC + Per Epoch Setup
Our Scheme	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	DKG + Powers-of-tau Setup

On the other hand, batched-threshold encryption introduced in [20] enables a committee to generate partial decryptions for a batch of B ciphertexts, without decrypting any ciphertexts outside the batch. Notably, the size of the partial decryptions is independent of B, implying that the transmission cost par committee party remains constant. However, their reception cost and the computational cost increase linearly with B.

Thus, our question is as follows:

Can we establish $\mathcal{O}(1)$ computational and communication costs for each committee party to generate partial decryptions for a batch of ciphertexts under distinct decryption policies?

Our Contribution: Our answer is yes. We propose a new batched-threshold encryption called tag-based batched threshold encryption (TBTE). This maintains constant computational and communication costs for each committee party, regardless of the number of batched ciphertexts under distinct decryption policies and committee parties (Table 1). We construct it over a bilinear group using a symmetric-key encryption (SKE) scheme in the random oracle model, and prove its security in the algebraic group model, assuming the hardness of (q_1, q_2)-discrete logarithm problem and the EAV-security of the SKE scheme. Evaluation of our implementation demonstrates that, for committee sizes of both 9 and 99, and with batch sizes ranging from 2^{10} to 2^{20}, the data size received and transmitted by each committee party is 48 bytes and 56 bytes, respectively, and the execution time required to generate a partial decryption remains constant.

The remainder of this paper is organized as follows. The rest of Section 1 introduces related work and provides an outline of our scheme. Section 2 describes the notations and preliminaries. Section 3 defines the TBTE scheme, and Section 4 presents our construction of the TBTE scheme. Section 5 presents the implementation and experimental evaluation to verify its efficiency. Section 6 discusses extending the TBTE scheme to one-time programs as its application and outlines future challenges. Finally, Section 7 concludes the paper.

1.1 Literature Review

Threshold encryption: we review how the efficiency of the threshold PKE scheme has been improved. Desmedt and Frankel [28] originally introduce the notion of threshold PKE and present its construction from the ELGamel PKE scheme [30], in which the computational and communication costs for encryption remain constant regardless of the committee size. Shoup and Gennaro [54] propose the first practical threshold PKE scheme secure against chosen ciphertext attack in the random oracle model. Boneh et al. [10] removes the random oracle without sacrificing efficiency by constructing the threshold PKE scheme from identity-based encryption.

In the above schemes, a public key and secret key shares are generated through a trusted setup, depending on a trusted authority, or distributed key generation (DKG) [48], requiring communications among the committee parties. Those setup processes can be removed by generating threshold-based secret shares of a message to be encrypted and encrypting each share under each party's public key [51]; however, unlike the schemes introduced in the previous paragraph, the running time of the encryption process and the data size of the ciphertext in this approach increase proportionally with the committee size. Reyzin et al. [51] propose a threshold PKE scheme that depends neither the trusted authority nor interactions for the setup process but maintains those costs for the encryptor constant, assuming indistinguishability obfuscation (iO). Garg1 et al. [37] show that such a threshold PKE scheme can be constructed without iO by building a new SWE scheme.

When considering decryption, in most conventional threshold PKE schemes, partial decryptions are generated for each individual ciphertext, causing the computational and communication costs for the committee to increase with the number of ciphertexts. Choudhuri et al. [20] address this issue, proposing the batched-threshold encryption scheme. The committee for this scheme outputs partial decryptions for a batch of ciphertexts so that their size is constant regardless of the number of batched ciphertexts; specifically, this scheme employs a Kate, Zaverucha, and Goldberg (KZG) commitment [44] to allow the constant-size partial decryptions to help decryption for multiple but only chosen ciphertexts. However, as mentioned above, their reception communication cost and the computational cost still increase linearly with the number of batched ciphertexts. In this manner, existing threshold PKE schemes do not achieve constant communication and computational costs for the committee regardless of the number of batched ciphertexts.

Practical witness encryption over bilinear groups: although witness encryption (WE) [36] has not necessarily been studied in the context of threshold encryption, we review some WE schemes because the construction of our scheme is inspired by practical WE schemes defined over bilinear groups. In WE, an encryptor encrypts a message under an instance of an NP language, and a decryptor can decrypt the ciphertext if the decryptor holds a witness for the instance. Unfortunately, all of the WE schemes supporting any NP language rely on inefficient cryptographic schemes or not well-studied assumptions [2, 4, 22, 36, 39, 40, 57–59] However, some WE scheme have been proposed that are practical and based on well-studied cryptographic tools such as bilinear groups, at the cost of significantly restricting the supported NP languages [1, 6, 16, 26, 29, 32, 45].

Among those practical ones, we review two schemes that are closely related to our construction. The first one is SWE [29] described above. In the context of WE, a public key and a tag associated with a ciphertext is defined as the instance, and signatures that meet the required threshold is defined as the witness. The construction of the SWE scheme in [29] is based on the Boneh, Lynn, and Shacham (BLS) signature scheme [11]. The second scheme is WE for KZG commitments introduced in [32], defining a KZG commitment and an expected polynomial evaluation result as the instance and its opening proof as the witness. Notably, the expected decryptor only needs to send a single commitment to allow the encryptor to encrypt messages under distinct evaluation results and the same commitment[5]. Our construction requires a variant of the BLS signatures for decryption, treated as partial decryptions, and also employs the KZG commitment to maintain constant costs for the committee regardless of the number of batched ciphertexts.

A concurrent and independent work in [21] proposes a new batched-threshold encryption scheme with a mathematical approach similar to ours, namely constructing a ciphertext that can be decrypted using a variant of the BLS signature on the KZG commitment. However, their work focuses on building a batched-threshold encryption scheme in the context of MEV, achieving constant transmission costs for each committee party after a one-time setup. In contrast, our work is not tied to any specific application and aims to maintain constant computational and communication costs for each committee party. Unlike their scheme, our scheme does not directly protect against attacks on pending transaction privacy, such as decrypting an honest user's ciphertext outside the batch by including another ciphertext with the same tag within the batch. Nevertheless, this protection can be easily implemented in our scheme by restricting the available tags associated with ciphertexts, similar to their approach[6].

1.2 Technical Overview

Interface of the TBTE scheme: the system design of our TBTE scheme is similar to the batched-threshold encryption scheme presented in [20]. Specifically, the system consists of encryptors, decryptors, and committee parties holding shares of a master secret key. The committee initiates a new epoch, identified by an ID eid, at regular intervals. For each epoch, multiple encryptors broadcast ciphertexts independently, and committee members produce partial decryptions

[5] The WE scheme in [32] assumes that a polynomial in the commitment has the exponentially-large randomness; therefore only the decryptor who knows that randomness can decrypt ciphertexts under that commitment [32].

[6] Specifically, an encryptor is required to prove a knowledge of a scalar $s \in \mathbb{Z}_p$ such that the tag T is derived from the element $[s]_1 \in \mathbb{G}_p$ via a random oracle [21].

for some ciphertexts with their own secret key shares. With at least $t+1$ partial decryptions, anyone can eventually decrypt the ciphertexts.

However, unlike the previous batched-threshold encryption scheme [20], our TBTE scheme ensures constant computational and communication costs for each committee party through three key mechanisms. Here, a tag T refers to data corresponding to a decryption policy, which is an integer in the finite field $\mathbb{Z}_p$ in our construction described later, and let a tag vector of size B be $\mathbf{T} := (T_1, \ldots, T_B)$. An index idx is a positive integer: $\mathsf{idx} \in \{1, \ldots, B\}$.

1. Associating a tag T and an index idx with each ciphertext.
2. Generating a constant-size digest for $\mathbf{T} := (T_1, \ldots, T_B)$, denoted by digest.
3. Generating partial decryptions for digest rather than for ciphertexts.

These partial decryptions for digest are considered under the condition that enable the decryption of any ciphertext associated with a tag T in $\mathbf{T}$ at the corresponding index idx, specifically when $T = T_{\mathsf{idx}}$. However, ciphertext not associating with digest cannot be decrypted. Under this assumption, each committee party only needs to receive the constant-size digest and output its partial decryption, requiring a constant amount of computation and communication, regardless of the batch size B. We note that in the TBTE scheme, batch size B represents the maximum number of distinct decryption policies defined by different tags that are associated with ciphertexts being decrypted in batch, unlike in [20] where it indicates the number of batched ciphertexts; for example, even if 100 ciphertexts are linked to each tag-index pair, the committee can still handle $100B$ ciphertexts in batch with constant costs.

Construction of the TBTE scheme: we construct the TBTE scheme over a bilinear group $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ with the above properties in two steps. In the following, an element g_t^a is represented as $[a]_t$ in the additive group for every $t \in \{1, 2, T\}$, and the bilinear map $e([a]_1, [b]_2)$ is denoted by $[a]_1 \circ [b]_2$:

Step 1: Represent constraints on a ciphertext and data helping decryption as a linear equation on elements in $\mathbb{G}_1$ over bg_λ, namely $\Sigma_{i,j}[x_i]_1 \circ [y_j]_1 = [z]_T$. The elements in $\mathbb{G}_1$, denoted by $[\mathbf{x}]_1$, are provided by a decryptor, whereas the other elements, denoted by $[\mathbf{y}]_2$ and $[z]_T$, are specified by an encryptor and available to both players.

Step 2: Transform the above linear equation into an encryption scheme such that the decryptor must provide $[\mathbf{x}]_1$ satisfying that equation to decrypt a ciphertext associated with $[\mathbf{y}]_2$ and $[z]_T$. Specifically, the encryptor samples a uniformly random scalar $s \in \mathbb{Z}_p$, and reveals $s \cdot [\mathbf{y}]_2 := (s \cdot [y_1]_2, \ldots, s \cdot [y_{|\mathbf{y}|}]_2)$ and an encryption of a message m under an one-time secret key $s \cdot [z]_T$.

Notably, many previous studies [6, 15, 16, 20, 26, 29, 32, 37, 45] have adopted a transformation similar to Step 2. Therefore, the technical novelty of our scheme lies in Step 1, where we express the constraints required to verify the partial decryptions for the digest as a linear equation.

In Step 1, the following two conditions should be verified as the linear equation:

Condition 1: At least $t+1$ parties issue partial decryptions for digest.
Condition 2: A tag vector $\mathbf{T}$ corresponding to digest contains a tag T at the index idx

We express them as a linear equation by combining BLS signature [11] and KZG commitment [44] schemes.

A BLS signature can be verified as a linear equation over bg_λ on a signer's master private key, denoted by msk [11]. Let RO be a random oracle mapping a tag T into a random element in $\mathbb{G}_1$. The signature for T using msk is defined as an element $[\sigma]_1 := \mathsf{msk} \cdot \mathsf{RO}(T)$ in $\mathbb{G}_1$ and verified as Equation 1.

$$[\sigma]_1 \circ [1]_2 = \mathsf{RO}(T) \circ [\mathsf{msk}]_2 \tag{1}$$

In a threshold-keys setting, msk is distributed among the committee parties as Shamir secret shares [53]. A linear combination of any $t+1$ secret key shares can recovery msk with coefficients $(\ell_{i_1}, \ldots, \ell_{i_{t+1}})$, i.e., $\mathsf{msk} = \ell_{i_1}\mathsf{sk}_{i_1} + \cdots + \ell_{i_{t+1}}\mathsf{sk}_{i_{t+1}}$. Since Equation 1 is linear on msk, the same linear combination of signatures using $(\mathsf{sk}_{i_1}, \ldots, \mathsf{sk}_{i_{t+1}})$, i.e., $[\sigma]_1 := \ell_{i_1}\mathsf{sk}_{i_1} \cdot \mathsf{RO}(T) + \cdots + \ell_{i_{t+1}}\mathsf{sk}_{i_{t+1}} \cdot \mathsf{RO}(T)$, also satisfies Equation 1. Therefore, by treating these signatures as partial decryptions, we can verify Condition 1 as Equation 1; however, it is not compatible with Condition 2 as the partial decryptions (a.k.a. signatures using the secret key shares) are generated for each tag rather than a digest.

For Condition 2, we next introduce the KZG commitment scheme [44]. Recall that a KZG commitment c for a polynomial $f(X)$ can be used to prove its evaluation result $f(\alpha) = \beta$. This is verified as Equation 2, where τ is a trapdoor that no one knows, and the commitment $[c]_1$ and an opening proof $[h(\tau)]_1$ are defined as $[c]_1 := [f(\tau)]_1$ and $[h(\tau)]_1 := [\frac{f(\tau)-\beta}{\tau-\alpha}]_1$, respectively [44].

$$([c]_1 - [\beta]_1) \circ [1]_2 = [h(\tau)]_1 \circ ([\tau]_2 - [\alpha]_2) \tag{2}$$

Notably, if we define $f(X)$ such that $f(i) = T_i$ holds for every $i \in \{1, \ldots, B\}$, then $[c]_1$ is treated as a vector commitment for the tag vector $\mathbf{T} = (T_1, \ldots, T_B)$. Threfore, by treating the KZG commitment as the vector commitment and verifying $f(\mathsf{idx}) = T_{\mathsf{idx}}$, i.e., setting $\alpha = \mathsf{idx}$ and $\beta = T_{\mathsf{idx}}$, we can verify Condition 2 as Equation 2. In the following, we refer to the KZG commitment as a digest denoted by $[d]_1$, rather than a commitment denoted by $[c]_1$, because it does not guarantee the hiding property.

