Registration-based Encryption with Logarithmic Registration

Xingfu Yan
South China Normal University
xfyan78@163.com

Guiqiu Liao
South China Normal University
liaogq1@163.com

Fan Yang
South China Normal University
yf202511@163.com

Haining Yang
Shandong University
hainingyang@sdu.edu.cn

Zheng Gong
South China Normal University
cis.gong@gmail.com

Fucai Luo
Zhejiang Gongshang University
lfucai@126.com

Bowen Zhao
Guangzhou University
bwinzhao@gmail.com

Abstract

Registration-based encryption (RBE) preserves IBE's identity-only encryption interface while eliminating key escrow: users generate their own keys, and a Key Curator manages only registered public keys. However, this design introduces a new challenge: user registration, since every joining user requires KC to update the registry. Although theoretical RBE constructions achieve polylogarithmic registration, they rely on costly cryptographic techniques and remain impractical. In existing deployable black-box RBE, registration takes $O(\sqrt{N})$ time (N denotes the maximum user capacity). This gap between theoretical registration efficiency and deployability makes registration a key performance bottleneck for large-scale deployments of pairing-based RBE.

This work presents a black-box pairing-based RBE construction under the decisional (R,S,T,f)-Diffie-Hellman assumption. Build on the multilinear tree, when registering a user's public key, KC updates only the corresponding root-to-leaf path associated with the user. Therefore, registration in our construction takes $O(\log\sqrt{N})$ time for the maximum user capacity N. Moreover, we provides membership and non-membership verification, and their batch verification supports to check registration status of multiple users. Experiments show registration and key generation complete in milliseconds when $N = 10^7$. Compared with Glaeser et al. (CCS'23), our scheme accelerates registration by up to 158× and key generation by up to 230× with practical overheads.

1 Introduction

Public-key encryption requires senders to locate and authenticate recipient's public key, typically through a public-key infrastructure. Identity-based encryption (IBE) offers a simpler interface: a sender encrypts messages by recipient's identity and some public system parameters [6,11,40]. A broad line of work has subsequently improved the security and efficiency of IBE under diverse assumptions [14,22,23,43,44]. This convenience of IBE comes at the cost of key escrow: an private-key generator (PKG) holds a master secret key from which it can derive decryption keys for arbitrary identities. It means that PKG is the core of trust and may decrypt any message sent through the system. Certificateless encryption, accountable-authority IBE, threshold PKGs, and multi-authority IBE reduce or distribute this trust, but retain secret authority state or non-collusion assumptions [1,27,28,31,38]. Registration-based encryption (RBE) was introduced to remove this trust assumption [20].

In RBE, each user generates a key pair locally and registers only its identity and public key with a *key curator* (KC). The KC aggregates registered public keys into an evolving public parameter but holds no user secret; encryption uses the recipient identity and current public parameter, while decryption combines the user's secret key with KC-provided auxiliary information. RBE thus shifts the central challenge to maintaining a compact, dynamic public registry.

Early RBE constructions established feasibility from strong tools such as indistinguishability obfuscation and garbled circuits, but remained far from deployable [20, 21]. In 2023, Glaeser et al. [25] gave the first deployable black-box RBE schemes for bounded identity sets from pairing-based constructions, together with a prototype supporting millions of users. Subsequent work has advanced other axes of the RBE, including extending efficient RBE from bounded integer identity spaces to arbitrary identity strings [17], constructing RBE in the plain model without a trusted setup [16], and developing lattice-based constructions with plausible post-quantum security security [32,42,46].

Within the deployable pairing-based line, the construction of Glaeser et al. [25] remains the main efficient benchmark. They achieve millisecond-level encryption and decryption, showing that pairing-based RBE can be practical. However, the registration procedures (usually performed by KC) remain substantially more expensive, requiring $O(\sqrt{N})$ group operations, where N denotes the maximum user capacity N. Their efficient-update variant reduced the update frequency performed by each user to $O(\log N)$, but increases the registration cost to $O(\sqrt{N}\log N)$. This bottleneck also reflects a

gap between the original efficiency goal of RBE and existing concretely efficient constructions: Although polylogarithmic-time registration has been achieved theoretically under standard assumptions [20, 21], realizing this goal in a concretely efficient black-box pairing-based construction has remained unresolved. This motivates the main question of this paper:

> *Can we design a black-box pairing-based RBE with logarithmic-time registration, while keeping the other core procedures practical for large-scale deployments?*

1.1 Our Results

To answer the question above, we construct a new RBE with efficient registration. Our contributions are as follows.

- **RBE with $O(\log \sqrt{N})$ registration:** We propose a new black-box pairing-based RBE construction for the bounded identity set of size N, where N denotes the maximum user capacity and is an arbitrary polynomially-bounded value. Our scheme operates in the common reference string model and its security is based on the the hardness of decisional (R,S,T,f)-Diffie–Hellman problem. The registration takes $O(\log \sqrt{N})$ time. To the best of our knowledge, our scheme achieves the lowest registration complexity among existing deployable pairing-based RBE constructions. As an additional benefit, the key generation also runs in time $O(\log \sqrt{N})$, while encryption, decryption, and updates remain efficient in practice. The common reference string and the public parameters consist of at most $O(\sqrt{N})$ group elements.

- **Verifiability and batch verification:** We extend our RBE construction with membership and non-membership verification for checking the registration status of identities and their public keys. Furthermore, motivated by the batch aggregation methods based on Inner-Product Argument (IPA) [9, 41], we optimize the preprocessing of IPA and design further a batch verification to check the registration status of multiple identities. Compared with the original IPA-based methods, our optimization substantially reduces the pairing computation during batch verification, while leaving the asymptotic group-exponentiation cost unchanged. In our evaluation, for a batch of 2000 identities, our optimization achieves a 7.8× speedup in overall batch-verification time with the unoptimized verification.

- **Implementation and evaluation:** We implement our RBE and compare its performance with that of the base and efficient-update variants in [25]. At a maximum user capacity of $N = 10^7$ users in our RBE, the registration of a single public key for KC takes 20.46 ms, while key generation takes 1.82 ms. This achieves up to 158× speedup in registration and up to 230× speedup in key generation over prior work. Encryption, decryption, and update generation take 6.75 ms, 24.85 ms, and 180.11 ms, respectively. The auxiliary data occupies 11.57 GB when $N = 10^7$, while it is 1.9 TB and 16.3 GB for the two variants in [25]. The common reference string, public parameter and a single ciphertext occupy 1.3 MB, 184 KB and 1.93 KB, respectively.

1.2 Related Work

The first RBE construction used indistinguishability obfuscation, and the subsequent standard-assumption construction made non-black-box use of garbled circuits [20, 21]. Subsequent optimization reduced the algorithm overhead, especially the ciphertext size, though it remained considerably large. [12], and verifiable RBE enabled the KC to prove that an identity is registered or unregistered [26], but these schemes remained far from a deployable black-box realization. Glaeser et al. [25] gave the first concretely efficient, black-box RBE from standard assumptions in bilinear groups and demonstrated a prototype for millions of users. Fiore et al. [17] used cuckoo commitments to extend efficient RBE to large identity spaces. Lower bounds show that the trade-offs among the public parameter size, update frequency, and assumptions are fundamental to any construction [29, 35]; plain model constructions further illustrate the cost of removing the trusted setup, namely the need to weaken the security model and the requirement to update the keys of nearly all users upon each user registration [16].

Post-quantum RBE began with the LWE-based laconic construction of Döttling et al. [15]. Klooß et al. subsequently obtained a scalable construction from LWE, and Zhang et al. improved the concrete ciphertext and online costs through lattice homomorphic encodings [32, 46]. Tomita gives a further LWE-based efficiency improvement [42]. These results pursue the complementary goal of replacing pairings with plausibly post-quantum assumptions.

Beyond identities. The registration paradigm now extends well beyond IBE. Registered Attribute-Based Encryption (RABE) supports fine-grained policies while allowing users to generate their own keys [30,48]. Subsequent work constructed RABE from witness encryption, supported unbounded predicates, and reduced the structured common reference string (CRS) cost [4, 18, 19]. Registered functional encryption moves beyond predicates to inner products and other functions, with applications including traitor tracing [8, 13]. Multi-authority RABE separates attribute domains among curators [34], while recent lattice-based constructions pursue RABE for circuits, unbounded-depth computation, and unbounded user populations [10, 37, 45, 47]. Revocation and certified deletion have also begun to be studied in the registered setting [3, 33, 36].

This broader literature confirms that compact, efficiently maintained public aggregation is a reusable problem across registered cryptography.

2 Preliminaries

Notation. Let $\lambda \in \mathbb{N}$ denote the security parameter, and let $\mathsf{negl}(\lambda)$ denote a negligible function. For any $n \in \mathbb{N}$, we write $[n] = \{1, 2, \ldots, n\}$. Let $\mathbb{Z}_p^*$ denote the multiplicative group modulo p. For a finite set $\mathcal{X}$, the notation $x \leftarrow \mathcal{X}$ means x is sampled uniformly at random from $\mathcal{X}$; $y \leftarrow \mathcal{A}(x)$ denotes the output of randomized algorithm $\mathcal{A}$ on input x. Bold symbols such as $\mathbf{v}$, $\mathbf{B}$ stand for vectors or matrices. Given a vector $\mathbf{y} = (y_1, \ldots, y_n)$ and a scalar r, we define $\mathbf{y}^r = (y_1^r, \ldots, y_n^r)$ and let $\langle \mathbf{y}, \mathbf{y}' \rangle = \prod_{j=1}^{n} e(y_j, y'_j)$ denote a pairing product. Vector concatenation is written as $[\mathbf{y}_1 \| \mathbf{y}_2]$. Let N be the maximum number of users in the system.