To design a linear equation as which both Conditions 1 and 2 can be verified, we first propose combining Equations 1 and 2 by multiplying msk by each term in Equation 2:

$$(\mathsf{msk} \cdot [d]_1) \circ [1]_2 - T_{\mathsf{idx}} \cdot [\mathsf{msk}]_T = [h(\tau)]_1 \circ ([\tau \circ \mathsf{msk}]_2 - \mathsf{idx} \cdot [\mathsf{msk}]_2) \tag{3}$$

Recall that in Step 2, while an encryptor specifies elements in $\mathbb{G}_2$ and $\mathbb{G}_T$, namely $\mathsf{idx} \cdot [\mathsf{msk}]_2$, $[\tau \circ \mathsf{msk}]_2$, and $T_{\mathsf{idx}} \cdot [\mathsf{msk}]_T$, a decryptor provides elements in $\mathbb{G}_1$, namely $\mathsf{msk} \cdot [d]_1$ and $[h(\tau)]_1$. We can observe that Equation 3 verifies Condition 1 because $\mathsf{msk} \cdot [d]_1$ requires at least $t+1$ parties holding the secret shares of msk

to output partial decryptions $(\mathsf{sk}_{i_1} \cdot [d]_1, \ldots, \mathsf{sk}_{i_{t+1}} \cdot [d]_1)$. This implies that each committee party receives the constant-size digest $[d]_1$ and outputs a variant of the BLS signature defined as $\mathsf{sk}_i \cdot [d]_1$. Furthermore, Condition 2 is also verified because the equation obtained by dividing both sides of Equation 3 by msk is identical to the verification equation for the KZG commitment scheme (Equation 2), which verifies that the polynomial $f(x)$ corresponding to $[d]_1$ satisfies $f(\mathsf{idx}) = T_{\mathsf{idx}}$. Thus, Equation 3 can handle both Conditions 1 and 2. Notably, it also satisfies our efficiency requirements, i.e., each committee party only needs to perform a fixed amount of computation and communication, as the data sizes of both $[d]_1$ and $\mathsf{sk}_i \cdot [d]_1$, as well as the computational cost to obtain $\mathsf{sk}_i \cdot [d]_1$, are constant.

However, the above construction is not secure when the same secret key shares are used for multiple distinct digests. Specifically, given $\mathsf{msk} \cdot [d_1]_1$ and $\mathsf{msk} \cdot [d_2]_1$ for two digests $[d_1]_1$ and $[d_2]_1$, corresponding to distinct tag vectors $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively, an adversary can compute a linear combination of them such as $\mathsf{msk} \cdot ([d]_1 - [d]_2)$, which is valid partial decryptions for digests corresponding to the same linear combination of $\mathbf{T}_1$ and $\mathbf{T}_2$, without helps from the non-corrupted committee parties.

We address this vulnerability by adding a random element in $\mathbb{G}_1$ unique to eid, denoted by $[r_{\mathsf{eid}}]_1$, to $\mathsf{msk} \cdot [d]_1$. In other words, we define the partial decryption as $[\sigma]_1 := \mathsf{msk} \cdot ([d]_1 + [r_{\mathsf{eid}}]_1)$. That random element is defined as the output of the random oracle on input eid, i.e., $[t_{\mathsf{eid}}] := \mathsf{RO}(\mathsf{eid})$. This change modifies Equation 3 as follows:

$$\begin{aligned} &\{\mathsf{msk} \cdot ([d]_1 + [r_{\mathsf{eid}}]_1)\} \circ [1]_2 - T_{\mathsf{idx}} \cdot [\mathsf{msk}]_T \\ =&[h(\tau)]_1 \circ ([\tau \circ \mathsf{msk}]_2 - \mathsf{idx} \cdot [\mathsf{msk}]_2) + ([r_{\mathsf{eid}}]_1 \circ [\mathsf{msk}]_2) \end{aligned} \tag{4}$$

The new element $[r_{\mathsf{eid}}]_1 \circ [\mathsf{msk}]_2$ is specified by the encryptor. Informally, unless more than t committee parties outputs multiple partial decryptions for the same eid but distinct digests, the above attack does not succeed because any non-trivial linear combination of two distinct random elements $[r_{\mathsf{eid}_1}]_1$ and $[r_{\mathsf{eid}_2}]_1$ should not be equal to $\mathsf{RO}(\mathsf{eid}^{'})$ for any $\mathsf{eid}^{'}$[7]. In this manner, we can define the linear equation as which both Conditions 1 and 2 are verified, along with satisfying the efficiency property for the committee. As we described, Step 2 directly transforms it into the encryption scheme, which is identical to our construction of the TBTE scheme in Section 4.

2 Preliminaries

2.1 Notations

For $n \in \mathbb{N}$, $[n]$ denotes a set of $\{1, \ldots, n\}$, and $[0]$ represents an empty set $\emptyset$. Similarly, $[i, j]$ for $i \leq j$ denotes a set of $\{i, i+1, \ldots, j\}$. A vector and a

[7] When the coefficients of $[r_{\mathsf{eid}_1}]_1$ and $[r_{\mathsf{eid}_2}]_1$ in the linear combination are zero, they are equal to the output of $\mathsf{RO}(\mathsf{eid}_1)$ and $\mathsf{RO}(\mathsf{eid}_2)$, respectively. However, they do not reveal any new information to the adversary, which is why such linear combinations are trivial.

matrix are denoted by bold letters, e.g., $\mathbf{a}$ and $\mathbf{A}$. For any n-length vector $\mathbf{a}$ and its index $i \in [n]$, a_i represents the i-th component of $\mathbf{a}$, and $|\mathbf{a}|$ is the length of $\mathbf{a}$. Let $\mathbb{Z}_p[X_1, \ldots, X_m]$ denote the ring of polynomials in m-variables $(X_1, \ldots, X_m)$ defined over $\mathbb{Z}_p$. We define the degree of the m-variable polynomial, represented by $F(X_1, \ldots, X_m) = \Sigma_{\mathbf{i} \in \mathbb{N}^m} f_{i_1,\ldots,i_m} \prod_{j \in [m]} X_j^{i_j} \in \mathbb{Z}_p[X_1, \ldots, X_m]$, as the maximum total degree, i.e., $\deg F := \max_{\mathbf{i} \in \mathbb{N}^m : f_{i_1,\ldots,i_m} \neq 0} \{\Sigma_{j \in [m]} i_j\}$. The i-th Lagrange basis for a degree $|D| - 1$ polynomial on a domain D is denoted by $\ell_{i,D}(X)$.

Let λ be a security parameter. A function $\mathsf{negl}(\lambda) : \mathbb{N} \to \mathbb{R}$ is called negligible, if there exists $n \in \mathbb{N}$ for all constants $c > 0$ such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ holds for all $\lambda > n$.

2.2 Bilinear Groups

For a security parameter 1^λ, a bilinear group $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is defined by a tuple of a prime p, cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p, their generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and a type-3 bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is efficient and non-degenerat. A generator of $\mathbb{G}_T$ is defined as $g_T = e(g_1, g_2)$. In the following, we fix a bilinear group bg with omitting a security parameter λ.

We adopt the same bracket notation for group elements as in [37]. That is, for $t \in \{1, 2, T\}$, $[a]_t$ represents g_t^a. All of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are represented as additive groups. For every $[a]_1$ and $[b]_2$, the bilinear map e satisfies $e([a]_1, [b]_2) = [ab]_T$, which is also denoted by $[a]_1 \circ [b]_2 = [ab]_T$

2.3 Hardness Assumptions over Bilinear Groups

We show some hardness assumptions defined over bilinear groups. They are employed to prove the security of our construction in Section 4.

We first present a (q_1, q_2)-discrete logarithm problem ((q_1, q_2)-DLP). In brief, it requires an adversary to output a discrete log z with access to $[1]_1, [z]_1, \ldots, [z^{q_1}]_1 \in \mathbb{G}_1$ and $[1]_2, [z]_2, \ldots, [z^{q_2}]_2 \in \mathbb{G}_2$. Definition 1 is taken from [5].

Definition 1 (Hardness of (q_1, q_2)-Discrete Logarithm Problem). *The (q_1, q_2)-discrete logarithm problem ((q_1, q_2)-DLP) is said to be ϵ-hard, if any PPT adversary $\mathcal{A}$ has an advantage less than or equal to ϵ in the following game (q_1, q_2)-$\boldsymbol{DLP}^{\mathcal{A}}_{\boldsymbol{bg}}$ between a challenger and $\mathcal{A}$.*

```
$(q_1, q_2)$-$\boldsymbol{DLP}^{\mathcal{A}}_{\boldsymbol{bg}}$
1 :  $z \leftarrow\!\$\ \mathbb{Z}_p$
2 :  $z^* \leftarrow \mathcal{A}([1]_1, [z]_1, \ldots, [z^{q_1}]_1, [1]_2, [z]_2, \ldots, [z^{q_2}]_2)$
3 :  return $(z = z^*)$
```

We define the advantage of $\mathcal{A}$ as $\boldsymbol{Adv}^{(q_1,q_2)\text{-}DLP}_{\boldsymbol{bg},\mathcal{A}} := \Pr\Big[(q_1, q_2)\text{-}\boldsymbol{DLP}^{\mathcal{A}}_{\boldsymbol{bg}} = 1\Big]$.

We next introduce a flexible Uber assumption. A basis of the Uber assumption is originally introduced in [9] as a generalization of cryptographic assumptions over bilinear groups. A security game for the Uber assumption is defined for three lists of polynomials $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T \in \mathbb{Z}_p[\mathbf{X}]^n$ and three challenge polynomials $Q_1(\mathbf{X}), Q_2(\mathbf{X}), Q_T(\mathbf{X}) \in \mathbb{Z}_p[\mathbf{X}]$ [9, 14]. Boyen [14] extends the Uber assumption to an adaptive adversary setting, called flexible Uber assumption, in which the adversary can adaptively choose the polynomials $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T$, $Q_1(X)$, $Q_2(X)$, and $Q_T(X)$ during the security game. Bauer et al. [5] prove that the hardness of the flexible Uber assumption is reduced to that of the (q_1, q_2)-DLP in an algebraic group model (AGM), in which an adversary is required to output not only a target group element but also coefficients that represents its output by group operations from given group elements [34].

We take a game definition of the flexible Uber assumption from Figure 3 in [5], with necessary modifications.

Definition 2 ((m, d_1, d_2, d_T, d^*)-Decisional Flexible Uber Assumption). *The $(m, d_1, d_2, d_T, d_1^*, d_2^*, d_T^*)$-flexible Uber assumption is said to be ϵ-hard, if any PPT adversary $\mathcal{A}$ has an advantage less than or equal to ϵ in the following game $(m, d_1, d_2, d_T, d_1^*, d_2^*, d_T^*)$-$\boldsymbol{FUber}_{\boldsymbol{bg},b}^{\mathcal{A}}$ between a challenger and $\mathcal{A}$, where $\boldsymbol{X}$ denotes $(X_1, \ldots, X_m)$.*

```
$(m, d_1, d_2, d_T, d_1^*, d_2^*, d_T^*)$-$\boldsymbol{FUber}_{\boldsymbol{bg}}^{\mathcal{A}}$
1: $\boldsymbol{x} := (x_1, \ldots, x_m) \leftarrow\$ \mathbb{Z}_p^m$
2: $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T \leftarrow \emptyset$
3: $(Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X}), [y^*]_1, [z^*]_2, [w^*]_T) \leftarrow \mathcal{A}^{\mathsf{O}_{poly}(\cdot,\cdot)}()$
4: return $([y^*]_1, [z^*]_2, [w^*]_T) = ([Q_1(\boldsymbol{x})]_1, [Q_2(\boldsymbol{x})]_2, [Q_T(\boldsymbol{x})]_T)$
     $\wedge\, (\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T, Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X}))$ are non-trivial
```

```
Oracle $\mathsf{O}_{poly}(t, P(\boldsymbol{X}))$
1: $\mathcal{L}_t \leftarrow \mathcal{L}_t \cup \{P\}$
2: return $[P(x_1, \ldots, x_m)]_t$
```

We define the advantage of $\mathcal{A}$ as follows:

$$\boldsymbol{Adv}_{\boldsymbol{bg},\mathcal{A}}^{(m,d_1,d_2,d_T,d_1^*,d_2^*,d_T^*)\text{-}\boldsymbol{FUber}} := \Pr\Big[(m, d_1, d_2, d_T, d_1^*, d_2^*, d_T^*)\text{-}\boldsymbol{FUber}_{\boldsymbol{bg}}^{\mathcal{A}} = 1\Big]$$

In the above game, for every $t \in \{1, 2, T\}$, each polynomial P in $\mathcal{L}_t$ must satisfy $\deg P \leq d_t$. Similarly, $\deg Q_1 \leq d_1^*, \deg Q_2 \leq d_2^*, \deg Q_T \leq d_T^*$ must hold, respectively. Besides, these polynomials must be non-trivial. Specifically, no tuple of coefficients $(\{\alpha_i\}_{i\in|\mathcal{L}_1|}, \{\beta_i\}_{i\in|\mathcal{L}_2|}, \{\gamma_{i,j}\}_{i\in|\mathcal{L}_1|,j\in|\mathcal{L}_2|}, \{\delta_i\}_{i\in|\mathcal{L}_T|})$ satisfies all of the following equations:

$$Q_1(\boldsymbol{X}) = \Sigma_{i\in|\mathcal{L}_1|}\alpha_i\mathcal{L}_{1,i}(\boldsymbol{X})$$
$$Q_2(\boldsymbol{X}) = \Sigma_{i\in|\mathcal{L}_2|}\beta_i\mathcal{L}_{2,i}(\boldsymbol{X})$$
$$Q_T(\boldsymbol{X}) = \Sigma_{i\in|\mathcal{L}_1|,j\in|\mathcal{L}_2|}\gamma_{i,j}\mathcal{L}_{1,i}(\boldsymbol{X})\mathcal{L}_{2,j}(\boldsymbol{X}) + \Sigma_{i\in|\mathcal{L}_T|}\delta_i\mathcal{L}_{T,i}(\boldsymbol{X})$$

2.4 Algebraic Group Model

The security of our schemes is proven in the AGM, introduced in [34]. This model requires an adversary to output not only a group element but also coefficients that represents its output by group operations from given group elements.