2.1 Multilinear extension

For $n = 2^\ell$, let $\mathbf{x} = (x_0, \ldots, x_{n-1})$ and let $\mathbf{v}_i = (v_{i,\ell} \ldots v_{i,1}) \in \{0,1\}^\ell$ denote the ℓ-bit representation of index i. The multilinear extension of vector $\mathbf{x}$ is a unique polynomial $\widetilde{f}$ with degree at most 1 in each variable, as follows:

$$\widetilde{f}(\mathbf{X}) = \sum_{i=0}^{n-1} x_i \chi_i(X_\ell, \ldots, X_1) = \sum_{i=0}^{n-1} x_i \chi_i(\mathbf{X}),$$

where $\chi_i(\mathbf{X})$ is selector multinomials defined as

$$\chi_i(\mathbf{X}) = \prod_{k=1}^{\ell} \chi_{i,k} = \prod_{k=1}^{\ell} \big(v_{i,k} X_k + (1 - v_{i,k})(1 - X_k)\big),$$

such that

$$\chi_{i,k} = \begin{cases} X_k, & \text{if } v_{i,k} = 1 \\ 1 - X_k, & \text{if } v_{i,k} = 0 \end{cases}.$$

Indeed, $\widetilde{f}(\mathbf{v}_i) = x_i$, because $\chi_i(\mathbf{v}_j) = 1$ for $i = j$, otherwise $\chi_i(\mathbf{v}_j) = 0$. Thus, $\widetilde{f}$ acts as an algebraic selector for $\mathbf{x}$: evaluating it at the Boolean encoding $\mathbf{v}_i$ of an index i returns the corresponding vector component x_i. For example, the extension of (x_0, x_1, x_2, x_3) is

$$\begin{aligned} \widetilde{f}(X_2, X_1) = & x_0(1 - X_2)(1 - X_1) + x_1(1 - X_2)X_1 \\ & + x_2 X_2 (1 - X_1) + x_3 X_2 X_1. \end{aligned}$$

In this example, $\widetilde{f}(0,0) = x_0$, $\widetilde{f}(0,1) = x_1$, $\widetilde{f}(1,0) = x_2$, and $\widetilde{f}(1,1) = x_3$.

MLE Decomposition. Let $\mathbf{x}^{(0)} = (x_0, \ldots, x_{n/2-1})$ and $\mathbf{x}^{(1)} = (x_{n/2}, \ldots, x_{n-1})$ be the left and right halves of $\mathbf{x}$, and let $\widetilde{f}_0$ and $\widetilde{f}_1$ denote their multilinear extensions in the first $\ell - 1$ bits. The MLE decomposes as

$$\begin{aligned} \widetilde{f}(\mathbf{X}) &= (1 - X_\ell)\widetilde{f}_0(X_{\ell-1}, \ldots, X_1) + X_\ell \widetilde{f}_1(X_{\ell-1}, \ldots, X_1) \\ &= (\widetilde{f}_1 - \widetilde{f}_0)(X_\ell - 1) + \widetilde{f}_1 \\ &= (\widetilde{f}_1 - \widetilde{f}_0)X_\ell + \widetilde{f}_0. \end{aligned} \tag{1}$$

Thus, for subscripts that belong to the same vector before MLE decomposes, $\widetilde{f}(\mathbf{X})$ division by $X_\ell - v_{i,\ell}$ will yield the same quotient $\widetilde{f}_1 - \widetilde{f}_0$.

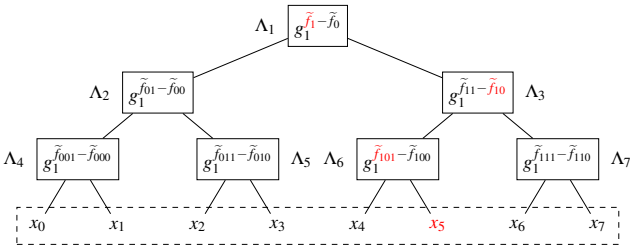


Figure 1: MLT [41] for an 8-entry vector. Internal nodes are openings, and the internal nodes on a leaf-to-root path form the corresponding evaluation proof. If x_5 is added to this vector, only the nodes related to the paths marked in red need to be modified to update all affected openings (note that x_i are not nodes on the tree; they are merely written to intuitively illustrate the path for each index).

Multilinear tree (MLT). In Hyperproofs, quotient proofs are organized into a tree structure by recursively applying the above decomposition: Dividing $\widetilde{f}$ by X_ℓ or $X_\ell - 1$ yields the same quotient $\widetilde{f}_1 - \widetilde{f}_0$, and the opening $g_1^{\widetilde{f}_1 - \widetilde{f}_0}$ is then stored at the root node; the corresponding remainders are $\widetilde{f}_0$ and $\widetilde{f}_1$, respectively. Then perform the same division operation recursively on the remainder. Each tree node stores the corresponding opening; therefore, the proof for a certain index consists of all quotient proofs encountered along its corresponding path. Since the indices share some quotient proofs, the time complexity of constructing all proofs can be reduced to $O(n \log n)$. Modifying a single record only requires updating $O(\log n)$ commitments along its path [41]. Fig. 1 shows an example for an 8-entry vector.

2.2 Bilinear Groups

Let $\mathcal{BG}(1^\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ denote the generation of a bilinear group tuple of prime order p, with g_i being a generator of $\mathbb{G}_i$. The pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies $e(u^a, w^b) = e(u, w)^{ab}$ for all $u \in \mathbb{G}_1$, $w \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$. By bilinearity, $e(u,h)e(v,h) = e(uv,h)$ for all $u, v \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Our construction is designed over Type-III bilinear groups, where no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$. Its security is based on the decisional (R, S, T, f)-Diffie–Hellman assumption in this setting, whose hardness has been demonstrated in the generic bilinear group model.

Definition 2.1 (Decisional (R,S,T,f)-Diffie-Hellman Assumption. [7])**.** Let $c \in \mathbb{N}$ and let $X_1,\ldots,X_c$ be formal variables. Let $R = \langle r_1,\ldots,r_{|R|}\rangle$, $S = \langle s_1,\ldots,s_{|S|}\rangle$, $T = \langle t_1,\ldots,t_{|T|}\rangle$ be three tuples of multivariate polynomials over $\mathbb{F}_p$ in the variables $X_1,\ldots,X_c$, and let $f \in \mathbb{F}_p[X_1,\ldots,X_c]$ be a target polynomial, where $r_1 = s_1 = t_1 = 1$. For a secret vector $\mathbf{x} = (x_1,\ldots,x_c) \leftarrow \mathbb{F}_p^c$ sampled uniformly at random, define the *instance vector*

$$U(\mathbf{x}) = \left(g_1^{r_1(\mathbf{x})},\ldots,g_1^{r_{|R|}(\mathbf{x})}; g_2^{s_1(\mathbf{x})},\ldots,g_2^{s_{|S|}(\mathbf{x})}; g_T^{t_1(\mathbf{x})},\ldots,g_T^{t_{|T|}(\mathbf{x})}\right),$$

where $g_T = e(g_1,g_2)$. The challenge target is $V(\mathbf{x}) = g_T^{f(\mathbf{x})}$.

The advantage of a PPT algorithm $\mathcal{A}$ in solving the decisional (R,S,T,f)-Diffie–Hellman problem is

$$\mathrm{Adv}_{\mathcal{A}}^{(R,S,T,f)}(\lambda) = \Big|\ \Pr\big[\mathcal{A}(U(\mathbf{x}), g_T^{f(\mathbf{x})}) = 1\big] - \Pr\big[\mathcal{A}(U(\mathbf{x}), V') = 1\big]\Big|.$$

2.3 Generic Lower Bound

Under the Type-III groups, f is *dependent* on $\langle R,S,T\rangle$ if there exist constants $\{a_{i,j}\}_{i\in[|R|],j\in[|S|]}$ and $\{b_k\}_{k\in[|T|]}$ such that

$$f = \sum_{i=1}^{|R|}\sum_{j=1}^{|S|} a_{i,j}\, r_i\, s_j \;+\; \sum_{k=1}^{|T|} b_k\, t_k, \tag{2}$$

and *independent* otherwise. In the Type-III setting, products of the form $r_i r_{i'}$ or $s_j s_{j'}$ are disallowed, as encoding them would require an efficient isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ (or its inverse), which is assumed not to exist. For the polynomial tuples R,S,T and the target f, let $d_R = \max_{r_i\in R}\deg(r_i)$, and define d_S, d_T, d_f analogously. The following master theorem establishes the generic hardness of Definition 2.1.

Theorem 1 (Master Theorem [5, 7])**.** Let $d = \max\{d_R + d_S, d_T, d_f\}$. If f is independent of $\langle R,S,T\rangle$, then any algorithm $\mathcal{A}$ that solves the decisional (R,S,T,f)-Diffie–Hellman problem in the generic bilinear group model, using at most q queries to the group-operation and pairing oracles, satisfies

$$\mathrm{Adv}_{\mathcal{A}}^{(R,S,T,f)}(\lambda) \le \frac{(q+2c+2)^2\cdot d}{2p}.$$

Consequently, when c, d, and q are polynomially bounded in λ and $p = \Theta(2^\lambda)$, the advantage is negligible.

3 Technical Overview

Registration bottleneck. The main constructions of prior pairing-based RBEs ultimately rely on the blockwise vector-commitment framework of Glaeser et al. [25], including its extension to arbitrary identity strings by Fiore et al. [17]. Although these scheme support practical encryption and decryption, registration remains costly: adding a public key requires the KC to recompute the auxiliary information (it is openings in VC) for all positions in the affected chunk. Consequently, registration requires $O(n)$ computation, which becomes $O(\sqrt{N})$ when $n = \sqrt{N}$. This falls short of the original RBE goal of registration time polylogarithmic in the user capacity [20, 21].

Vector Commitments (VC) based on MLT. Inspired by Merkle-tree-based RBE constructions, which achieve polylogarithmic registration in registration [16, 20, 21], we revisit tree-structured registration. However, turning Merkle-tree verification into encryption requires cryptographic tools such as indistinguishability obfuscation or garbled circuits, making these constructions costly and far from deployable in practice.

Hyperproof VC [41] provides a possible tree-structured way forward. It maintains all openings in a ℓ-depth multilinear tree (MLT, it is the auxiliary information in our RBE), with each path corresponding to an opening of a position. Therefore, changing a position only requires updating nodes of the corresponding root-to-leaf path to complete the update of all openings. Importantly, an opening in MLT is verified through a pairing equation, which provide a possible method for constructing the encryption of RBE in a black-box way.

Given $N = 2^\ell$, a vector $\mathbf{x} = (x_0,\ldots,x_{N-1})$, and a secret vector $\mathbf{s} = (s_1,\ldots,s_\ell)$, its commitment, obtained by committing to its multilinear extension $\widetilde{f}$, is defined as

$$C = g_1^{\widetilde{f}(\mathbf{s})} = g_1^{\sum_{i=0}^{N-1} x_i\chi_i},$$

whereas the opening of the position i is $\pi_i = (\pi_{i,1},\ldots,\pi_{i,\ell}) \in \mathbb{G}_1^\ell$, where $\pi_{i,j} = g_1^{q_{i,j}(\mathbf{s})}$ and $q_{i,1},\ldots,q_{i,\ell}$ are a set of quotient polynomials associated with opening position i. Since $\widetilde{f}(\mathbf{v}_i) = x_i$ (where $\mathbf{v}_i = v_{i,\ell}v_{i,\ell-1}\cdots v_{i,1}$ is the binary representation of i), these quotient polynomial satisfy

$$\widetilde{f}(\mathbf{s}) - \widetilde{f}(\mathbf{v}_i) = \widetilde{f}(\mathbf{s}) - x_i = \sum_{j=1}^{\ell}(s_j - v_{i,j})q_{i,j}(\mathbf{s}). \tag{3}$$

To check that the value at index i is x_i, the verifier retrieves the ℓ nodes on the corresponding path in the MLT (see Fig. 1) and check if:

$$e(C, g_2) \stackrel{?}{=} e\left(g_1^{x_i}, g_2\right)\prod_{j=1}^{\ell} e\left(\pi_{i,j}, g_2^{s_j - v_{i,j}}\right) \tag{4}$$

where $\{g_2^{s_1-v_{i,1}},\ldots,g_2^{s_\ell - v_{i,\ell}}\}_{i=1}^N$ are public parameters (they are the common reference strings in our RBE), g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Warm-up: RBE with logarithmic registration. To illustrate the core construction, we first consider a single-tree warm-up in which one MLT covers the entire identity space of up to N users. We treat the committed vector

$\mathbf{a} = (a_0, a_1, \cdots, a_{N-1})$ as a registry. The index i is set as the identity of the user. When user i registers, the user samples $a_i \leftarrow \mathbb{Z}_p^*$ and sets $sk_i = a_i$ as the private key, $pk_i = (g_1^{\chi_i})^{a_i}$ to be public key, where $g_1^{\chi_i}$ is included in the common reference string (crs). If μ ($\mu \leq N$) users have registered, $C = \prod_{i=0}^{\mu} pk_i$ is the public parameter. The opening π_i of the position i is computed by KC and necessary for a user to decrypt.

1) Encryption and Decryption We first describe how to design a ciphertext ct and how to decrypt it with the secret key a_i and the opening π_i. Inputting a message m and identity i, the encryption algorithm outputs

$$\mathbf{ct} = (C, A_0, A_1, \ldots, A_\ell, C_m),$$

where (r is the encryption randomness over $\mathbb{Z}_p^*$):

$$C_m = e(g_1, g_2)^r \cdot m, \quad A_0 = e(C, g_2)^r,$$
$$A_j = g_2^{r(s_j - v_{i,j})} \quad (j = 1, \ldots, \ell).$$

When the opening $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$ is valid with respect to the commitment C contained in the ciphertext, only the user holding sk_i can perform decryption from Eq. (4) as follows:

$$e(g_1, g_2)^r = \left(A_0 / \prod_{j=1}^{\ell} e(\pi_{i,j}, A_j) \right)^{a_i^{-1}}$$
$$m = C_m / e(g_1, g_2)^r.$$

2) Registration. The remaining question is how the KC updates C and the MLT when a user i with public key pk_i registers to the system. Although updating C requires only one group multiplication $C \leftarrow C \cdot pk_i$, the opening information must also be updated to maintain valid opening information for the new registry state. Recall that in Hyperproofs [41], i's opening $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$ is actually an ordered tuple of nodes along the root-to-leaf path associated with i, where $\pi_{i,j}$ is the node at level j on this path. We next exploit how the KC updates these path nodes during registration.

Without loss of generality, we take $N = 2^3$ as an example, it is a 3−depth MLT(see Fig.1). Suppose that identities $0, 1$, and 4 are already registered, so the commitment vector $\mathbf{a} = (a_0, a_1, 0, 0, a_4, 0, 0, 0)$. From Eq. (1), $\mathbf{a}$'s multilinear extension on $\mathbf{s} = (s_1, s_2, s_3)$ is

$$\begin{aligned}\widetilde{f}_{\mathbf{a}}(\mathbf{s}) =& a_0(1-s_3)(1-s_2)(1-s_1) + a_1(1-s_3)(1-s_2)s_1 \\ &+ a_4 s_3 (1-s_2)(1-s_1) \\ =& (\widetilde{f}_1 - \widetilde{f}_0)(s_3 - 1) + (\widetilde{f}_{11} - \widetilde{f}_{10})s_2 + (\widetilde{f}_{101} - \widetilde{f}_{100})s_1.\end{aligned}$$

Now identity 5 is joining the system, then $\mathbf{a}$ becomes $\mathbf{a}' = (a_0, a_1, 0, 0, a_4, a_5, 0, 0)$, and its multilinear extension changes by only one selector term (the red indicates the parts required for the update):

$$\widetilde{f}_{\mathbf{a}'}(\mathbf{s}) = \widetilde{f}_{\mathbf{a}}(\mathbf{s}) + a_5 s_3 (1 - s_2) s_1$$

From Fig. 1, registering identity 5 updates only 3 nodes on its path. The updated nodes are computed as follows:

$\mathbf{a}$'s MLT	$\longrightarrow$	$\mathbf{a}'$'s MLT	
$g_1^{\widetilde{f}_1 - \widetilde{f}_0}$	$\longrightarrow$	$g_1^{\widetilde{f}_1 - \widetilde{f}_0 + a_5(1-s_2)s_1}$	$= g_1^{\widetilde{f}_1 - \widetilde{f}_0} \cdot g_1^{a_5(1-s_2)s_1}$
$g_1^{\widetilde{f}_{11} - \widetilde{f}_{10}}$	$\longrightarrow$	$g_1^{\widetilde{f}_{11} - \widetilde{f}_{10} - a_5 s_1}$	$= g_1^{\widetilde{f}_{11} - \widetilde{f}_{10}} \cdot g_1^{-a_5 s_1}$
$g_1^{\widetilde{f}_{101} - \widetilde{f}_{100}}$	$\longrightarrow$	$g_1^{\widetilde{f}_{101} - \widetilde{f}_{100} + a_5}$	$= g_1^{\widetilde{f}_{101} - \widetilde{f}_{100}} \cdot g_1^{a_5}$

From this example, we see that when the user 5 registers, it needs to provide $g_1^{a_5(1-s_2)s_1}, g_1^{a_5 s_1}$, and $g_1^{a_5}$ to hlep KC to update the MLT path. These factors depend on the user's secret value a_5, which is known only to user 5. Like [25], we call them *helping information* for updating the corresponding MLT path and denote it as $\xi_5 = (\xi_{5,1}, \xi_{5,2}, \xi_{5,3})$.

Generally, let $N = 2^\ell$, when a new user i registers, it provide its pk_i and the helping information $\xi_i = (\xi_{i,1}, \ldots, \xi_{i,\ell})$ to KC, where for $j = 1, 2, \cdots, \ell$,

$$\xi_{i,j} = g_1^{a_i \prod_{k=1}^{\ell-j} \chi_{i,j}}.$$

Then KC first checks the correctness of the *helping information* by pairings. Next, it updates the corresponding commitment as $C \leftarrow C \cdot pk_i$ and modifies only the nodes along the root-to-leaf path associated with i. At level j, the corresponding node is updated as:

$$\pi_{i,j} \leftarrow \begin{cases} \pi_{i,j} \cdot \xi_{i,j}, & v_{i,\ell-j+1} = 1 \\ \pi_{i,j} \cdot \xi_{i,j}^{-1}, & v_{i,\ell-j+1} = 0 \end{cases},$$

where $\mathbf{v}_i = v_{i,\ell} v_{i,\ell-1} \cdots v_{i,1}$ is the binary representation of i.