Our formal definition of the AGM is based on Definition 2.4 in [5], which extends the original definition in [34] to type-3 bilinear groups.

Definition 3 (Algebraic Adversarie). *Let* $\mathsf{bg} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ *be bilinear groups. A PPT adversary* $\mathcal{A}$ *with input* $[\boldsymbol{x}]_1 \in \mathbb{G}_1^n$, $[\boldsymbol{y}]_2 \in \mathbb{G}_2^m$, $[\boldsymbol{z}]_T \in \mathbb{G}_T^l$ *is said to be algebraic, if in addition to its outputs*

$$[\boldsymbol{p}]_1 = ([p_1]_1, \ldots, [p_{n'}]_1) \in \mathbb{G}_1^{n'}$$
$$[\boldsymbol{q}]_2 = ([q_1]_2, \ldots, [q_{m'}]_2) \in \mathbb{G}_2^{m'}$$
$$[\boldsymbol{r}]_T = ([r_1]_T, \ldots, [r_{l'}]_T) \in \mathbb{G}_T^{l'}$$

$\mathcal{A}$ *also provides coefficients* $(a_{i,j})_{i\in[n'],j\in[n]}$, $(b_{i,j})_{i\in[m'],j\in[m]}$, $(c_{i,j,k})_{i\in[l'],j\in[n],k\in[m]}$, $(d_{i,j})_{i\in[l'],j\in[l]}$ *such that the following equations hold:*

$$\forall i \in [n'],\ [p_i]_1 = \Sigma_{j\in[n]} a_{i,j}[x_j]_1$$
$$\forall i \in [m'],\ [q_i]_2 = \Sigma_{j\in[m]} b_{i,j}[y_j]_2$$
$$\forall i \in [l'],\ [r_i]_T = \Sigma_{j\in[n],k\in[m]} c_{i,j,k}([x_j]_1 \circ [y_k]_2) + \Sigma_{j\in[l]} d_{i,j}[z_j]_T$$

2.5 Symmetric-Key Encryption

We use the definition of a symmetric-key encryption (SKE) scheme for a key space $\mathcal{K}$ and a message space $\mathcal{M}$ from Definition 10 in [32]. Notably, it assumes that the key denoted by k is randomly sampled from $\mathcal{K}$.

The SKE scheme defined for a key space $\mathcal{K}$ and a message space $\mathcal{M}$ consists of the following algorithms $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Enc}(k, m) \to \mathsf{ct}$: it takes as input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and outputs a ciphertext ct.
- $\mathsf{Dec}(k, \mathsf{ct}) \to m$: it takes as input a key $k \in \mathcal{K}$ and a ciphertext ct, and outputs a message $m \in \mathcal{M}$.

The correctness of the SKE scheme is defined as follows:

Definition 4 (Correctness of SKE). *The SKE scheme* $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ *is said to be correct, if for every* $\lambda \in \mathbb{N}$, $k \in \mathcal{K}$, *and* $m \in \mathcal{M}$, *the following holds:*

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

We require the SKE scheme to satisfy EAV-security, which is weaker than IND-CPA security because the same key is used only once for the encryption of the challenge message in the EAV-security.

Definition 5 (EAV-Security of the SKE scheme). *Let $\mathcal{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme defined for a key space $\mathcal{K}$ and a message space $\mathcal{M}$. The SKE scheme is said to be ϵ-EAV-secure if for every $\lambda \in \mathbb{N}$, any PPT adversary $\mathcal{A}$ has an advantage less than or equal to ϵ in the following game between a challenger and $\mathcal{A}$.*

```
$\boldsymbol{SKE}^{\mathcal{A}}_{\mathcal{M},\mathcal{K}}$
1 :  $k \leftarrow\!\!\$\ \mathcal{K}$
2 :  $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$
3 :  $b \leftarrow\!\!\$\ \{0,1\}$
4 :  $\mathsf{ct} \leftarrow \mathsf{Enc}(k, m_b)$
5 :  $b' \leftarrow \mathcal{A}(\mathsf{ct})$
6 :  return $b = b'$
```

We define the advantage of $\mathcal{A}$ as $\boldsymbol{Adv}^{\mathsf{SKE}}_{\mathcal{M},\mathcal{K},\mathcal{A}} :=\mid \Pr\Big[\boldsymbol{SKE}^{\mathcal{A}}_{\mathcal{M},\mathcal{K}} = 1\Big] - \frac{1}{2} \mid$.

3 Definition

In the TBTE scheme, there are encryptors, decryptors, and committee parties that hold secret key shares for a public key pk. For each epoch specified by eid, multiple encryptors non-interactively output ciphertexts under pk. Each ciphertext is associated with a tag T and an index idx. A decryptor chooses B tags $\mathbf{T} := (T_1, \ldots, T_B)$ with distinct indexes and then computes their digest, denoted by digest. If at least $t+1$ committee parties outputs partial decryptions for digest, the decryptor can decrypt any ciphertexts associated with a tag T and an index idx such that the idx-th tag in $\mathbf{T}$ is T.

The syntax of the algorithms for the TBTE scheme is formally defined in Definition 6.

Definition 6 (TBTE). *A tag-based batched-threshold encryption (TBTE) scheme for a committee size $n \in \mathbb{N}$, a threshold $t \in [n]$, a tag space $\mathcal{T}$, a batch size $B \in \mathbb{N}$, and a message space $\mathcal{M}$ consists of a tuple of algorithms $\mathsf{TBTE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Digest}, \mathsf{BatchDec}, \mathsf{Combine})$ with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n))$*: it takes as input a security parameter 1^λ and outputs a public key and n secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{eid}, \mathsf{idx}, T, m) \rightarrow \mathsf{ct}$*: it takes as input a public key pk, an epoch id $\mathsf{eid} \in \mathbb{N}$, an index $\mathsf{idx} \in [B]$, a tag $T \in \mathcal{T}$, and a message $m \in \mathcal{M}$. It outputs a ciphertext ct.*
- $\mathsf{Digest}(\mathsf{pk}, (T_1, \ldots, T_B)) \rightarrow \mathsf{digest}$*: it takes as input a public key pk and a vector of B tags $(T_i)_{i \in [B]} \in \mathcal{T}^B$, and outputs a digest digest.*
- $\mathsf{BatchDec}(\mathsf{sk}, \mathsf{eid}, \mathsf{digest}) \rightarrow \mathsf{pd}$*: it takes as input a secret key sk, an epoch id $\mathsf{eid} \in \mathbb{N}$, and a digest digest. It outputs a partial decryption pd or a symbol $\perp$.*

- *$\mathsf{Combine}(\mathsf{pk}, \mathsf{eid}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_B), \{\mathsf{pd}_i\}_{i \in S}) \to (m_1, \ldots, m_B)$: it takes as input a public key pk, an epoch id $\mathsf{eid} \in \mathbb{N}$, B ciphertexts $(\mathsf{ct}_i)_{i \in [B]}$, and $t+1$ partial decryptions $\{\mathsf{pd}\}_{i \in S}$, where $S \subseteq [n]$ and $|S| = t+1$. It outputs B messages $(m_i)_{i \in [B]} \in \mathcal{M}^B$ or a symbol $\perp$.*

The correctness of the TBTE scheme is defined in Definition 7.

Definition 7 (Correctness of TBTE). *The TBTE scheme TBTE=(Setup, Enc, Digest, BatchDec, Combine) is said to be correct, if for every $\lambda \in \mathbb{N}$, $n, B \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $t \in [n]$, an epoch id eid, $(T_i)_{i \in [B]} \in \mathcal{T}^B$, and $(m_i)_{i \in [B]} \in \mathcal{M}^B$, the following holds:*

$$\Pr[\mathsf{Combine}(\mathsf{pk}, \mathsf{eid}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_B), \{\mathsf{pd}_i\}_{i \in S}) = (m_1, \ldots, m_B)] \geq 1 - \mathsf{negl}(\lambda)$$

, where $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow \mathsf{Setup}(1^\lambda)$, $ct_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{eid}, i, T_i, m_i)$ for every $i \in [B]$, $\mathsf{pd}_i \leftarrow \mathsf{BatchDec}(\mathsf{sk}_i, \mathsf{eid}, \mathsf{Digest}(\mathsf{pk}, (T_i)_{i \in [B]}))$ for every $i \in S$, and S is a set such that $S \subseteq [n]$ and $|S| = t+1$ hold.

The TBTE scheme should satisfies the following efficiency properties:

Definition 8 (Efficiency of TBTE). *Let TBTE=(Setup, Enc, Digest, BatchDec, Combine) be a TBTE scheme for a committee size $n \in \mathbb{N}$, a threshold $t \in [n]$, a tag space $\mathcal{T}$, a batch size $B \in \mathbb{N}$, and a message space $\mathcal{M}$. The TBTE scheme is said to be efficient if for every $\lambda \in \mathbb{N}$, the following efficiency properties hold:*

- *The running time of the algorithm Enc and the data size of ct are bounded by $\mathit{poly}(1^\lambda, |\mathsf{eid}|, \log B, |T|, |m|)$, i.e., independent of n and sub-linear in B.*
- *The data size of digest is bounded by $\mathit{poly}(1^\lambda)$, i.e., independent of n and B.*
- *The running time of the algorithm $\mathsf{BatchDec}$ and the data size of pd are, respectively, bounded by $\mathit{poly}(1^\lambda, |\mathsf{eid}|)$ and $\mathit{poly}(1^\lambda)$, i.e., independent of n and B.*

The security of the TBTE scheme is defined as follows. Notably, the adversary is allowed to adaptively query an oracle $\mathsf{O}_{\mathsf{dec}}$, returning a partial decryption for the given eid and $\mathbf{T}$ unless the partical decryption helps decryption of the challenge ciphertext ct^* in a trivial manner, even after receiving ct^*.

Definition 9 (Security of TBTE). *Let TBTE=(Setup, Enc, Digest, BatchDec, Combine) be a TBTE scheme for a committee size $n \in \mathbb{N}$, a threshold $t \in [n]$, a tag space $\mathcal{T}$, a batch size $B \in \mathbb{N}$, and a message space $\mathcal{M}$. The TBTE scheme is said to be ϵ-secure if for every $\lambda \in \mathbb{N}$, $n, B, \mathsf{eid}^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $t \in [n]$, any PPT adversary $\mathcal{A}$ has an advantage less than or equal to ϵ in the following game $\boldsymbol{TBTE}^{\mathcal{A}}_{\mathsf{eid}^*, b}$ between a challenger and $\mathcal{A}$.*

```
$\boldsymbol{TBTE}^{\mathcal{A}}_{eid^*}$
1:  $\mathcal{C} \leftarrow \mathcal{A}()$
2:  if $\mathcal{C} \not\subset [n] \vee |\mathcal{C}| > t$ then
3:      return 0
4:  fi
5:  $\mathcal{L} \leftarrow \emptyset, idx^*, T^*_{idx^*} \leftarrow \perp$
6:  $(pk, (sk_1, \ldots, sk_n)) \leftarrow Setup(1^\lambda)$
7:  $(idx^*, T^*_{idx^*}, m_{idx^*,0}, m_{idx^*,1}) \leftarrow \mathcal{A}^{\mathsf{O}_{dec}(\cdot,\cdot,\cdot)}(pk, \{sk_i\}_{i \in \mathcal{C}})$
8:  if $\exists \boldsymbol{T}', (eid^*, \boldsymbol{T}') \in \mathcal{L} \wedge T^*_{idx^*} = T'_{idx^*}$ then
9:      return 0
10: fi
11: $b \leftarrow\$ \{0,1\}$
12: $ct^* \leftarrow Enc(pk, eid^*, idx^*, T^*_{idx^*}, m_{idx^*,b})$
13: $b' \leftarrow \mathcal{A}^{\mathsf{O}_{dec}(\cdot,\cdot,\cdot)}(ct^*)$
14: return $(b = b')$
```

```
Oracle $\mathsf{O}_{dec}(eid, \boldsymbol{T}, i)$
1:  if $\exists \boldsymbol{T}', (eid, \boldsymbol{T}') \in \mathcal{L} \wedge \boldsymbol{T} \neq \boldsymbol{T}'$ then
2:      return $\perp$
3:  fi
4:  if $eid = eid^* \wedge idx^* \neq \perp \wedge T_{idx^*} = T^*_{idx^*}$ then
5:      return $\perp$
6:  fi
7:  if $(eid, \boldsymbol{T}) \notin \mathcal{L}$ then
8:      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(eid, \boldsymbol{T})\}$
9:  fi
10: $pd_i \leftarrow BatchDec(sk_i, eid, Digest(pk, \boldsymbol{T}))$
11: return $pd_i$
```

We define the advantage of $\mathcal{A}$ as follows.

$$\boldsymbol{Adv}^{\boldsymbol{TBTE}}_{eid^*,\mathcal{A}} := \left| \Pr\left[\boldsymbol{TBTE}^{\mathcal{A}}_{eid^*} = 1\right] - \frac{1}{2} \right|$$

Remark 1 (Security against Malicious Adversaries). The security definition in Definition 9 assumes a semi-honest adversary, i.e., the adversary follows the protocol but tries to learn the honest party's information as much as possible. For example, the adversary provides a tag vector $\mathbf{T}$ for $\mathsf{O}_{\mathsf{dec}}$, whereas the committee only receives its digest $\mathsf{Digest}(\mathsf{pk}, \mathsf{T})$ in practice. To fill this gap, zk-SNARKs [47] can be used, ensuring the security against malicious adversaries. Specifically, the algorithms Enc and Digest additionally output proofs to claim that ct and digest are generated in honest manners, respectively. The committee verifies the proof for digest before outputting the corresponding partial decryptions. However, to maintain the computational and communication costs for the committee constant as defined in Definition 8, we should use the zk-SNARKs schemes in which the proof size and the verification cost remain constant irrespective of the complexity of the NP relation to be proved such as Groth16 [41] and Plonk [35].