Note. As shown in Hyperproofs [41], each MLT node commits to the difference between the MLEs of its right and left subtrees. In our warm-up, registering a new identity i changes only the nodes on $\bar{i}$'s path. At level j, $\xi_{i,\ell}$ commits to the magnitude of the resulting change: the change is positive when $\bar{v}_{i,\ell-t+1} = 1$ and negative otherwise. Hence, Eq. (7) updates the affected node to the commitment corresponding to the new registration vector.

Amortizing the CRS. The warm-up achieves logarithmic registration, but it is based single-tree, which requires a structured crs containing $O(N)$ group elements. This makes it unsuitable for large-scale deployment. Since crs depends only on the tree capacity, the same crs can be reused across multiple MLTs of equal size. Following the blockwise amortization strategy of Glaeser et al. [25], We partition the capacity-N registry into $B = \lceil \frac{N}{n} \rceil$ chunks, each containing at most n positions. Consequently, when setting the chunk size $n = \sqrt{N}$, our RBE have $O(\sqrt{N})$-size crs and $O(\sqrt{N})$ commitments, and the registration takes $O(\log \sqrt{N})$ time.

4 Definition

In this section, we define the syntax and security of Registration-Based Encryption (RBE), following the formulations in [20, 25]. The definitions of completeness, compactness, and efficiency also follow these formulations and are deferred to Appendix A.1 for brevity. Consistent with their scheme, we require a predefined upper bound N on the identity set size. The formal definition is presented as follows:

Definition 4.1 (Registration-Based Encryption)**.** An RBE scheme has six PPT algorithms: Setup, Gen, Reg, Enc, Upd, and Dec. The Key Curator (KC) executes Reg and Upd, while users execute the remaining algorithms locally except Setup.

- $\mathsf{Setup}(1^\lambda, n) \to \mathsf{crs}$: Takes the security parameter λ and an upper bound N on the number of users as input, outputs the common reference string (crs), which is generated once and for all at the protocol launch phase.[1]
- $\mathsf{Gen}(\mathsf{crs}, i) \to (sk_i, pk_i)$: Takes crs and an identity i as input, outputs a secret key sk_i, a public key pk_i.
- $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i, pk_i) \to \mathsf{pp}'$: Takes crs, the current public parameter pp, the identity i, the public key pk_i, as input, outputs an updated public parameter pp′ and maintain an auxiliary information aux (i.e. multilinear trees). (Initially, $\mathsf{pp} = \perp$ and $\mathsf{aux} = \perp$.)
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, i, m) \to \mathbf{ct}$: Takes crs, the public parameter pp, the recipient's identity i, and a message m as input, outputs a ciphertext **ct**.
- $\mathsf{Upd}^{[\mathsf{aux}]}(\mathsf{pp}, i) \to \mathbf{u}_i$: Takes the current public parameter pp and an identity i as input. It reads the auxiliary information aux in a read-only manner and outputs an update information $\mathbf{u}_i$ to assist user i in decryption.
- $\mathsf{Dec}(sk_i, \mathbf{u}_i, \mathbf{ct}) \to m$: Takes the secret key sk_i, the update information $\mathbf{u}_i$, and the ciphertext **ct** as input. It outputs either a message $m \in \{0,1\}^*$ or a special symbol: $\perp$ (indicating a syntax error) or GetUpd (indicating that the user needs to request a fresh update information).

Definition 4.2 (Security of RBE)**.** For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathrm{Sec}_{\mathcal{A}}^{RBE}(\lambda)$ played between it and the challenger $\mathcal{C}$:

(1) Initialization. $\mathcal{C}$ sets $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathcal{D} = \emptyset$, $i^* = \perp$, $t = 0$, and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, then sends the sampled crs to $\mathcal{A}$.

(2) Query Phase. $\mathcal{A}$ performs any of the following types of queries in polynomial time.

(a) **Registering a non-target identity**. $\mathcal{A}$ sends $(regnew, i, pk_i)$ that are to be registered in the system to $\mathcal{C}$. $\mathcal{C}$ checks that $i \notin \mathcal{D}$. If the check fails, the process terminates. Otherwise, it runs the registration algorithm to register (i, pk_i) into the system: $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i, \mathsf{pk}_i)$, and then adds i to $\mathcal{D}$.

(b) **Registering target identity**. $\mathcal{A}$ sends $(regnew, i, pk_i)$ for system registration to $\mathcal{C}$. $\mathcal{C}$ first checks whether $i^* = \perp$. If the check fails, the operation is terminated; otherwise, it sets $i^* := i$, samples a challenge key pair $(pk^*, sk^*) \leftarrow \mathsf{Gen}(\mathsf{crs}, i)$, and then runs the registration algorithm to update the public parameter $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i^*, pk^*)$. After that, $\mathcal{C}$ adds i^* to $\mathcal{D}$, and finally sends pk^* to $\mathcal{A}$.

(3) Challenge Phase. $\mathcal{A}$ sends i and the message pair (m_0, m_1) to the challenger $\mathcal{C}$. $\mathcal{C}$ first verifies that $i \notin \mathcal{D} \backslash \{i^*\}$; if the verification fails, the process aborts. Otherwise, it samples a bit $b \leftarrow \{0,1\}$ and computes the challenge ciphertext $\mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, i, m_b)$.

(4) Output Phase. The adversary $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$. We say that an RBE scheme is secure if and only if for every interactive PPT adversary $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, there exists a negligible function $\mathrm{negl}(\lambda)$ such that:

$$\Pr\left[\mathcal{A} \text{ wins } Sec_{\mathcal{A}}^{RBE}(\lambda)\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

5 Our RBE with Logarithmic Registration

In this section, we present our main RBE construction with $O(\log \sqrt{N})$ registration for bounded identity sets construction. As outlined in Section 3, the warm-up based a single MLT enables logarithmic registration, but requires an $O(N)$-size structured CRS. For large N, this incurs prohibitive local storage overhead and high one-time setup latency. Therefore, Our main construction amortizes crs across multiple chunks by reusing a single smaller crs across multiple chunks.

5.1 Main Construction

Let N be the maximum number of users to be registered. The chunked construction partitions the N users into $B := \lceil N/n \rceil$ chunks, each holding at most n users. Let $\ell = \lceil \log n \rceil$. If n is not a power of two, each chunk is padded with zero entries to length 2^ℓ. For user $i \in \{0, \ldots, N-1\}$, we define $\tau = \lfloor i/n \rfloor + 1$ and $\bar{i} := i \pmod n$ as its chunk index and the intra-chunk identity. $\bar{\mathbf{v}}_i = \bar{v}_{i,\ell} \bar{v}_{i,\ell-1} \cdots \bar{v}_{i,1}$ denotes the binary presentation of $\bar{i}$. KC maintains B independent commitments $\mathsf{pp} = (C^{(1)}, \ldots, C^{(B)})$, where $C^{(\tau)}$ commits to the registry state of chunk τ and all algebraic operations are confined to the $\ell-$depth MLT in the target chunk. The KC maintains the auxiliary state $\mathsf{aux} = (\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(B)})$, where $\mathcal{T}^{(\tau)}$ is the MLT associated with $C^{(\tau)}$. Fig. 2 gives the complete specific description.

The construction applies the single-tree mechanism from Section 3 only within the target chunk. In the warm-up

[1] ***Notice.*** *Similar with [25], our scheme also requires a private-coin trusted setup, that means our system must be initialized in a trusted manner, or via a multi-party computation (MPC) protocol.*

$\underline{\mathsf{Setup}(1^\lambda, n) \to \mathsf{crs}}$**:** Generate $(p, G_1, G_2, G_T, g_1, g_2, e) \leftarrow \mathcal{BG}(1^\lambda)$ and sample $s_1, \ldots, s_\ell \leftarrow \mathbb{Z}_p^*$. Set the structured crs, with $\ell = \lceil \log n \rceil$.

$$\mathsf{crs} = \left(\left\{ \{g_1^{s_k}, g_1^{1-s_k}\}_{k\in[\ell]}, \{g_1^{\prod_{j=1}^{k} \chi_{i,j}}\}_{k\in[2,\ell], i\in[0,2^k-1]} \right\}, \left\{ \{g_2^{s_k}, g_2^{s_k-1}\}_{k\in[\ell]}, \{g_2^{\prod_{j=k}^{\ell} \chi_{i,j}}\}_{k\in[2,\ell], i\in[0,2^k-1]} \right\} \right).$$

$\underline{\mathsf{Gen}(\mathsf{crs}, i) \to (sk_i, pk_i, \xi_i)}$**:** If $i \notin [0, N-1]$, output $\perp$; otherwise, sample $a_i \leftarrow \mathbb{Z}_p^*$, let $\bar{i} = i \bmod n$, and set

$$sk_i = a_i, \qquad pk_i = g_1^{a_i \chi_{\bar{i}}}, \qquad \xi_{i,t} = g_1^{a_i \prod_{j=1}^{\ell-t} \chi_{\bar{i},j}} \text{ for } t = 1, \ldots, \ell.$$

Note that $\xi_{i,\ell} = g_1^{a_i}$ and we call it membership public key, denote by Mpk_i, which is used for later verification.