4 Construction

Using the SKE scheme for a key space $\mathcal{K}$ and a message space $\mathcal{M}$, denoted by SKE, we construct the TBTE scheme TBTE=(Setup, Enc, Digest, $\mathsf{BatchDec}$, $\mathsf{Combine}$) for a committee size $n \in \mathbb{N}$, a threshold $t \in [n]$, a tag space $\mathcal{T} \subseteq \mathbb{Z}_p$, a batch size $B \in \mathbb{N}$, and a message space $\mathcal{M}$. Let RO_1 and RO_2 be random oracles mapping an epoch id eid into an element in $\mathbb{G}_1$, and an element in $\mathbb{G}_T$ into a key in $\mathcal{K}$, respectively.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n))$:
  1. Sample $\tau, \mathsf{msk} \leftarrow\$ \mathbb{Z}_p$.
  2. Compute τ^i for every $i \in [B-1]$.
  3. Sample $(t+1)$-of-n Shamir secret shares [53] of msk denoted by $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in \mathbb{Z}_p$. They satisfy $\mathsf{msk} = \Sigma_{i \in S} \mathsf{sk}_i \ell_{i,S}(0)$ for any subset $S \subseteq [n]$ of a size $t+1$.
  4. Let $\mathsf{pk} \leftarrow ([1]_1 [\tau]_1, \ldots, [\tau^{B-1}]_1, [1]_2, [\tau]_2, [\mathsf{msk}]_2, [\tau \cdot \mathsf{msk}]_2)$.
  5. Output pk and $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{eid}, \mathsf{idx}, T, m) \to \mathsf{ct}$:
  1. Sample $s \leftarrow\$ \mathbb{Z}_p$.
  2. Extract $([1]_1, [1]_2, [\tau]_2, [\mathsf{msk}]_2, [\tau \cdot \mathsf{msk}]_2)$ from pk.
  3. Compute $[u]_2 \leftarrow (s \cdot \mathsf{idx})[\mathsf{msk}]_2 - s[\tau \cdot \mathsf{msk}]_2$.
  4. Compute $[r]_1 \leftarrow \mathsf{RO}_1(\mathsf{eid})$.
  5. Compute $[k]_T \leftarrow \{s \cdot ([T]_1 + [r]_1)\} \circ [\mathsf{msk}]_2$.
  6. Compute $\mathsf{key} \leftarrow \mathsf{RO}_2([k]_T)$
  7. Compute $\mathsf{ct}_m \leftarrow \mathsf{SKE.Enc}(\mathsf{key}, m)$.
  8. Let $\mathsf{ct} \leftarrow (\mathsf{eid}, \mathsf{idx}, T, [s]_2, [u]_2, \mathsf{ct}_m)$.
  9. Output ct.
- $\mathsf{Digest}(\mathsf{pk}, (T_1, \ldots, T_B)) \to \mathsf{digest}$:
  1. Parse pk as $([1]_1 [\tau]_1, \ldots, [\tau^{B-1}]_1, [1]_2, [\tau]_2, [\mathsf{msk}]_2, [\tau \cdot \mathsf{msk}]_2)$.
  2. Compute a polynomial $f(X) \leftarrow \Sigma_{i \in [B]} T_i \ell_{i,[B]}(X)$.
  3. Compute $[f(\tau)]_1 \leftarrow \Sigma_{i \in [B]} f_i [\tau^{i-1}]_1$.
  4. Output $\mathsf{digest} := [f(\tau)]_1$.
- $\mathsf{BatchDec}(\mathsf{sk}, \mathsf{eid}, \mathsf{digest}) \to \mathsf{pd}$:
  1. Compute $[r]_1 \leftarrow \mathsf{RO}_1(\mathsf{eid})$.
  2. Parse digest as $[f(\tau)]_1$.
  3. Compute $[\sigma]_1 \leftarrow \mathsf{sk} \cdot ([f(\tau)]_1 + [r]_1)$.
  4. Output $\mathsf{pd} := [\sigma]_1$.
- $\mathsf{Combine}(\mathsf{pk}, \mathsf{eid}, (\mathsf{ct}_1, \mathsf{ct}_B), \{\mathsf{pd}_i\}_{i \in S}) \to (m_1, \ldots, m_B)$:
  1. Parse pk as $([1]_1 [\tau]_1, \ldots, [\tau^{B-1}]_1, [1]_2, [\tau]_2, [\mathsf{msk}]_2, [\tau \cdot \mathsf{msk}]_2)$.
  2. For every $i \in [B]$, parse ct_i as $(\mathsf{eid}, i, T_i, [s_i]_2, [u_i]_2, \mathsf{ct}_{m_i})$.
  3. For every $i \in S$, parse pd_i as $[\sigma_i]_1$.
  4. Let $[\sigma]_1 \leftarrow \Sigma_{i \in S} \ell_{i,S}(0)[\sigma_i]_1$.
  5. Compute a polynomial $f(X) \leftarrow \Sigma_{i \in [B]} T_i \ell_{i,[B]}(X)$.
  6. For every $i \in [B]$, compute a polynomial $h_i(X) = \frac{f(X) - T_i}{X - i}$ and an element $[h_i(\tau)]_1 \leftarrow \Sigma_{j \in [B]} h_{i,j} [\tau^{i-1}]_1$.
  7. For every $i \in [B]$, compute $[k'_i]_T \leftarrow [\sigma]_1 \circ [s_i]_2 + [h_i(\tau)]_1 \circ [u_i]_2$ and $m_i \leftarrow \mathsf{SKE.Dec}(\mathsf{RO}_2([k']_T), \mathsf{ct}_{m_i})$.
  8. Output $(m_1, \ldots, m_B)$.

Remark 2 (Distributed setup). While the algorithm Setup in the above construction is assumed to be executed by a trusted authority, i.e., trusted setup, it can be easily modified to a protocol executed by the committee parties in a distributed manner. Specifically, the protocol can employ DKG schemes [33, 38, 42, 48] and powers-of-tau setup schemes [13, 23, 43, 46], respectively, to generate the public key pk and the corresponding secret key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$, and the elements for powers of taus $([1]_1[\tau]_1, \ldots, [\tau^{B-1}]_1, [1]_2, [\tau]_2)$.

The correctness of the above construction is proven in Theorem 1.

Theorem 1. *The construction of the TBTE scheme in Section 4 is correct (Definition 7).*

Proof. *We can express the values of elements $[k_i]_T, [\sigma]_1 \circ [s_i]_2, [q_i(\tau)]_1 \circ [u_i]_2$ as follows:*

$$\begin{aligned}
[k_i]_T &= \{s_i \cdot ([T_i]_1 + [t]_1)\} \circ [\mathsf{msk}]_2 \\
&= [s_i \cdot \mathsf{msk} \cdot (T_i + t)]_T \\
[\sigma]_1 \circ [s_i]_2 &= \{\Sigma_{i \in S} \ell_{i,S}(0)[\sigma_i]_1\} \circ [s_i]_2 \\
&= \{\mathsf{msk} \cdot ([f(\tau)]_1 + [t]_1)\} \circ [s_i]_2 \\
&= [s_i \cdot \mathsf{msk} \cdot (f(\tau) + t)]_T \\
[h_i(\tau)]_1 \circ [u_i]_2 &= [\frac{f(\tau) - T_i}{\tau - i}]_1 \circ \{(s_i \cdot i)[\mathsf{msk}]_2 - s_i[\tau \cdot \mathsf{msk}]_2\} \\
&= [\frac{f(\tau) - T_i}{\tau - i}]_1 \circ \{-(s_i \cdot \mathsf{msk})[\tau - i]_2\} \\
&= [-s_i \cdot \mathsf{msk} \cdot (f(\tau) - T_i)]_T
\end{aligned}$$

Therefore, for every $i \in [B]$, $[k_i^{'}]_T = [\sigma]_1 \circ [s_i]_2 + [h_i(\tau)]_1 \circ [u_i]_2 = [k_i]_T$ holds. Hence, from the correctness of the SKE scheme (Definition 4), the i-th output of the $\mathsf{BatchDec}$ algorithm, namely $\mathsf{SKE.Dec}(\mathsf{RO}_2([k']_T), \mathsf{ct}_{m_i})$, should be equal to the given message m_i.

The efficiency of our construction is proven in Theorem 8.

Theorem 2. *The construction of the TBTE scheme in Section 4 is efficienct (Definition 8).*

Proof. *We prove that our constuction satisfies all of the three conditions in Definition 8. First, the data size of ct only depends on 1^λ, $|\mathsf{eid}|$, $\log B$, $|T|$, and $|m|$ because $\mathsf{idx} \in [B]$ can be represented by $\log B$ bits. The running time of the algorithm Enc is also bounded in the same manner as only constrant number of elements are extracted from pk. Second, the data size of digest is bounded by 1^λ since it only consists of one element in $\mathbb{G}_1$. Third, the running time of the algorithm $\mathsf{BatchDec}$ is bounded by 1^λ and $|\mathsf{eid}|$ since the data size of digest is bounded by 1^λ, regardless of n and B. Similarly, the data size of pd is bounded by 1^λ as it only consists of one element in $\mathbb{G}_1$. Therefore, all conditions hold.*

We finally show that our construction of the TBTE scheme is secure as defined in Definition 9.

Lemma 1. *Let q_1 be the maximum number of queries to RO_1 except for ones for eid^*. In the AGM model over a bilinear group $\mathsf{bg} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, for every algebraic adversary $\mathcal{A}$, if the $(0,1)$−DLP (Definition 1) is ϵ_{DLP}-hard, the $(q_1+3, B, 2, 1, B+2, 0, 0)$-DF-Uber assumption (Definition 2) is $\epsilon_{\mathsf{FUber}}$-hard, and there exists a ϵ_{SKE}-EAV-secure SKE scheme for a key space $\mathcal{K}$ and a message space $\mathcal{M}$, the construction of the TBTE scheme for a committee size $n \in \mathbb{N}$, a threshold $t \in [n]$, a tag space $\mathcal{T} \subseteq \mathbb{Z}_p$, a batch size $B \in \mathbb{N}$, and a message space $\mathcal{M}$ in Section 4 is $(\epsilon_{\mathsf{SKE}} - (1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{FUber}} - \epsilon_{\mathsf{DLP}}))$-secure (Definition 9).*

Proof. *We define two variances of the game $\boldsymbol{TBTE}^{\mathcal{A}}_{\mathsf{eid}^*}$ and prove that they are indistinguishable.*

***Game 0**: it is the same as the real game $\boldsymbol{TBTE}^{\mathcal{A}}_{\mathsf{eid}^*}$ in Definition 9 except for the below modifications.*

1. *The adversary $\mathcal{A}$ does not send challenge messages $m_{idx^*,0}$ and $m_{idx^*,1}$.*
2. *Instead of $\mathsf{ct}_{m_{idx^*,0}}$ in the ciphertext ct^*, the challenger sends key^*_0 defined as $\mathsf{key}^*_0 := \mathsf{RO}_2([k^*]_T)$.*

***Game 1**: it is the same as Game 0 except that key^*_0 in ct^* is replaced with a uniformally random key $\mathsf{key}^*_1 \leftarrow\!\!\$\ \mathcal{K}$.*

In a nutshell, we first show that assuming the hardnesses over bilinear groups, Games 0 and 1 are indistinguishable except for negligible advantage. We next demonstrate that if those games are indistinguishable and there exists the secure SKE scheme, the TBTE scheme is secure.

Suppose there exists an algebraic distinguisher $\mathcal{D}$ that can distinguish between Games 0 and 1 with an advantage $\epsilon_{\mathcal{D}}$. Specifically, $\mathcal{D}$ outputs 0 if it estimates that it is in Game 0, and 1 otherwise. In Game b_1, where $b_1 \leftarrow\!\!\$\ \{0,1\}$, the challenger against $\mathcal{D}$ programs random oracles RO_1 and RO_2 in a lazy manner. For example, RO_2 is programmed as the function $\mathsf{programRO2}$.

```
programRO2([k]_T)
1 : key ← RO_2([k]_T)
2 : if key ≠⊥ then
3 :   return key
4 : else
5 :   key ← K
6 :   RO_2([k]_T) ← key
7 :   return key
8 : fi
```

Let Hit be the event that $\mathcal{D}$ queries RO_2 on input $[k^*]$, where $[k^*] \leftarrow \{s^* \cdot ([T^*_{\mathsf{idx}^*}]_1 + [r_{\mathsf{eid}^*}]_1)\} \circ [\mathsf{msk}]_2$, and $[r_{\mathsf{eid}^*}]_1 \leftarrow \mathsf{RO}_1(\mathsf{eid}^*)$. The probability that $\mathcal{D}$

outputs b_1 in Game b_1 can be expressed as follows:

$$\begin{aligned}\frac{1}{2}+\epsilon_{\mathcal{D}} &= \Pr[\mathcal{D}=b_1]\\ &= \Pr[\mathcal{D}=b_1 \mid \mathsf{Hit}]\Pr[\mathsf{Hit}] + \Pr\left[\mathcal{D}=b_1 \mid \overline{\mathsf{Hit}}\right]\Pr\left[\overline{\mathsf{Hit}}\right] \end{aligned} \quad (5)$$

When Hit occurs, $\mathcal{D}$ can definitely output b_1, i.e., $\Pr[\mathcal{D}=b_1 \mid \mathsf{Hit}]=1$. Specifically, after receiving ct^*, $\mathcal{D}$ outputs 0 if the output of $\mathsf{RO}_2([k^*])$ is the same as the given $\mathsf{key}^*_{b_1}$ and 1 otherwise. This succeeds as key^*_0 should be set in RO_2 for $[k^*]_T$ when ct^* is generated, but this is not the case in Game 1.