$\underline{\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i, pk_i, \xi_i) \to \mathsf{pp}'}$**:**

- If $i \notin [0, N-1]$ output $\perp$; otherwise, if it is the first identity to be registered then initialize $\mathsf{pp} = (C^{(1)}, \ldots, C^{(B)}) = (1_{G_1}, \ldots, 1_{G_1})$ and $\mathsf{aux} = (\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(B)})$ as B empty versioned trees.
- Check the correctness of ξ: If the following check fails, output $\perp$ (where $\chi_{\bar{i}} = \prod_{j=1}^{\ell} \chi_{\bar{i},j}$):

$$e(pk_i, g_2) = e(\xi_{i,1}, g_2^{\chi_{\bar{i},\ell}}) = \cdots = e(\xi_{i,\ell}, g_2^{\chi_{\bar{i}}}).$$

- Update Commitment: Compute $\tau = \lfloor i/n \rfloor + 1$, update only $C^{(\tau)'} = C^{(\tau)} \cdot pk_i$, and set $\mathsf{pp}' = (C^{(1)}, \ldots, C^{(\tau)'}, \ldots, C^{(B)})$.
- Update aux: Let $\mathcal{T}_{\bar{M}-1}^{(\tau)}$ be the latest version of the target chunk. Using copy-on-write, create a new version $\mathcal{T}_{\bar{M}}^{(\tau)}$ from $\mathcal{T}_{\bar{M}-1}^{(\tau)}$.[a] Starting from $\kappa = 1$ and, update the corresponding node $\Lambda_\kappa^{(\tau)}$ in the new version by

$$\Lambda_\kappa'^{(\tau)} \leftarrow \Lambda_\kappa^{(\tau)} \cdot \xi_{i,t}^{2\bar{v}_{i,\ell-t+1}-1}.$$

Then move to the next node on the path through $\kappa \leftarrow 2\kappa + \bar{v}_{i,\ell-t+1}$. Finally, append $\mathcal{T}_{\bar{M}}^{(\tau)}$ to the history of chunk τ.

$\underline{\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, i, m) \to \mathbf{ct}}$**:** On input $m \in G_T$, random sample $r \leftarrow \mathbb{Z}_p^*$, compute $\tau = \lfloor i/n \rfloor + 1$, set $ct_0 = C^{(\tau)}$, $A_0 = e(ct_0, g_2)^r$, $C_m = e(g_1, g_2)^r \cdot m$, compute $A_t = (g_2^{s_t - \bar{v}_{i,t}})^r$ for $t = 1, \ldots, \ell$, and output $\mathbf{ct} = (ct_0, A_0, A_1, \ldots, A_\ell, C_m)$.

$\underline{\mathsf{Upd}^{[\mathsf{aux}]}(\mathsf{pp}, i) \to \mathbf{u}_i}$**:** Output the opening $\mathbf{u}_i$ by fetching the path nodes from leaf $\bar{i}$ up to the root in $\mathcal{T}^{(\tau)}$.

$\underline{\mathsf{Dec}(sk_i, \mathbf{u}_i, \mathbf{ct}) \to m}$**:** Parse $\mathbf{u}_i = (\pi_{i,1}^1, \ldots, \pi_{i,\ell}^1, \ldots, \pi_{i,1}^{\bar{M}}, \ldots, \pi_{i,\ell}^{\bar{M}})$ and check

$$e(ct_0, g_2) \stackrel{?}{=} e(g_1^{sk_i}, g_2) \cdot \prod_{t=1}^{\ell} e\left(\pi_{i,t}, g_2^{s_t - \bar{v}_{i,t}}\right). \tag{5}$$

If false, output GetUpd triggering the request of refreshing $\mathbf{u}_i$; otherwise output $m = C_m / (A_0 / \prod_{t=1}^{\ell} e(\pi_{i,t}, A_t))^{sk_i^{-1}}$.

[a] When a registration is performed in chunk τ, KC constructs a new version $\mathcal{T}_{\bar{M}}^{(\tau)}$ from $\mathcal{T}_{\bar{M}-1}^{(\tau)}$. Only the nodes on the $\bar{i}$'s path are updated; all other nodes are shared with the previous version. The new version is appended to the history of chunk τ, and the tree histories of the remaining chunks are left unchanged.

Figure 2: Our RBE with Logarithmic Registration

based a single MLT, we write the opening of identity i as $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$, where each $\pi_{i,t}$ denotes the node at level t on the root-to-leaf path associated with i. In the main construction, the opening of i in chunk τ consist of the nodes along the path associated with $\bar{i}$ in the tree $\mathcal{T}^{(\tau)}$. KC stores the nodes of $\mathcal{T}^{(\tau)}$ in an array of a binary tree and denotes the node at index κ by $\Lambda_\kappa^{(\tau)}$. Therefore, the components of π_i are the nodes encountered when traversing the path determined by $\bar{i}$ in $\mathcal{T}^{(\tau)}$. Specifically, the traversal is from the roots to to the leaf. At level t, the current node $\Lambda_\kappa^{(\tau)}$ is exactly the t-th component of the opening:

$$\pi_{i,t} = \Lambda_\kappa^{(\tau)}. \tag{6}$$

The node index of traversal follows the corresponding bit of

the binary representation of $\bar{i}$, as follows:

$$\kappa \leftarrow 2\kappa + \bar{v}_{i,\ell-t+1}.$$

At level t the KC updates the current node as

$$\Lambda_{\kappa}^{\prime(\tau)} = \begin{cases} \Lambda_{\kappa}^{(\tau)} \cdot \xi_{i,t}, & \bar{v}_{i,\ell-t+1} = 1, \\ \Lambda_{\kappa}^{(\tau)} \cdot \xi_{i,t}^{-1}, & \bar{v}_{i,\ell-t+1} = 0. \end{cases} \tag{7}$$

where $\xi_{i,t}$ is the helping information provided by user i. Only the nodes on the path of $\bar{i}$ are updated, while all other nodes are reused from the previous version.

Correctness. Consider user i contained in the chunk τ and its commitment is $C^{(\tau)}$. Let $sk_i = a_i$ be the user's secret key and let $\mathbf{u}_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$ be a fresh update information (i.e., i's opening). By the correctness of the underlying Hyperproofs vector commitment (see Eq. (4)) , if user i registers honestly, Eq. (5) is satisfied. Then we can derive: Raising both sides of Eq. (5) to the power r, where r is the randomness sampled by Enc. By bilinearity, we obtain

$$\begin{aligned} e(ct_0, g_2)^r &= e(g_1^{sk_i}, g_2)^r \cdot \prod_{j=1}^{\ell} e\left(\pi_{i,j}, g_2^{s_j - v_{i,j}}\right)^r \\ &= e(g_1, g_2)^{sk_i \cdot r} \cdot \prod_{j=1}^{\ell} e\left(\pi_{i,j}, g_2^{(s_j - v_{i,j})r}\right) \\ &= e(g_1, g_2)^{sk_i \cdot r} \cdot \prod_{j=1}^{\ell} e(\pi_{i,j}, A_j). \end{aligned}$$

Recall the Enc, $A_0 = e(ct_0, g_2)^r$, $A_j = g_2^{(s_j - v_{i,j})r}$, and $C_m = e(g_1, g_2)^r \cdot m$. Substituting these definitions yields:

$$\begin{aligned} & A_0 = e(g_1, g_2)^{a_i r} \cdot \prod_{j=1}^{\ell} e(\pi_{i,j}, A_j) \\ \Leftrightarrow \quad & e(g_1, g_2)^{a_i \cdot r} = \frac{A_0}{\prod_{j=1}^{\ell} e(\pi_{i,j}, A_j)} \\ \Rightarrow \quad & m = \frac{C_m}{(e(g_1, g_2)^{a_i \cdot r})^{a_i^{-1}}} \end{aligned}$$

Efficiency. This section analyzes the asymptotic complexity of the computational overhead, storage overhead, and update frequency of our proposed RBE construction. we adopt the balanced setting $n = B = \sqrt{N}$ (In practical deployment, n and B can be tuned to meet specific system requirements and accommodate diverse application scenarios.). Under this setting, the main construction achieve the computational overhead $O(\log \sqrt{N})$ in registration and key generation, and has $O(\sqrt{N})$-size crs and public parameters. Tab. 1 and 2 summarize the computational and storage complexities, respectively, and compare them with prior works. Our warm-up is included as a reference in both the asymptotic comparison and the experimental evaluation to illustrate the effect of chunking in our main construction. Except for our constructions, the complexity bounds are taken from Glaeser et al. [25]. A more detailed analysis is in Appendix B. Compared with the state-of-the-art pairing-based construction [25], our main construction reduces KC-side registration from $O(\sqrt{N})$ to $O(\log \sqrt{N})$, while also reducing key generation and auxiliary storage. These improvements come at the cost of logarithmic-size ciphertexts and higher update costs. Nevertheless, our evaluation in Section 7 shows that these overheads remain practical for a user capacity of up to $N = 10^7$.The warm-up is included as a reference in both the asymptotic comparison and the experimental evaluation to illustrate the effect of chunking.

Security. The construction is secure under the decisional (R, S, T, f)-Diffie–Hellman assumption. The detailed reduction and generic-group analysis appear in Appendix C.

Theorem 2 (Security)**.** If the decisional (R, S, T, f)-Diffie–Hellman assumption holds. Then the RBE construction is secure in the sense of Definition 4.2.

6 Verifiability

In this section, we extend our RBE construction with verifiability. Verifiable RBE introduced by constructions in [25, 26] allows one to verify that a specified public key pk_i is registered under identity i, or a non-membership proof certifying that identity i is currently unregistered. We slightly adapt the syntax in [26] to reflect the structure of our construction. Unlike the construction in [25], which directly verify the registration of i's public key pk_i, we performs membership verification indirectly through the membership public key $Mpk_i = g_1^{a_i}$. It checks the one-to-one correspondence between pk_i and Mpk_i (For a fixed identity i, Mpk_i uniquely determines the corresponding RBE public key pk_i, given the selector $\chi_i(\mathbf{s}) \neq 0$), and uses an MLT opening to verify the value represented by Mpk_i at position i. Together, these checks certify the registration of pk_i.