When Hit does not occur, $\mathcal{D}$ can output b_1 only with negligible probability, i.e., $\Pr\left[\mathcal{D}=b_1 \mid \overline{\mathsf{Hit}}\right] \le \frac{1}{2}+\mathsf{negl}(\lambda)$. This is because both in Games 0 and 1, $\mathcal{D}$ cannot obtain any data associated with $\mathsf{key}^*_{b_1}$ in the view of $\mathcal{D}$ without accessing to $\mathsf{RO}_2([k^*])$.

Based on the above analysis, Inequality 5 is simplified as follows.

$$\begin{aligned} &\frac{1}{2}+\epsilon_{\mathcal{D}}\\ \le& 1-\Pr\left[\overline{\mathsf{Hit}}\right] + (\frac{1}{2}+\mathsf{negl}(\lambda))\Pr\left[\overline{\mathsf{Hit}}\right]\\ =&(-\frac{1}{2}+\mathsf{negl}(\lambda))\Pr\left[\overline{\mathsf{Hit}}\right]+1\\ \Rightarrow& \Pr\left[\overline{\mathsf{Hit}}\right] \le \frac{1-2\epsilon_{\mathcal{D}}}{1-\mathsf{negl}(\lambda)} \end{aligned} \quad (6)$$

We can observe that $\Pr\left[\overline{\mathsf{Hit}}\right]$ is negligible if $\epsilon_{\mathcal{D}}$ is non-negligible.

As $\mathcal{D}$ is algebraic, when Hit occurs, it also outputs coefficients $(a_{i,j})_{i\in[B+q_1+q_2+2]}$ and c such that the following holds, where $\boldsymbol{x} := (1,\tau,\ldots,\tau^{B-1}, r_1,\ldots,r_{q_1}, r_{\mathsf{eid}^*}, \sigma_1,\ldots,\sigma_{q_2},\sigma_{\mathsf{eid}^*})$, $\boldsymbol{y}=(1,\tau,\mathsf{msk},\tau\cdot\mathsf{msk},s^*,u^*)$, $\boldsymbol{z}=(1)$, and q_2 is the number of queries to $\mathsf{O}_{\mathsf{dec}}$ except for ones for eid^*.

$$\begin{aligned}&(T^*_{\mathsf{idx}^*}+r_{\mathsf{eid}^*})\cdot\mathsf{msk}\cdot s^*\\ =&\Sigma_{i\in[|\boldsymbol{x}|],j\in[4]}a_{i,j}x_iy_j + \Sigma_{i\in[|\boldsymbol{x}|]}a_{i,5}x_is^* + \Sigma_{i\in[|\boldsymbol{x}|]}a_{i,6}x_iu^* + c\end{aligned} \quad (7)$$

We define $\sigma^* = \Sigma_{i\in[|\boldsymbol{x}|]}a_{i,5}x_i$ and $h^*_{\mathsf{idx}^*} = \Sigma_{i\in[|\boldsymbol{x}|]}a_{i,6}x_i$. Let Good be the condition that they satisfy Equation 8:

$$\begin{aligned}&[\sigma^*]_1\circ[1]_2 + [h^*_{\mathsf{idx}^*}]_1\circ(\mathsf{idx}^*\cdot[\mathsf{msk}]_2 - [\tau\cdot\mathsf{msk}]_2)\\ =&([T^*_{\mathsf{idx}^*}]_1+[r_{\mathsf{eid}^*}]_1)\circ[\mathsf{msk}]_2\end{aligned} \quad (8)$$

Here, we claim that the probability $\Pr\left[\mathsf{Hit}\wedge\overline{\mathsf{Good}}\right]$ is bounded by ϵ_{DLP}.

Claim 1. *In Game b_1, $\Pr\left[\mathsf{Hit}\wedge\overline{\mathsf{Good}}\right]\le\epsilon_{\mathsf{DLP}}$ holds.*

Proof. *We construct a PPT adversary $\mathcal{B}$ participating in $(0,1)$-$\boldsymbol{DLP}^{\mathcal{A}}_{\boldsymbol{bg}}$. At the begging of that game, $\mathcal{B}$ receives $([1]_1,[1]_2,[z])$, where $z \leftarrow\$\ \mathbb{Z}_p$. $\mathcal{B}$ internally* *invokes $\mathcal{D}$ by simulating the view of $\mathcal{D}$ in the same manner as the challenger in Game b_1. Notably, $\mathcal{B}$ knows trapdoors (τ, msk) generated during the setup and the randomnesses used for the output of* RO_1*. However, when returning* ct^**, $\mathcal{E}$ defines* $[s^*]_2 = [z]_2$*,* $[u^*]_2 = (\mathsf{idx}^* \cdot \mathsf{msk} - \tau \cdot \mathsf{msk})[z]_2$*, and* $[k^*]_T \leftarrow [(T^*_{\mathsf{idx}^*} + r_{\mathsf{eid}^*}) \cdot \mathsf{msk}]_1 \circ [z]_2$*, respectively.*

Since Hit *is assumed to occur, $\mathcal{D}$ should query* RO_2 *on input* $[k^*]_T$*. Recall that $\mathcal{D}$ also outputs coefficients* $(a_{i,j})_{i \in [B+q_1+q_2+2]}$ *and c satisfying Equation 7, which can be rearranged as follows:*

$$\begin{aligned}&\{(T^*_{\mathsf{idx}^*} + r_{\mathsf{eid}^*}) \cdot \mathsf{msk} - \Sigma_{i \in [|\boldsymbol{x}|]} a_{i,5} x_i - \Sigma_{i \in [|\boldsymbol{x}|]} a_{i,6} x_i (\mathsf{idx}^* \cdot \mathsf{msk} - \tau \cdot \mathsf{msk})\} z \\ =&\Sigma_{i \in [|\boldsymbol{x}|], j \in [4]} a_{i,j} x_i y_j + c \end{aligned} \tag{9}$$

Since there is no outputs $([\sigma^*]_1, [h^*_{\mathsf{idx}^*}]_1)$ *satisfying Equation 8 when* Good *is not true, the coefficient of z in Equation 9 is not zero. Therefore, $\mathcal{B}$ can compute z by dividing that right-hand side by that coefficient, which wins in the game* $(0,1)$-$\boldsymbol{DLP}^{\mathcal{A}}_{\boldsymbol{bg}}$*. Hence, this concludes the proof as follows:*

$$\Pr\left[\mathsf{Hit} \wedge \overline{\mathsf{Good}}\right] \leq \epsilon_{\mathsf{DLP}} \tag{10}$$

We next construct an algebraic adversary $\mathcal{E}$ participating in the game $(q_1+3, B, 2, 1, B+2, 0, 0)$-$\boldsymbol{FUber}^{\mathcal{A}}_{\boldsymbol{bg}}$*. Let* q_1+3*-variables $\boldsymbol{X}$ be* $\boldsymbol{X} = (X_\tau, X_{\mathsf{msk}}, X_{r_1}, \ldots, X_{r_{q_1}}, X_{r^*})$*. In brief, $\mathcal{E}$ simulates the view of $\mathcal{D}$ and obtains the coefficients output by $\mathcal{D}$. $\mathcal{D}$ will finally submit a challenge polynomial* $Q_1(\boldsymbol{X})$*, depending on* $\{a_{i,6}\}_{i \in [B+q_1+q_2+2]}$*, and its evaluation* $[\sigma^*]_1$*, where* $Q_1(\boldsymbol{X})$ *is non-trivial for any* $\{a_{i,6}\}_{i \in [B+q_1+q_2+2]}$ *and* $Q_1(\tau, \mathsf{msk}, r_1, \ldots, r_{q_1}, r^*) = \sigma^*$*.*

$\mathcal{E}$ first receives a set of corrupted parties $\mathcal{C}$ from $\mathcal{D}$. If $|\mathcal{C}| < t$*, $\mathcal{E}$ samples random* $t - |\mathcal{C}|$ *indexes* $\{i^*_1, \ldots, i^*_{t-|\mathcal{C}|}\}$ *from* $[n]/\mathcal{C}$ *and sets* $C^* \leftarrow \mathcal{C} \cup \{i^*_1, \ldots, i^*_{t-|\mathcal{C}|}\}$*; otherwise it simply sets* $\mathcal{C}^* \leftarrow \mathcal{C}$*. $\mathcal{E}$ then initializes a list* $\mathcal{L}_{\mathsf{dec}}$ *to store decryption queries* $(\mathsf{eid}, \boldsymbol{T}, i) \in \mathbb{N} \times \mathcal{T}^B \times [n]/\mathcal{C}$ *and a mapping $\mathcal{M}$ from an epoch id* $\mathsf{eid} \in \mathbb{N}$ *to an index* $l \in [q_1]$*.*

$\mathcal{E}$ programs random oracles RO_1 *and* RO_2*, respectively, as the functions* $\mathsf{programRO1}$ *and* $\mathsf{programRO2}$*, where* $\mathsf{programRO1}$ *is defined as follows:*

```
programRO1(eid)
    $[r_{eid}]_1 \leftarrow \mathsf{RO}_1(eid)$
    if $[r_{eid}]_1 \neq \perp$ then
        return $[r_{eid}]_1$
    elseif $eid = eid^*$ then
        $[r_{eid^*}]_1 \leftarrow \mathsf{O}_{poly}(1, X_{r^*})$
        $\mathsf{RO}_1(eid^*) \leftarrow [r_{eid^*}]_1$
        return $[r_{eid^*}]_1$
    else
        $l \leftarrow |\mathcal{M}| + 1$
        $\mathcal{M}[eid] \leftarrow l$
        $[r_l]_1 \leftarrow \mathsf{O}_{poly}(1, X_{r_l})$
        $\mathsf{RO}_1(eid) \leftarrow [r_l]_1$
        return $[r_l]_1$
    fi
```

During the setup, $\mathcal{E}$ submits the following queries to O_{poly} and receives the corresponding elements, respectively.

- $(1, X_\tau^{i-1})$ *for every* $i \in [B]$, *for which the responses correspond to* $([1]_1, [\tau], \ldots, [\tau^{B-1}]_1)$.
- $(2, 1)$, $(2, X_\tau)$, $(2, X_{msk})$, *and* $(2, X_\tau X_{msk})$, *for which the responses correspond to* $([1]_2, [\tau]_2, [msk]_2, [\tau \cdot msk]_2)$.
- $(T, 1)$, *for which the response corresponds to* $[1]_T$.

$\mathcal{E}$ then samples secret key shares for $\mathcal{C}^*$, namely $(sk_i)_{i \in \mathcal{C}^*} \leftarrow\$ \mathbb{Z}_p^t$. Using the above elements, $\mathcal{E}$ provides $\mathcal{D}$ a public key $pk = ([1]_1[\tau]_1, \ldots, [\tau^{B-1}]_1, [1]_2, [\tau]_2, [msk]_2, [\tau \cdot msk]_2)$ and $(sk_i)_{i \in \mathcal{C}}$.

When $\mathcal{D}$ queries O_{dec} on input $(eid, \boldsymbol{T}, i)$, $\mathcal{E}$ returns the output of the function resDecQuery.

```
resDecQuery(eid, T, i)
1 :  if $\exists \boldsymbol{T}', (\mathsf{eid}, \boldsymbol{T}') \in \mathcal{L}_{\mathsf{dec}} \wedge \boldsymbol{T} \neq \boldsymbol{T}'$ then
2 :    return $\perp$
3 :  fi
4 :  if $\mathsf{eid} = \mathsf{eid}^* \wedge \mathsf{idx}^* \neq \perp \wedge T_{\mathsf{idx}^*} = T^*_{\mathsf{idx}^*}$ then
5 :    return $\perp$
6 :  fi
7 :  if $(\mathsf{eid}, \boldsymbol{T}) \notin \mathcal{L}_{\mathsf{dec}}$ then
8 :    $\mathcal{L}_{\mathsf{dec}} \leftarrow \mathcal{L}_{\mathsf{dec}} \cup \{(\mathsf{eid}, \boldsymbol{T})\}$
9 :  fi
10 : $[r_l]_1 \leftarrow \mathsf{programRO}(\mathsf{eid})$
11 : if $\mathsf{eid} = \mathsf{eid}^*$ then
12 :   $S(\boldsymbol{X}) := X_{\mathsf{msk}}(\Sigma_{j \in [B]} T_{l,j} \ell_{j,[B]}(X_\tau) + X_{r^*})$
13 : else
14 :   $l \leftarrow \mathcal{M}[\mathsf{eid}]$
15 :   $S(\boldsymbol{X}) := X_{\mathsf{msk}}(\Sigma_{j \in [B]} T_j \ell_{j,[B]}(X_\tau) + X_{r_l})$
16 : fi
17 : $[\sigma]_1 \leftarrow \mathsf{O}_{\mathsf{poly}}(1, S(\boldsymbol{X}))$
18 : $[f(\tau)]_1 \leftarrow \mathsf{Digest}(\mathsf{pk}, \boldsymbol{T})$
19 : for $j \in \mathcal{C}^*$ do
20 :   $[\sigma_j]_1 \leftarrow \mathsf{BatchDec}(\mathsf{sk}_j, \mathsf{eid}, [f(\tau)]_1)$
21 : endfor
22 : $[\sigma_i]_1 \leftarrow \ell_{0,\mathcal{C}^* \cup \{0\}}(i)[\sigma]_1 + \Sigma_{j \in \mathcal{C}^*} \ell_{j,\mathcal{C}^* \cup \{0\}}(i)[\sigma_j]_1$
23 : return $[\sigma_i]_1$
```

Since it holds that $\mathcal{C}^* \subset [n]$ and $|\mathcal{C}^*| = t$, $[\sigma_i]_1$ output by the above function satisfies $[\sigma_i]_1 = (\ell_{0,\mathcal{C}^* \cup \{0\}}(i)\mathsf{msk} + \Sigma_{j \in \mathcal{C}^*} \ell_{j,\mathcal{C}^* \cup \{0\}}(i)\mathsf{sk}_j)([f(\tau)]_1 + [r_l]_1) = \mathsf{sk}_i([f(\tau)]_1 + [r_l]_1)$, which is the same as the output of $\mathsf{BatchDec}(\mathsf{sk}_i, \mathsf{eid}, [f(\tau)]_1)$.

When $\mathcal{D}$ sends $(\mathsf{eid}^*, \mathsf{idx}^*, T^*_{\mathsf{idx}^*}) \in \mathbb{N} \times [B] \times \mathbb{Z}_p$ to $\mathcal{E}$, $\mathcal{E}$ samples $s^* \leftarrow\$ \mathbb{Z}_p$ and computes $[s^*]_2$, $[u^*]_2$, and $[k^*]_1$ in the same manner as the algorithm Enc. $\mathcal{E}$ then samples $b_1 \leftarrow\$ \{0,1\}$, $\mathsf{key}^*_0, \mathsf{key}^*_1 \leftarrow \mathcal{K}$, and sets $\mathsf{RO}_2([k^*]_T) \leftarrow \mathsf{key}^*_0$. $\mathcal{E}$ provides $\mathsf{key}^*_{b_1}$ for $\mathcal{D}$.

$\mathcal{D}$ might continue to query $\mathsf{O}_{\mathsf{dec}}$. $\mathcal{E}$ returns the output of resDecQuery in the same manner as before for each query.

After $\mathcal{D}$ terminates, $\mathcal{E}$ submits the output to the challenger in the game $(q_1 + 3, B, 2, 1, B+2, 0, 0)$-$\boldsymbol{FUber}^{\mathcal{A}}_{\mathsf{bg}}$. If Hit has not occurred or Good does not hold, $\mathcal{E}$ outputs $\perp$, which fails the game. In the following, we assume Hit has occurred and Good holds.

The polynomials that has been queried to $\mathsf{O}_{\mathsf{poly}}$ are summarized as follows, where $S_l(\boldsymbol{X})$ and $S_{\mathsf{eid}^*}(\boldsymbol{X})$ are, respectively, defined in the recDecQuery function *for* eid *corresponding to* l *and* eid*.

$$\begin{aligned}
&\mathcal{L}_1 = (1, X_\tau, \ldots, X_\tau^{B-1}, X_{r_1}, \ldots, X_{r_{q_1}}, X_{r^*}, S_1(\boldsymbol{X}), \ldots, S_{q_2}(\boldsymbol{X}), S_{\mathsf{eid}^*}(\boldsymbol{X}))\\
&\mathcal{L}_2 = (1, X_\tau, X_{\mathsf{msk}}, X_\tau X_{\mathsf{msk}})\\
&\mathcal{L}_T = (1)
\end{aligned}$$

$\mathcal{E}$ defines challenge polynomials $Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X})$ as follows, where $\beta_i := a_{i,6}$ for every $i \in [B + q_1 + q_2 + 2]$.

$$\begin{aligned}
&Q_1(\boldsymbol{X}) := (T^*_{\mathsf{idx}^*} + X_{r^*})X_{\mathsf{msk}} - (\Sigma_{i\in[B+q_1+q_2+2]}\beta_i \mathcal{L}_{1,i})(\mathsf{idx}^* X_{\mathsf{msk}} - X_\tau X_{\mathsf{msk}})\\
&Q_2(\boldsymbol{X}) := 1\\
&Q_T(\boldsymbol{X}) := 1
\end{aligned}$$

We verify that polynomials in $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T$ and $Q_1(\boldsymbol{X})$, $Q_2(\boldsymbol{X})$, $Q_T(\boldsymbol{X})$ are valid, that is 1) the total degree of each polynomial is less than or equal to the required maximum degree, and 2) the polynomials are non-trivial as defined in Definition 2. Every polynomials in $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T$, and challenge polynomials $Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X})$, respectively, satisfy $\deg \mathcal{L}_{1,i} \le d_1 = B$, $\deg \mathcal{L}_{2,i} \le d_2 = 2$, $\deg \mathcal{L}_{T,i} \le d_T = 1$, $\deg Q_1 = d_1^* = B + 2$, $\deg Q_2 = d_2^* = 0$, $\deg Q_T = d_T^* = 0$. Therefore, the first condition holds. Claim 2 shows that the second condition holds.

Claim 2. *For any $(\beta_i)_{i\in[B+q_1+q_2+2]}$, the polynomials $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T, Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X}))$ are non-trivial. Specifically, there is no coefficients $\{\alpha_i\}_{i\in|B+q_1+q_2+2|}$ such that the following holds:*

$$Q_1(\boldsymbol{X}) = \Sigma_{i\in[B+q_1+q_2+2]}\alpha_i \mathcal{L}_{1,i} \tag{11}$$

Proof. *The polynomails $Q_2(\boldsymbol{X})$ and $Q_T(\boldsymbol{X})$ are clearly trivial; thus we prove that $Q_1(\boldsymbol{X})$ is non-trivial. To do that, suppose that the polynomials $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_T$, $Q_1(\boldsymbol{X})$ are trivial, and we derive a contradiction.*

Comparing the coefficients of the constant term, X_{msk}, X^2_{msk}, we can derive the following equations respectively.

$$\begin{aligned}
&0 = \Sigma_{i\in[B+q_1+1]}\alpha_i \mathcal{L}_{1,i}\\
&(T^*_{\mathsf{idx}^*} + X_{r^*})X_{\mathsf{msk}} - (\Sigma_{i\in[B+q_1+1]}\beta_i \mathcal{L}_{1,i})(\mathsf{idx}^* - X_\tau)X_{\mathsf{msk}} = \Sigma_{i\in[q_2+1]}\alpha_i \mathcal{L}_{1,B+q_1+1+i}\\
&- (\Sigma_{i\in[q_2+1]}\beta_{B+q_1+1+i}\mathcal{L}_{1,B+q_1+1+i})(\mathsf{idx}^* - X_\tau)X_{\mathsf{msk}} = 0
\end{aligned}$$

In the following, the given $(\beta_{B+q_1+1+i})_{i\in[q_2+1]}$ are assumed to satisfy $\beta_{B+q_1+1+i} = 0$ for every $i \in [q_2+1]$; otherwise it inconsistent with the coefficient of X^2_{msk} in the right-term. Similarly, comparing the coefficients of $X_{r_1}X_\tau X_{\mathsf{msk}}, \ldots, X_{r_{q_1}}X_\tau X_{\mathsf{msk}}$, $X_{r^*}X_\tau X_{\mathsf{msk}}$, respectively, $\beta_{B+i} = 0$ and $\beta_{B+q_1+1} = 0$ must hold for every $i \in [q_1]$. Besides, from the coefficients of $X_{r_1}X_{\mathsf{msk}}, \ldots, X_{r_{q_2}}X_{\mathsf{msk}}$ and $X_{r^*}X_{\mathsf{msk}}$, respectively, $\alpha_{B+q_1+1+i} = 0$ and $\alpha_{B+q_1+q_2+2} = 1$ hold for every $i \in [q_2]$.

In summary, Equation 11 is simplified as follows:

$$T^*_{\mathsf{idx}^*} - (\Sigma_{j\in[B]}\beta_j X_\tau^{j-1})(\mathsf{idx}^* - X_\tau) = \Sigma_{j\in[B]} T_{\mathsf{eid}^*,j}\ell_{j,B}$$
$$\Leftrightarrow \Sigma_{j\in[B]} T_{\mathsf{eid}^*,j}\ell_{j,B} - T^*_{\mathsf{idx}^*} = (\Sigma_{j\in[B]}\beta_j X_\tau^{j-1})(X_\tau - \mathsf{idx}^*) \quad (12)$$

By the Remainder Theorem, Equation 12 implies that $f_{\mathsf{eid}^*}(\mathsf{idx}^*) = T^*_{\mathsf{idx}^*}$*, where* $f_{\mathsf{eid}^*}(\boldsymbol{X}) := \Sigma_{j\in[B]} T_{\mathsf{eid}^*,j}\ell_{j,[B]}(X_\tau)$*. However,* $\mathcal{D}$ *is not allowed to submit* $(\mathsf{eid}^*, \mathsf{idx}^*, T^*_{\mathsf{idx}^*})$ *such that there exists any pair* $(\mathsf{eid}^*, \boldsymbol{T}^{'})$ *in* $\mathcal{L}_{\mathsf{dec}}$ *satisfying* $T^*_{\mathsf{idx}^*} = T^{'}_{\mathsf{idx}^*}$ *(Definition 9). Therefore,* $T_{\mathsf{eid}^*,\mathsf{idx}^*} \neq T^*_{\mathsf{idx}^*}$ *holds, which is inconsistent with* $f_{\mathsf{eid}^*}(\mathsf{idx}^*) = T_{\mathsf{eid}^*,\mathsf{idx}^*} = T^*_{\mathsf{idx}^*}$*. Hence, Claim 2 is proven.*

Recall that when Good *holds,* $[\sigma^*]_1$ *and* $[h^*_{idx^*}]_1$ *satisfy Equation 8, implying* $[\sigma^*]_1 = [(T^*_{\mathsf{idx}^*}+r_{\mathsf{eid}^*})\cdot\mathsf{msk} - h^*_{idx^*}\cdot(\mathsf{idx}^*\cdot\mathsf{msk} - \tau\cdot\mathsf{msk})]_1 = [Q_1(\tau, \mathsf{msk}, r_1, \ldots, r_{q_1}, r^*)]_1$*. Therefore, by submitting* $(Q_1(\boldsymbol{X}), Q_2(\boldsymbol{X}), Q_T(\boldsymbol{X}), [\sigma^*]_1, [1]_2, [1]_T)$*,* $\mathcal{E}$ *wins the game* $(q_1+3, B, 2, 1, B+2, 0, 0)$*-$\boldsymbol{FUber}^{\mathcal{A}}_{\mathsf{bg}}$.*

Hence, Inequality 13 holds:

$$\Pr[\mathit{Hit} \wedge \mathit{Good}] \leq \epsilon_{\mathsf{FUber}} \quad (13)$$

Inequalities 6, 10, 13 leads the following relation:

$$\Pr\left[\overline{\mathit{Hit}}\right] = 1 - \Pr[\mathit{Hit}\wedge \mathit{Good}] - \Pr\left[\mathit{Hit}\wedge\overline{\mathit{Good}}\right]$$
$$\Rightarrow \frac{1-2\epsilon_{\mathcal{D}}}{1-\mathsf{negl}(\lambda)} \geq 1 - \epsilon_{\mathsf{FUber}} - \epsilon_{\mathsf{DLP}}$$
$$\Leftrightarrow \epsilon_{\mathcal{D}} \leq \frac{1}{2} - \frac{(1-\mathsf{negl}(\lambda))(1-\epsilon_{\mathsf{FUber}}-\epsilon_{\mathsf{DLP}})}{2} \quad (14)$$

Therefore, the advantage of $\mathcal{D}$ *is bounded as shown in Inequality 14.*

We finally bound the advantage of $\mathcal{A}$ *participating in the real game* $\boldsymbol{TBTE}^{\mathcal{A}}_{\mathsf{eid}^*}$ *by* $\epsilon_{\mathcal{D}}$ *and* ϵ_{SKE}*. To do that, we construct* $\mathcal{D}$*, distinguishing between Games 0 and 1, by internally invoking* $\mathcal{A}$*.* $\mathcal{D}$ *simulates the view of* $\mathcal{A}$ *by forwarding the given messages from the challenger in Game* b_1 *to* $\mathcal{A}$ *or relay the oracle accesses from* $\mathcal{A}$*. However,* $\mathcal{D}$ *additionally receives two challenge messages* $m_{\mathsf{idx}^*,0}, m_{\mathsf{idx}^*,1}$ *from* $\mathcal{A}$ *and needs to return SKE encryption of either message. When* $\mathcal{A}$ *outputs* $(\mathsf{idx}^*, T^*_{\mathsf{idx}^*}, m_{\mathsf{idx}^*,0}, m_{\mathsf{idx}^*,1})$*,* $\mathcal{D}$ *submits* $(\mathsf{idx}^*, T^*_{\mathsf{idx}^*})$ *to the challenger and receives a key denoted by* key^**, which is* $\mathsf{RO}([k^*]_T)$ *if* $b_1 = 0$ *and a random key otherwise.* $\mathcal{D}$ *uniformly samples* $b_2 \leftarrow\$ \{0,1\}$ *and encrypts* m_{idx^*,b_2} *under* key^**, i.e.,* $\mathsf{ct}_{m_{\mathsf{idx}^*,b_2}} \leftarrow \mathsf{SKE.Enc}(\mathsf{key}^*, m_{\mathsf{idx}^*,b_2})$*.* $\mathcal{A}$ *finally outputs* b_2'*.* $\mathcal{D}$ *outputs* $b_1' = 0$ *if* $b_2 = b_2'$ *and* $b_1' = 1$ *otherwise.*