Definition 6.1 (Verifiable RBE)**.** A key feature of our RBE construction is the ability to generate sound membership and non-membership proofs. We extend the base scheme with four additional deterministic algorithms $(\mathsf{MProve}, \mathsf{MVerify}, \mathsf{NMProve}, \mathsf{NMVerify})$:

- $\mathsf{MProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i, pk_i, \mathsf{Mpk}_i) \to \pi_i$: Takes as input the common reference string crs, the public parameter pp, an identity i, its RBE public key pk_i, and its membership public key Mpk_i. Outputs a proof π_i certifying that pk_i is registered under identity i in pp, with Mpk_i as its corresponding membership public key.
- $\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, i, pk_i, \mathsf{Mpk}_i, \pi_i) \to \{0, 1\}$: Takes as input crs, pp, identity i, public key pk_i, membership public key Mpk_i, and a proof π_i. Outputs 1 if pk_i corresponds to Mpk_i for identity i and π_i validly certifies the registration of pk_i under i; otherwise, it outputs 0.

Table 1: Comparison of Asymptotic Computational Complexities. For the main construction, let $B = n = \lceil \sqrt{N} \rceil$, where N is the maximum user capacity, M is the total number of registered users, and $\bar{M}$ is the number of registered users in the target chunk.

	Setup	Gen	Reg	Enc	Upd	Dec	#Upd	Prove	Verify
Garg [20, §4]	1	1	$\log M$	$\log M$	$\log M$	$\log M$	$\log M$	–	–
Cong [12], Garg [21]	1	1	$\log^2 M$	$\log^2 M$	$\log M$	$\log^2 M$	$\log M$	–	–
Goyal [26]	1	1	$\log^2 M$	$\log^2 M$	$\log M$	$\log M$	$\log M$	$\log^2 M$	$\log^2 M$
Glaeser [25, §4]	$\sqrt{N}$	$\sqrt{N}$	$\sqrt{N}$	1	1	1	$\sqrt{N}$	1	1
Glaeser [25, §5.1]	$\sqrt{N}$	$\sqrt{N}$	$\sqrt{N}\log N$	$\log N$	$\log N$	$\log N$	$\log N$	$\log N$	$\log N$
Warm-up (Single-Tree)	N	$\log N$	$\log N$	$\log N$	$M \log N$	$\log N$	N	$\log N$	$\log N$
Our Main Construction	$\sqrt{N}$	$\log \sqrt{N}$	$\log \sqrt{N}$	$\log \sqrt{N}$	$\bar{M} \log \sqrt{N}$	$\log \sqrt{N}$	$\sqrt{N}$	$\log \sqrt{N}$	$\log \sqrt{N}$

Table 2: Comparison of Asymptotic Storage Complexities

	$\lvert\mathsf{crs}\rvert$	$\lvert\mathsf{pp}\rvert$	$\lvert\mathsf{aux}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{u}\rvert$	$\lvert\pi\rvert$
Garg [20, §4], [21]	–	$\log M$	M	1	1	$\log M$	$\log M$	–
Cong [12]	1	$\log M$	M	1	1	$\log M$	$\log M$	–
Goyal [26]	1	$\log M$	M	1	1	$\log M$	$\log M$	$\log^2 M$
Glaeser [25, §4]	$\sqrt{N}$	$\sqrt{N}$	$M\sqrt{N}$	1	1	1	$\sqrt{N}$	1
Glaeser [25, §5.1]	$\sqrt{N}$	$\sqrt{N}\log N$	$N \log N$	1	1	$\log N$	$\log N$	$\log N$
Warm-up (Single-Tree)	N	1	$M \log N$	1	1	$\log N$	$N \log N$	$\log N$
Our Main Construction	$\sqrt{N}$	$\sqrt{N}$	$M \log \sqrt{N}$	1	1	$\log \sqrt{N}$	$\sqrt{N} \log \sqrt{N}$	$\log \sqrt{N}$

- $\mathsf{NMProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i) \to \pi_i^{NM}$: Takes as input crs, pp, and an identity i, and outputs a non-membership proof π_i certifying that i is not registered in pp.
- $\mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, i, \pi_i^{NM}) \to \{0,1\}$: Takes as input crs, pp, identity i, and a proof π_i, and outputs a bit $\{0,1\}$ denoting whether the proof is accepted or rejected.

The definition of completeness can be found in [26]. For efficiency, we achieve both the proof size of π and the running time of the verification algorithm to be sublinear in N. The definition of non-membership soundness follows that in [26]. For brevity, we omit its definition and only provide the definition of membership soundness below.

Definition 6.2 (Membership Soundness)**.** If a verifiable RBE satisfies membership soundness, then for any PPT adversary $\mathcal{A}$, its probability of winning the following experiment is negligible close to 1/2:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$
- $\mathsf{Gen}(\mathsf{crs}) \to (\mathsf{sk}, \mathsf{pk}, \mathsf{Mpk})$
- $b \to \{0,1\}$
- $\mathcal{A}(\mathsf{crs}, \mathsf{pk}, \mathsf{Mpk}) \to (\mathsf{pp}, i, \pi, m_0, m_1)$
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, i, m_b) \to \mathbf{ct}$
- The adversary wins if:

$$\mathcal{A}(\mathbf{ct}) = b \quad \text{and} \quad \mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, i, \mathsf{pk}, \mathsf{Mpk}, \pi) = 1$$

Algorithms for Verifiability. In fact, the opening π_i is the membership proof. The specific algorithm is illustrated as follows:

$\underline{\mathsf{MProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i, pk_i, Mpk_i) \to \pi_i}$**:** Parse aux to find the target tree $\mathcal{T}^{(\tau)}$. Output the lasted all nodes $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$ on the path from leaf to root associated with $\bar{i}$.

$\underline{\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, i, pk_i, Mpk_i, \pi_i) \to 1/0}$**:** First, verify that

$$e(pk_i, g_2) = e(Mpk_i, g_2^{\chi_{\bar{i}}}). \tag{8}$$

If it fails, output 0. Otherwise, extract the corresponding chunk commitment $C^{(\tau)}$, and output 1 if the following condition holds; otherwise, output 0:

$$e(C^{(\tau)}/Mpk_i, g_2) \stackrel{?}{=} \prod_{j=1}^{\ell} e(\pi_{i,j}, g_2^{s_j - \bar{v}_{i,j}}). \tag{9}$$

$\underline{\mathsf{NMProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, i) \to \pi_i}$**:** (Same as MProve)

$\underline{\mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, i, \pi_i) \to 1/0}$**:** Extract the corresponding commitment $C^{(\tau)}$. Output 1 if the following holds, else 0:

$$e(C^{(\tau)}, g_2) \stackrel{?}{=} \prod_{j=1}^{\ell} e(\pi_{i,j}, g_2^{s_j - \bar{v}_{i,j}}).$$

Batch Verification. In practice, a verifier may need to check the registration status of many identities at once. Verifying each identity separately gives linear proof size and verification cost in the batch size. We therefore extend our RBE

with batch verification, supporting efficient membership and non-membership verification both within a single chunk and across multiple chunks. We first focus on batch membership verification for β identities in a single chunk. Without loss of generality, assume it is the first β identities in first chunk, i.e., $i = 0, \cdots, \beta-1$. The membership verification in our construction consists of two checks: i) verify the one-to-one correspondence between pk_i and Mpk_i, ii) ensuring Mpk_i at position i by an MLT opening . Therefore, We aggregate these two types of checks separately.

i) To verify the one-to-one correspondence between multiple Mpk_i and pk_i, we apply the Schwartz-Zippel lemma [39] to construct a random linear combination for Eq.(8):

$$e(\prod_{i=0}^{\beta-1} pk_i^{\rho_i}, g_2) = \prod_{i=0}^{\beta-1} e(Mpk_i, g_2^{\chi_i})^{\rho_i} \tag{10}$$

where $\rho_0, \ldots, \rho_{\beta-1} \in \mathbb{Z}_p$ are sampled randomly.

ii) For all identities $i \in [0, \beta-1]$ and its binary representation $v_i = v_{i,\ell} v_{i,\ell-1} \cdots v_{i,1}$, we batch verify each Mpk_i at position i. Each membership proof π_i satisfies Eq.(9). We aggregate these proofs using IPA aggregation of Hyperproofs [41], without modifying its proof generation and subsequent IPA verification. Our only modification is an optimization of the verifier preprocessing.

In original preprocessing of $\mathcal{V}_{\mathsf{BATCH}}$ of Hyperproofs, computing $C_2 = \langle \mathbf{w}, \mathbf{F}' \rangle$ involves $\beta\ell$ pairings $e(w_{i,k}, F_{i,k}^{r_i})$. In our construction, $F_{i,k} = g_2^{s_k - v_{i,k}}$ is either $g_2^{s_k}$ or $g_2^{s_k-1}$ at each level k. We restructure the computation of C_2 by exploiting this structure and the bilinearity of the pairing, namely, $e\left(w_{i,k}, F_{i,k}^{r_i}\right) = e\left(w_{i,k}^{r_i}, F_{i,k}\right)$. Instead of computing $\mathbf{F}'$, we compute $w'_{i+1,k} = w_{i+1,k}^{r_i}$. For each bit position $k \in [\ell]$, we aggregate the $w'_{i+1,k}$ values associated with the queried identities according to the value of their k-th identity bit. Specifically, for $b \in \{0,1\}$, we define

$$W_{k,b} = \prod_{\substack{0 \le i \le \beta-1 \\ v_{i,k}=b}} w_{i+1,k}^{r_{i+1}}$$

Finally, we pair these aggregated elements with their corresponding elements in $\mathbb{G}_2$. The result is identical to the original computation of C_2, and therefore produces exactly the same preprocessing output $\mathbf{C}$, and subsequent verification steps are then completed using IPA in [41]. The complete optimized algorithm is presented in Fig. 3. This optimization reduces the number of pairing operations from $O(\beta\ell)$ to $O(\ell)$. Although it introduces additional group multiplications, pairing operations are substantially more expensive and typically dominate the verification cost. Consequently, this optimization significantly improves verification efficiency.