When $b_1 = 1$*, since there is no data associated with* key^*_1 *used for SKE encryption in the view of* $\mathcal{A}$*, it is obvious that the probability of* $b_2 = b_2'$ *is bounded by* ϵ_{SKE}*, i.e.,* $\Pr[b_2 = b_2' \mid b_1 = 1] \leq \frac{1}{2} + \epsilon_{\mathsf{SKE}}$*. Thus, the following holds:*

$$\Pr[\mathcal{D} = b_1] = \Pr[\mathcal{D} = 0 \mid b_1 = 0]\Pr[b_1 = 0] + \Pr[\mathcal{D} = 1 \mid b_1 = 1]\Pr[b_1 = 1]$$
$$= \frac{1}{2}\{\Pr[b_2 = b_2' \mid b_1 = 0] + (1 - \Pr[b_2 = b_2' \mid b_1 = 1])\}$$
$$\geq \frac{1}{2}\{\Pr[b_2 = b_2' \mid b_1 = 0] + (\frac{1}{2} - \epsilon_{\mathsf{SKE}})\}$$

Since $\Pr[\mathcal{D} = b_1] = \epsilon_{\mathcal{D}}$*, from Inequality 14,*

$$\frac{1}{2}\{\Pr[b_2 = b_2' \mid b_1 = 0] + (\frac{1}{2} - \epsilon_{\mathsf{SKE}})\} \leq \frac{1}{2} - \frac{(1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{FUber}} - \epsilon_{\mathsf{DLP}})}{2}$$

$$\Leftrightarrow \Pr[b_2 = b_2' \mid b_1 = 0] \leq \frac{1}{2} + \epsilon_{\mathsf{SKE}} - (1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{FUber}} - \epsilon_{\mathsf{DLP}})$$

As $\Pr[b_2 = b_2' \mid b_1 = 0]$ *is the same as the probability that* $\mathcal{A}$ *wins the game* $\boldsymbol{TBTE}^{\mathcal{A}}_{\mathsf{eid}^*}$*, the advantage of* $\mathcal{A}$ *is bounded as follows:*

$$\boldsymbol{Adv}^{\boldsymbol{TBTE}}_{\mathsf{eid}^*,\mathcal{A}} \leq \epsilon_{\mathsf{SKE}} - (1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{FUber}} - \epsilon_{\mathsf{DLP}})$$

This completes the proof of Lemma 1.

Using Theorem 4.1 in [5], we furthermore reduce the hardness of the flexible Uber assumption to that of the $(B, 2)$-DLP.

Theorem 3. *In the AGM model over a bilinear group* $\mathsf{bg} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$*, for every algebraic adversary* $\mathcal{A}$*, if the* $(0, 1)-DLP$ *(Definition 1) is* $\epsilon_{\mathsf{DLP}_1}$*-hard, the* $(B, 2)-DLP$ *is* $\epsilon_{\mathsf{DLP}_2}$*-hard, and there exists a* ϵ_{SKE}*-EAV-secure SKE scheme for a key space* $\mathcal{K}$ *and a message space* $\mathcal{M}$*, the construction of the TBTE scheme for a committee size* $n \in \mathbb{N}$*, a threshold* $t \in [n]$*, a tag space* $\mathcal{T} \subseteq \mathbb{Z}_p$*, a batch size* $B \in \mathbb{N}$*, and a message space* $\mathcal{M}$ *in Section 4 is* $(\epsilon_{\mathsf{SKE}} - (1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{DLP}_1} - \epsilon_{\mathsf{DLP}_2} - \frac{B+2}{p-1}))$*-secure (Definition 9).*

Proof. Recall that q_1 *is the maximum number of queries to* RO_1 *except for ones for* eid^**. From Theorem 4.1 in [5], we can reduce the security of the* $(q_1 + 3, B, 2, 1, B + 2, 0, 0)$*-DF-Uber assumption such that the advantage is bounded by* $\epsilon_{\mathsf{FUber}}$*-hard and the challenge polynomial* $Q_1(\boldsymbol{X})$ *is non-trivial to the hardness of the* $(B, 2)-DLP$ *as follows:*

$$\epsilon_{\mathsf{FUber}} \leq \epsilon_{\mathsf{DLP}_2} + \frac{B + 2}{p - 1}$$

Thus, from Lemma 1, the advantage of $\mathcal{A}$ *is bounded by* $\epsilon_{\mathsf{DLP}_1}, \epsilon_{\mathsf{DLP}_2}, \epsilon_{\mathsf{SKE}}, B, p$*:*

$$\boldsymbol{Adv}^{\boldsymbol{TBTE}}_{\mathsf{eid}^*,\mathcal{A}} \leq \epsilon_{\mathsf{SKE}} - (1 - \mathsf{negl}(\lambda))(1 - \epsilon_{\mathsf{DLP}_1} - \epsilon_{\mathsf{DLP}_2} - \frac{B + 2}{p - 1})$$

Remark 3 (Security loss in the TBTE scheme). As shown in Theorem 3, the security loss of the TBTE scheme construction depends on B *in the term* $\frac{B+2}{p-1}$*. However, assuming that* p *grows exponentially with respect to* λ *and that* B *is bounded by a polynomial on* λ*, this term becomes negligible.*

5 Implementation and Evaluation

We implement our construction of the TBTE scheme in Section 4. The evaluation of our implementation demonstrates that it establishes the efficiency properties defined in Definition 8.

5.1 Implementation

Our implementation is written in Rust. It instantiates the random oracles RO_1 and RO_2 using SHA256 hash function [55]. It employs ChaCha20 [7] as the SKE scheme. The bilinear group is implemented using BLS12-381 [12], a pairing-friendly elliptic curve in the Barreto-Lynn-Scott family [3].

To improve the running time of the algorithms Digest and $\mathsf{Combine}$, of which computational complexities increase at least linearly with respect to the batch size B, the following two optimizations are applied to our implementation. Notably, we assume B is a power of 2, i.e., $B = 2^d$.

The first optimization replaces the evaluation domain of the polynomial $f(x)$, which is derived through the polynomial interpolation of the tag vector $(T_1, \ldots, T_B)$, with $\mathbb{H} = (\omega^0, \ldots, \omega^{B-1})$, where ω is a B-th root of unity. This reduces the computational complexity of polynomial interpolation from $\mathcal{O}(B^2)$ to $\mathcal{O}(B \log B)$ using Number Theoretic Transform (NTT). This also helps the second optimization as described below.

As the second optimization, the algorithm Setup outputs $([\ell_{i,\mathbb{H}}(\tau)]_1)_{i\in[B]}$ instead of $([\tau^{i-1}])_{i\in[B]}$. This allows the algorithm Digest to avoid the polynomial interpolation for $f(X)$ in the same manner as the method in [31]. Specifically, we can obtain the digest $[f(\tau)]_1$ from $(T_1, \ldots, T_B)$ by computing $[f(\tau)]_1 = \Sigma_{i\in[B]} T_i [\ell_{i,\mathbb{H}}(\tau)]_1$, reducing the computational complexity to $\mathcal{O}(B)$. Similarly, in the algorithm $\mathsf{Combine}$, the element $[h_i(\tau)]_1 = [\frac{f(\tau)-T_i}{\tau-i}]_1$ can be computed directly from $(T_1, \ldots, T_B)$ without computing polynomials $f(X)$ and $h(X)$ as described in [31]. When $B = 2^d$ and $\mathbb{H}$ is the evaluation domain, this computation is further optimized because each Lagrange basis is simplified as $\ell_{i,\mathbb{H}}(X) := \frac{X^d-1}{dX^{d-1}(X-\omega^{i-1})}$ [31, 56].

5.2 Evaluation

Evaluation setting: The evaluation was conducted on an n2d-standard-32 instance with 32 virtual cores, running on the AMD Milan CPU platform, and equipped with 128 GB of RAM. We evaluated the algorithms Enc, Digest, $\mathsf{BatchDec}$ and $\mathsf{Combine}$ with parameters $(n, t) = (9, 3)$ and $(n, t) = (99, 33)$, varying the batch size B from 2^{10} to 2^{20}. We measured the average execution time by running each case 10 times. However, the algorithms $\mathsf{Enc}/\mathsf{Digest}$ in the evaluation encrypts/decrypts only one plaintext/ciphertext regardless of B because they can be easily parallelized[8]. For the same reason, the algorithms $\mathsf{BatchDec}$ in the evaluation was executed only for the first committee party for any n and t. Additionally, we measured the data sizes of ct, digest, pd, and the portion of pk whose size is independent of B, namely $([\mathsf{msk}]_2, [\tau \cdot \mathsf{msk}]_2)$.

Evaluation result:

[8] Although the algorithm $\mathsf{Combine}$ takes as input B ciphertexts, it internally performs the same computation for every ciphertext except for the computation of $[\sigma]_1$, whose complexity is independent of B. Notably, the polynomial interpolation for $f(X)$ is removed after the optimizations described above since we can directly compute $[h_i(\tau)]_1$ from the tag vector.

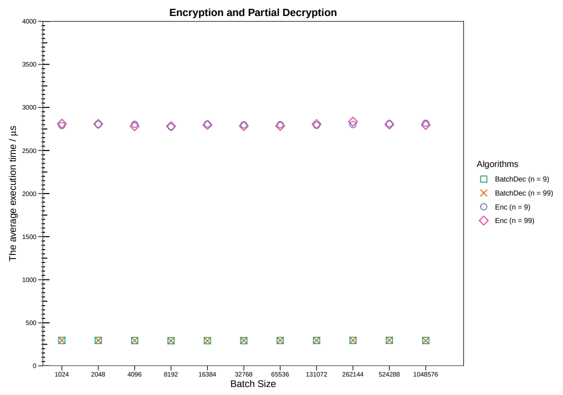

Fig. 1. The average execution times of algorithms Enc and $\mathsf{BatchDec}$ for each committee and batch sizes, where n denotes the committee size.

Figure 1 shows the changes in average execution time with respect to the batch size B for each algorithm and each parameters (n, t). It demonstrates that the execution times of the algorithms $\mathsf{BatchDec}$ and Enc, respectively, remain constant regardless of n and B. The former result supports our claim that the TBTE scheme can maintain a constant computational cost for each committee party. However, the latter result contradicts the theoretical analysis in Theorem 2 that the running time of the algorithm Enc is bounded by a polynomial on $\log B$. This difference can be explained by the fact that our implementation always represents idx as a 64-bit unsigned integer. Therefore, by adjusting the size of idx to match the bit size of B, we can improve the execution time of the algorithm Enc when B is smaller than 2^{63}.

The average execution time of the algorithm Digest in Figure 2 increases lineally with B, but remains constant with respect to n. This is because the public key in our implementation contains $([\ell_{i,\mathbb{H}}(\tau)]_1)_{i \in [B]}$, allowing the algorithm to compute the digest in $\mathcal{O}(B)$ by calculating the inner product between $([\ell_{i,\mathbb{H}}(\tau)]_1)_{i \in [B]}$ and the tag vector $(T_1, \ldots, T_B)$. This removes the $\mathcal{O}(B \log B)$ computation that would otherwise be required due to NTT.

The average execution time of the algorithm $\mathsf{Combine}$ in Figure 2 increases proportionally with B. For the same B, the execution time is larger when $n = 99$ compared to $n = 9$. However, the difference due to n decreases as B becomes larger. In that algorithm, the computation of σ depends on n but not on B, while the other processes behave inversely. Therefore, as B increases, the proportion of execution time spent on the B-dependent processes grows, which can explain the decreasing difference in execution time as B becomes larger.

Table 2 summarizes the data sizes of ct, digest, and the portion of pd whose size is independent of B. Notably, those sizes remain constant regardless of B and n, which are consistent with the theoretical analysis in Theorem 2, except for the

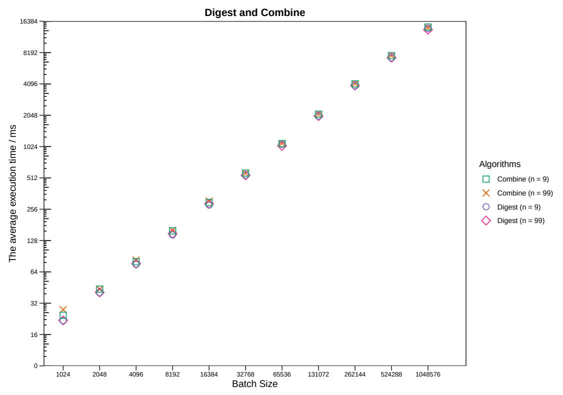

Fig. 2. The average execution times of algorithms Digest and $\mathsf{Combine}$ for each committee and batch sizes, where n denotes the committee size.

Table 2. Measured data sizes of the ciphertext (ct), the digest (digest), the partial decryption (pd), and the portion of the public key (pk) whose size is independent of the batch size B.

data	size
ct	284 byte
digest	48 byte
pd	56 byte
pk (B-independent)	192 byte

data size of ct. The measured size of ct is constant since idx in ct is represented as a 64-bit unsigned integer in our implementation as described above.

In summary, the evaluation results support the theoretical analysis of the efficiency of our scheme, as stated in Theorem 2. Importantly, they demonstrate that the computation and communication costs par committee party remain constant, independent of both B and n.

6 Discussion

Application of TBTE: as theoretical results in Section 4 and empirical evidence in Section 5.2 show, the TBTE scheme ensures constant communication and computation costs for each committee party, regardless of the committee size or number of batched ciphertexts. This property can extend to other cryptographic schemes that use a committee.