Extension. The above two-stage batch verification also extends to identities from different chunks. First, the one-to-one correspondence relations between pk_i and Mpk_i are aggregated using the same random linear combination by Schwartz-Zippel lemma. Then, The IPA-based aggregation of Hyperproofs supports cross-aggregation of openings with respect to different commitments and therefore applies directly to membership verification across multiple chunks, with each opening verified against its corresponding chunk commitment. Therefore, the same batch verification procedure supports both single-chunk and cross-chunk membership verification. Similarly, batch non-membership verification can be performed using the same aggregation approach as membership verification.

$\mathcal{P}_{\mathsf{BATCH}}(PK, (\mathbf{A}_j, \mathbf{F}_j))_{j=1}^{\beta}) \to \pi$:

1. Let $\mathbf{A} = [\mathbf{A_1} || \mathbf{A_2} || \cdots || \mathbf{A}_\beta]$ and $\mathbf{F} = [\mathbf{F_1} || \mathbf{F_2} || \cdots || \mathbf{F}_\beta]$
2. Let $C_1 = \langle \mathbf{A}, \eta \rangle$
3. Let $r_j = H(C_1, \mathbf{F}, j) \in \mathbb{Z}_p$ for $j = 1, \ldots, \beta$
4. Let $\mathbf{F}' = [\mathbf{F}_1^{r_1} || \mathbf{F}_2^{r_2} || \cdots || \mathbf{F}_\beta^{r_\beta}]$
5. Let $\pi^* = \mathcal{P}_{IPA}(\mathbf{ck}, \mathbf{A}, \mathbf{F}')$ and return $\pi = (C_1, \pi^*)$.

Optimized $\mathcal{V}^*_{\mathsf{BATCH}}(VK, (\mathbf{F}_j, Z_j)_{j=1}^{\beta}, \pi) \to \{0,1\}$:

1. Parse the proof $\pi = (C_1, \pi^*)$.
2. Let $r_j = H(C_1, \mathbf{F}, j) \in \mathbb{Z}_p$ for $j = 1, \ldots, \beta$.
3. Compute $Z' = \prod_{j=1}^{\beta} Z_j^{r_j}$.
4. Parse the verification state vector $\mathbf{w} = [\mathbf{w}_1 \| \mathbf{w}_2 \| \ldots \| \mathbf{w}_\beta] \in \mathbb{G}_1^{\beta\ell}$, where each sub-vector is defined as $\mathbf{w}_j = (w_{j,1}, w_{j,2}, \ldots, w_{j,\ell})$.
5. For each level $k \in [\ell]$, aggregate the $\mathbb{G}_1$ elements based on the identity bit $v_{i,k}$:
$$W_{k,0} = \prod_{\substack{1 \le j \le \beta \\ v_{i,k}=0}} w_{j,k}^{r_j} \quad \text{and} \quad W_{k,1} = \prod_{\substack{1 \le j \le \beta \\ v_{i,k}=1}} w_{j,k}^{r_j}$$
6. Compute C_2 using exactly 2ℓ pairings instead of $\beta\ell$:
$$C_2 = \prod_{k=1}^{\ell} \left(e(W_{k,0}, g_2^{s_k}) \cdot e(W_{k,1}, g_2^{s_k-1}) \right)$$
7. Let $\mathbf{C} = (C_1, C_2, Z')$ and return $\mathcal{V}_{\mathsf{IPA}}(\mathbf{ck}, \mathbf{C}, \pi^*)$.

Figure 3: The optimized preprocessing algorithm used for aggregation. Where H is a random oracle, and both η and $\mathbf{w}$ are publicly available random group element vectors

Security Analysis. We state the security theorem of verifiability algorithm and provide a proof sketch in Appendix D. Moreover, the soundness of the aggregated membership proof follows from the batch aggregation argument of Hyperproofs [41].Our optimization for the preprocessing of IPA changes neither proof generation nor the subsequent IPA verification; it only computes C_2 in the verifier preprocessing more efficiently. Consequently, any proof accepted by our optimized verifier is also accepted by the original Hyperproofs verifier, and its soundness follows directly from that of Hyperproofs.

Theorem 3 (Soundness)**.** Suppose that, for any admissible challenge identity, the Hyperproofs vector commitment is evaluation binding and the decisional (R,S,T,f)-Diffie-Hellman assumption is hard, the above verifiability satisfies membership and non-membership soundness.

7 Evaluation

In this section, we implement the warm-up construction and the main construction described in Section 3 abd Section 5 using Python, and then compare them with the existing efficient black-box RBE scheme based on pairings [25]. For varying system capacities $N \in \{10^4, 10^5, 10^6, 10^7\}$, we measure algorithm runtime and parameter storage overhead. Experimental results show that our constructions significantly reduce the overhead of the two algorithms—Gen and Reg, while maintaining encryption, decryption, and storage costs within practical bounds.

Implementation and platform. Experiments are conducted on a computer equipped with an Intel Core i7-10700 @2.9GHz and 16 GB RAM. Benchmarking is performed inside an Ubuntu 18.04 virtual machine configured with 8 GB RAM and 16 virtual cores. All schemes are implemented in Python 3.6. To ensure a fair comparison with [25], we instantiate the BLS12-381 elliptic curve via `petrelic` [24], a Python wrapper for the RELIC library [2]. A serialized group element occupies 49 B in $\mathbb{G}_1$, 97 B in $\mathbb{G}_2$, and 384 B in $\mathbb{G}_T$. In our experiment, the average latency for a scalar multiplication is 0.16 ms in $\mathbb{G}_1$ and 0.29 ms in $\mathbb{G}_2$, while a bilinear pairing operation averages 0.98 ms.

Benchmark. For the main construction, we set both the number of chunks and the capacity of a single chunk to $B = n = \lceil \sqrt{N} \rceil$. As shown in Tab.3, we measure the running time of Setup, Gen, Reg, Enc, Upd, and Dec, where all values except Setup are averaged over $\sqrt{N}$ executions. This table also reports the storage overhead of $|\mathsf{pp}|$, $|\mathsf{crs}|$, $|\mathsf{aux}|$, and $|\mathsf{ct}|$, where $|\mathsf{pp}|$, $|\mathsf{aux}|$, and $|\mathsf{ct}|$ are generated and measured under the worst-case, fully-loaded configuration. Notably, since the Setup time of the warm-up construction grows linearly with N, it cannot complete within a reasonable time when $N = 10^7$; accordingly, for this case, Tab. 3 reports only the storage overhead and omits the algorithm runtimes. we utilize Fig. 4 to provide a clearer view of the trends in the data reported in Tab. 3 for varying values of N. For time performance, as shown in Fig. 4a, the Gen and Reg algorithms in our scheme have logarithmic asymptotic complexity, and their running times increase slowly with N. In contrast, the running times of both algorithms under the two constructions in [25] increase substantially as N grows. When $N = 10^7$, our primary construction completes Gen and Reg in 1.82 ms and 20.46 ms, respectively. In comparison, the construction in [25, § 4] requires 403.26 ms and 3.046 s, respectively, whereas the construction in [25, § 5.1] requires 419.44 ms and 3.223 s. Thus, our primary construction achieves speedups of approximately 220–230× for Gen and 150–160× for Reg. These results validate our design objective: by leveraging the MLT-based update mechanism, our scheme reduces the asymptotic costs of user-key generation and registration to logarithmic time while keeping update latency practical, thereby substantially improving the overall performance of RBE.

For storage overhead, as shown in Fig. 4b, although the $|\mathsf{pp}|$ of the main construction is slightly larger than the $|\mathsf{pp}|$ of the warm-up construction, it substantially reduces the costly $|\mathsf{crs}|$ overhead required by the latter. Specifically, when $N = 10^7$, the $|\mathsf{crs}|$ size of the warm-up construction increases to 5.33 GB, whereas that of the main construction is only 1.3 MB. Additionally, compared with the two schemes in [25], the $|\mathsf{pp}|$ of our main construction is 184 KB, which is comparable to the size of the $|\mathsf{pp}|$ in [25, §4] and significantly smaller than the 2.6 MB required by the scheme in [25, §5.1]. Furthermore, the auxiliary storage $|\mathsf{aux}|$ required by our main construction is also strictly lower than that of both baseline schemes. In particular, when $N = 10^7$, our scheme requires 11.57 GB of $|\mathsf{aux}|$, whereas the schemes in [25, §4] and [25, §5.1] require 1.9 TB and 16.3 GB, respectively.

For the encryption and decryption algorithms, the ciphertext contains ℓ elements from the group $\mathbb{G}_2$ to encode the recipient's identity path. Consequently, the running times of Enc and Dec grow logarithmically rather than remaining constant. When $N = 10^7$, encryption and decryption take 6.75 ms and 24.85 ms, respectively. Although slightly slower than the scheme in [25, §4], the proposed scheme still satisfies the performance requirements of practical encryption scenarios. Meanwhile, our decryption latency is significantly lower than that of the efficient-update variant presented in [25, §5.1]. Although the latency of the update operation is higher than that of the two benchmark schemes in [25], a single update can still be completed within 0.2 seconds even when $N = 10^7$.