One such example is one-time programs based on a committee, as proposed in [18]. Briefly, a one-time program scheme transforms a private circuit with ℓ-bit input into a form that allows evaluation on a single input $\mathbf{x} \in \{0,1\}^{\ell}$ without revealing any information other than the output. The compiled circuit consists of a garbled circuit [60] and 2ℓ garbled inputs, each of which is encrypted under threshold PKE[9]. To evaluate the compiled circuit on $\mathbf{x}$, the committee produces partial decryptions for ℓ encrypted garbled inputs, each corresponding to the bit x_i. This allows evaluation of the garbled circuit on $\mathbf{x}$.

In a traditional threshold PKE, as partial decryptions are generated for each ciphertext, the computational and communication costs par committee party increase with input size ℓ. However, by using the TBTE scheme and assigning each input bit x_i to the i-th tag T_i, where $T_i = x_i$ for every $i \in [\ell]$, those costs remain constant.

This adaptation for the one-time program scheme addresses a common limitation of schemes shared by SWE [29], batched-threshold encryption [20], and TBTE. Since partial decryptions are shared among multiple decryptors, the ciphertexts cannot be privately decrypted by specific individuals but must be decrypted by anyone who has access to the partial decryptions. This limitation is problematic in scenarios where the committee should dynamically restrict who can decrypt each ciphertext, such as in data integrity audits [19] or committee-based WE [18].

We address this using one-time program as follows:

1. An encryptor compiles a circuit that takes as input a user's public key for a standard PKE scheme and outputs an encryption of the hardcoded private message under that public key.
2. The encryptor publishes the compiled circuit as a ciphertext.
3. A decryptor samples a pair of secret and public keys with a standard PKE scheme and generates a proof to claim that the decryptor is legitimate[10].
4. If the proof submitted by the decryptor is valid, the committee generates partial decryptions for a digest of a tag vector consisting of the decryptor's public key bits.
5. Once the partial decryptions are available, each user evaluates the compiled circuit, retrieves the encrypted message, and decrypts it using their private key.

Notably, even when multiple compiled circuits are evaluated simultaneously, the computational and communication costs for each committee party remain constant, as the tag vector containing all decryptors' public keys is compressed into a constant-size digest.

Drawback of TBTE: compared to the existing threshold PKE schemes proposed in [37, 51], the TBTE scheme has the drawback of requiring either a trusted authority or interaction among committee parties during the setup process. This implies that the TBTE scheme is not well-suited for applications where the parties of the committee change frequently.

[9] Strictly, the scheme in [18] uses a threshold-based key derivation algorithm rather than threshold PKE for deriving a secret key for each input bit.

[10] To keep this verification cost constant, users must provide a zk-SNARKs proof claiming that the tag vectors in the digest meet the required conditions, as described in Remark 1.

Interestingly, the work of [37] also adopts a similar approach to ours, namely expressing the conditions to be verified as linear equations over bilinear groups, to construct their threshold PKE scheme. However, their scheme is not compatible with ours because it assumes that a public key is specified not by the encryptor but by decryptor. Since our scheme involves multiplication between the public key in $\mathbb{G}_2$ and the element in $\mathbb{G}_1$ provided by the decryptor as shown in Equation 4, it is unclear whether the product of that element in $\mathbb{G}_1$ and the public key provided by the decryptor can be expressed in such a linear equation.

7 Conclusion

We review previous research on threshold PKE schemes and identify that they fail to maintain constant communication and computation costs per committee member, regardless of the committee size and the number of batched ciphertexts being decrypted under distinct decryption policies. This inefficiency leads to higher participation costs due to increased hardware requirements as the number of ciphertexts to be decrypted per unit time grows with demand. Consequently, the number of committee members and their decentralization become limited, increasing the likelihood of adversarial collusion with more than the threshold number of parties.

To address this issue, we propose a new threshold encryption scheme called TBTE. It achieves constant costs by defining a constant-size digest for a vector of tags, each associated with a distinct ciphertext along with an index, and generating partial decryptions for the digest. This allows for the decryption of any ciphertext where the associated tag exists at the associated index in the tag vector.

Our TBTE scheme construction is built in the random oracle model, utilizing a bilinear group and a SKE scheme. We prove its security in the algebraic group model, assuming the hardness of the (q_1, q_2)-discrete logarithm problem and the EAV-security of the SKE scheme.

We implemented and evaluated our construction. The results show that for committee sizes of both 9 and 99, with batch sizes ranging from 2^{10} to 2^{20}, each committee party receives and transmits data of 48 bytes and 56 bytes, respectively, with the constant execution time to generate a partial decryption.

The TBTE scheme can be also applied to other cryptographic schemes, such as committee-based one-time programs, while maintaining constant costs for each committee party. This one-time program restricts which decryptors can decrypt each ciphertext while still allowing partial decryptions to be shared among multiple decryptors.

However, the TBTE scheme is not well-suited for applications where the committee parties change frequently because it requires interaction among the committee parties during the setup process. Developing a non-interactive setup for the TBTE scheme is an open problem for future research.

Acknowledgments. This work has been supported by Endowed Chair for Blockchain Innovation and the Mohammed bin Salman Center for Future Science and Technology for Saudi-Japan Vision 2030 (MbSC2030) at The University of Tokyo.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: New constructions and applications. In: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34. pp. 69–100. Springer (2015)
2. Abusalah, H., Fuchsbauer, G., Pietrzak, K.: Offline witness encryption. In: International Conference on Applied Cryptography and Network Security. pp. 285–303. Springer (2016)
3. Barreto, P.S., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3. pp. 257–267. Springer (2003)
4. Bartusek, J., Ishai, Y., Jain, A., Ma, F., Sahai, A., Zhandry, M.: Affine determinant programs: a framework for obfuscation and witness encryption. In: 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Schloss-Dagstuhl-Leibniz Zentrum für Informatik (2020)
5. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: Annual International Cryptology Conference. pp. 121–151. Springer (2020)
6. Benhamouda, F., Lin, H.: Mr nisc: multiparty reusable non-interactive secure computation. In: Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part II 18. pp. 349–378. Springer (2020)
7. Bernstein, D.J., et al.: Chacha, a variant of salsa20. In: Workshop record of SASC. vol. 8, pp. 3–5. Citeseer (2008)
8. Blakley, G.R.: Safeguarding cryptographic keys. In: Managing requirements knowledge, international workshop on. pp. 313–313. IEEE Computer Society (1979)
9. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 440–456. Springer (2005)
10. Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Cryptographers' Track at the RSA Conference. pp. 226–243. Springer (2006)
11. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of cryptology **17**, 297–319 (2004)
12. Bowe, S.: Bls12-381: New zk-snark elliptic curve construction. `https://electriccoin.co/blog/new-snark-curve/` (2017), accessed: 2024-09-28
13. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive (2017)
14. Boyen, X.: The uber-assumption family: A unified complexity framework for bilinear groups. In: International Conference on Pairing-Based Cryptography. pp. 39–56. Springer (2008)

15. Campanelli, M., David, B., Khoshakhlagh, H., Konring, A., Nielsen, J.B.: Encryption to the future: a paradigm for sending secret messages to future (anonymous) committees. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 151–180. Springer (2022)
16. Campanelli, M., Fiore, D., Khoshakhlagh, H.: Witness encryption for succinct functional commitments and applications. In: IACR International Conference on Public-Key Cryptography. pp. 132–167. Springer (2024)
17. Cao, Z.: A threshold key escrow scheme based on public key cryptosystem. Science in China Series E: Technological Sciences **44**, 441–448 (2001)
18. Cerulli, A., Connolly, A., Neven, G., Preiss, F.S., Shoup, V.: vetkeys: How a blockchain can keep many secrets. Cryptology ePrint Archive (2023)
19. Chen, Y., Liu, H., Wang, B., Sonompil, B., Ping, Y., Zhang, Z.: A threshold hybrid encryption method for integrity audit without trusted center. Journal of Cloud Computing **10**, 1–14 (2021)
20. Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.V.: Mempool privacy via batched threshold encryption: Attacks and defenses. Cryptology ePrint Archive (2024)
21. Choudhuri, A.R., Garg, S., Policharla, G.V., Wang, M.: Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive (2024)
22. Chvojka, P., Jager, T., Kakvi, S.A.: Offline witness encryption with semi-adaptive security. In: International Conference on Applied Cryptography and Network Security. pp. 231–250. Springer (2020)
23. Cohen, R., Doerner, J., Kondi, Y., Shelat, A.: Guaranteed output in o (n) rounds for round-robin sampling protocols. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 241–271. Springer (2022)
24. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE symposium on security and privacy (SP). pp. 910–927. IEEE (2020)
25. Dengguo, F., Weidong, C.: Analysis on the two classes of robust threshold key escrow schemes. Progress on Cryptography: 25 Years of Cryptography in China pp. 137–144 (2004)
26. Derler, D., Slamanig, D.: Practical witness encryption for algebraic languages or how to encrypt under groth–sahai proofs. Designs, Codes and Cryptography **86**(11), 2525–2547 (2018)
27. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Conference on the Theory and Application of Cryptographic Techniques. pp. 120–127. Springer (1987)
28. Desmedt, Y.: Threshold cryptosystems. In: International Workshop on the Theory and Application of Cryptographic Techniques. pp. 1–14. Springer (1992)
29. Döttling, N., Hanzlik, L., Magri, B., Wohnig, S.: Mcfly: verifiable encryption to the future made practical. In: International Conference on Financial Cryptography and Data Security. pp. 252–269. Springer (2023)
30. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
31. Feist, D.: Polynomial commitments and multi-proofs (2021), `https://dankradfeist.de/ethereum/2021/06/18/pcs-multiproofs.html`, accessed: 2024-09-28
32. Fleischhacker, N., Hall-Andersen, M., Simkin, M.: Extractable witness encryption for kzg commitments and efficient laconic ot. Cryptology ePrint Archive (2024)

33. Fouque, P.A., Stern, J.: One round threshold discrete-log key generation without private channels. In: Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4. pp. 300–316. Springer (2001)
34. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38. pp. 33–62. Springer (2018)
35. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive (2019)
36. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 467–476 (2013)
37. Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. In: Annual International Cryptology Conference. pp. 352–386. Springer (2024)
38. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology **20**, 51–83 (2007)
39. Gentry, C., Lewko, A., Waters, B.: Witness encryption from instance independent assumptions. In: Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34. pp. 426–443. Springer (2014)
40. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. pp. 536–553. Springer (2013)
41. Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 305–326. Springer (2016)
42. Groth, J.: Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive (2021)
43. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-snarks. In: Annual International Cryptology Conference. pp. 698–728. Springer (2018)
44. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 177–194. Springer (2010)
45. Madathil, V., Thyagarajan, S.A., Vasilopoulos, D., Fournier, L., Malavolta, G., Moreno-Sanchez, P.: Cryptographic oracle-based conditional payments. Cryptology ePrint Archive (2022)
46. Nikolaenko, V., Ragsdale, S., Bonneau, J., Boneh, D.: Powers-of-tau to the people: Decentralizing setup ceremonies. In: International Conference on Applied Cryptography and Network Security. pp. 105–134. Springer (2024)
47. Nitulescu, A.: zk-snarks: A gentle introduction (2020), `https://api.semanticscholar.org/CorpusID:211530704`

48. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
49. Petitcolas, F.A.: Kerckhoffs' principle. In: Encyclopedia of Cryptography, Security and Privacy, pp. 1–2. Springer (2023)
50. Piet, J., Nair, V., Subramanian, S.: Mevade: An mev-resistant blockchain design. In: 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–9. IEEE (2023)
51. Reyzin, L., Smith, A., Yakoubov, S.: Turning hate into love: compact homomorphic ad hoc threshold encryption for scalable mpc. In: International Symposium on Cyber Security Cryptography and Machine Learning. pp. 361–378. Springer (2021)
52. Saxena, N., Tsudik, G., Yi, J.H.: Threshold cryptography in p2p and manets: The case of access control. Computer Networks **51**(12), 3632–3649 (2007)
53. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
54. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. Journal of Cryptology **15**, 75–96 (2002)
55. of Standards, N.I., (NIST), T.: Federal information processing standards publication 180-4: Secure hash standard. Tech. rep., National Institute of Standards and Technology (NIST) (2015), `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf`, federal Information Processing Standards Publication
56. Szeg, G.: Orthogonal polynomials, vol. 23. American Mathematical Soc. (1939)
57. Tsabary, R.: Candidate witness encryption from lattice techniques. In: Annual International Cryptology Conference. pp. 535–559. Springer (2022)
58. Uberti, G., Luo, K., Cheng, O., Goh, W.: Building usable witness encryption. arXiv preprint arXiv:2112.04581 (2021)
59. Vaikuntanathan, V., Wee, H., Wichs, D.: Witness encryption and null-io from evasive lwe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 195–221. Springer (2022)
60. Yao, A.C.C.: How to generate and exchange secrets. In: 27th annual symposium on foundations of computer science (Sfcs 1986). pp. 162–167. IEEE (1986)
61. Yu, K., Tan, L., Yang, C., Choo, K.K.R., Bashir, A.K., Rodrigues, J.J., Sato, T.: A blockchain-based shamir's threshold cryptography scheme for data protection in industrial internet of things settings. IEEE Internet of Things Journal **9**(11), 8154–8167 (2021)
62. Zhen-Fu, C., Jiguo, L.: A threshold key escrow scheme based on eigamal public key cryptosystem. Chinese Journal of Computers **25**(4), 346–350 (2002)
63. Zhou, L., Haas, Z.J.: Securing ad hoc networks. IEEE network **13**(6), 24–30 (1999)