IPA-based batch verification. The verifiable extension in Section 6 leverages IPA-based batch proofs to aggregate multiple membership and non-membership verification queries. In the original IPA verifier preprocessing step, validating a batch of β identities (each of bit-length ℓ, matching the MLT depth) requires $\beta\ell$ pairing operations to compute C_2. Our optimized preprocessing exploits the binary structure of user identities: each bit position of every identity maps to only two

Table 3: Performance Comparison

N		Setup(s)	Gen(s)	Reg(s)	Enc(s)	Upd(s)	Dec(s)	|pp|	|crs|	|aux|	|ct|
10K	[25, §4]	**0.09925**	0.01432	0.11597	**0.00324**	0.00277	**0.00500**	16KB	**40 KB**	57.8MB	**914B**
	[25, §5.1]	0.12977	0.01467	0.1217	0.02094	**0.00072**	0.20348	106KB	**40 KB**	9.2MB	6.07KB
	Warm-up	14.23350	0.00219	0.02306	0.00696	0.02242	0.02968	**8KB**	5.21MB	13.04MB	2.12KB
	Main construction	0.13200	**0.00110**	**0.01173**	0.00469	0.00486	0.01455	16KB	52KB	**7.11MB**	1.46KB
100K	[25, §4]	**0.30628**	0.04472	0.35024	**0.00328**	0.00836	**0.00492**	28KB	**112 KB**	1.9GB	**914B**
	[25, §5.1]	0.47035	0.04313	0.34408	0.02534	**0.00081**	0.31248	258KB	**112 KB**	119.6MB	7.59KB
	Warm-up	118.12993	0.00271	0.04484	0.00827	0.09218	0.03587	**8KB**	41.68MB	151.2MB	2.42KB
	Main construction	0.46714	**0.00140**	**0.01464**	0.00535	0.01700	0.01917	28KB	176KB	**95.31MB**	1.65KB
1M	[25, §4]	**0.87403**	0.12110	0.93119	**0.00323**	0.02339	**0.00431**	64KB	328 KB	59.3GB	**914B**
	[25, §5.1]	1.84997	0.12447	0.94715	0.02551	**0.00095**	0.34808	716 KB	328 KB	1.3 GB	9.11 KB
	Warm-up	899.52939	0.00271	0.13025	0.00828	0.34694	0.03579	**8KB**	339.35MB	1.72GB	2.69KB
	Main construction	0.87739	**0.00161**	**0.01852**	0.00647	0.05191	0.02301	64KB	**334KB**	**970.35MB**	1.75KB
10M	[25, §4]	**3.37454**	0.40326	3.04609	**0.00310**	0.07417	**0.00460**	184KB	**1.0 MB**	1.9 TB	**914 B**
	[25, §5.1]	17.79602	0.41944	3.22273	0.03286	**0.00115**	0.54624	2.6 MB	**1.0 MB**	16.3 GB	10.63 KB
	Warm-up	———	———	———	———	———	———	**8KB**	5.33GB	21.48GB	3.07KB
	Main construction	3.68209	**0.00182**	**0.02046**	0.00675	0.18011	0.02485	184KB	1.3MB	**11.57GB**	1.93KB

Table 4: IPA verification performance with and without the proposed preprocessing optimization. Here, b denotes the batch size and ℓ denotes the bit length of each identity.

b	ℓ	**Preprocessing**			**Overall IPA verification**		
		Naive (ms)	**Optimized (ms)**	**Speedup**	**Naive (ms)**	**Optimized (ms)**	**Speedup**
10	12	132.46	39.64	3.34×	153.87	60.59	2.54×
20	12	267.60	58.78	4.55×	300.03	82.73	3.63×
50	12	645.52	103.23	6.25×	675.07	131.83	5.12×
100	12	1301.72	185.81	7.01×	1333.27	219.39	6.08×
1000	12	13214.91	1658.00	7.97×	13256.29	1697.89	7.81×
2000	12	25294.31	3437.13	7.36×	25336.31	3485.35	7.27×
3000	12	37777.73	4748.28	7.97×	37822.31	4792.55	7.89×

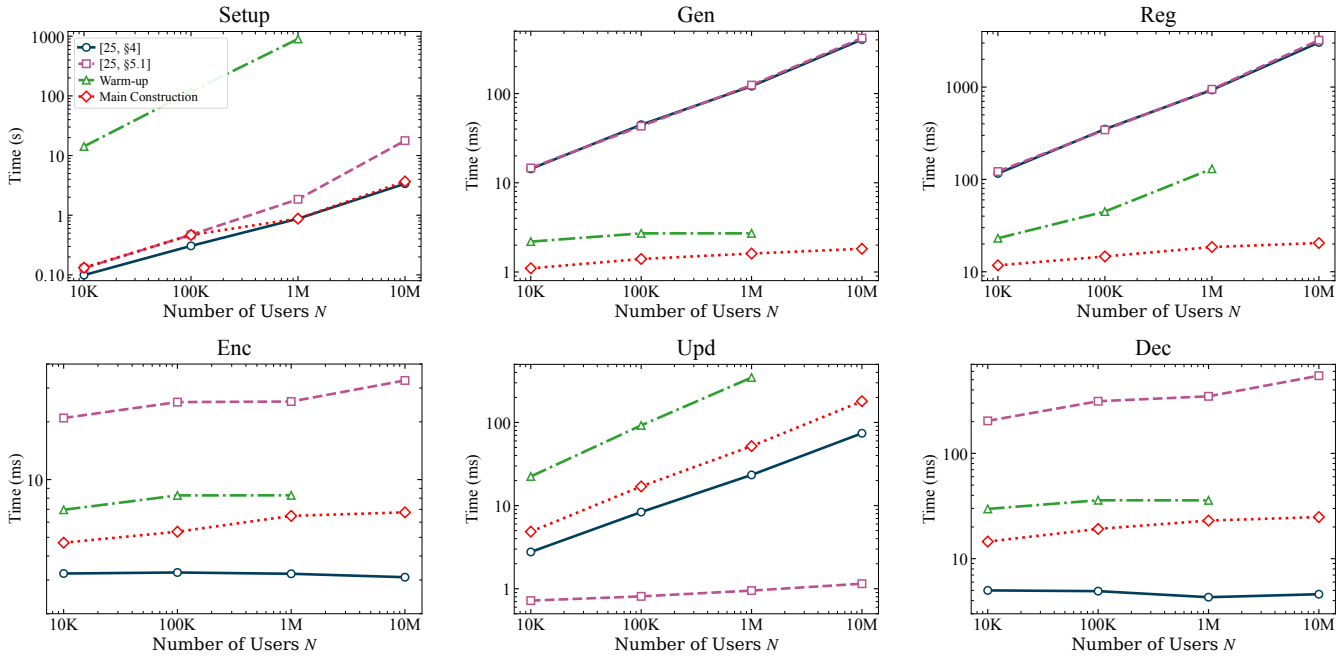
possible values in $\mathbb{G}_2$. This reduces the pairing count drastically to 2ℓ, independent of the batch size β. Tab. 4 quantifies this performance improvement for the case $\ell = 12$.

As the batch size β grows, the speedup ratio of our optimized scheme exhibits an increasing trend, before eventually reaching a plateau. For $\beta = 10$, preprocessing time falls from 132.46 ms to 39.64 ms, and the total IPA verification time drops from 153.87 ms to 60.59 ms, yielding a 2.54× speedup. For $\beta = 1000$, preprocessing time falls from 13.21 s to 1.66 s, and total verification time decreases from 13.26 s to 1.70 s, achieving a 7.81× speedup. However, as β scales further, the speedup ratio ceases to increase, slightly declines, and eventually stabilizes. This phenomenon occurs because our optimization specifically targets pairing operations, while the number of group exponentiations remains $O(\beta\ell)$. At this point, the cost of exponentiation has become the dominant component of the runtime, so further increasing β does not result in any additional speedup. These results confirm that our optimized IPA preprocessing process delivers tangible performance improvements for verifier in batch verification scenarios.

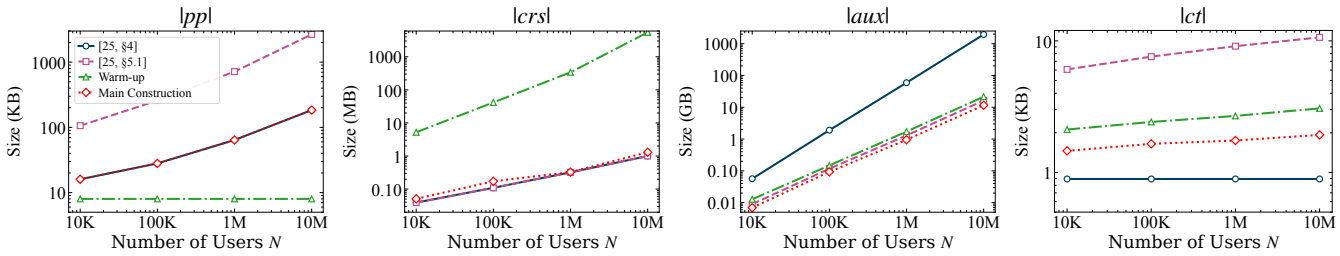
Summary. Our experimental results validate the asymptotic analysis in Section 5. Compared to prior pairing-based black-box RBE schemes [25], the warm-up construction improves registration efficiency; however, its $|\mathsf{crs}|$ scale renders it infeasible for deployments with a large user scale N. The main construction resolves this bottleneck: it compresses crs, pp, ct, and aux to sizes viable for systems supporting tens of millions of users, while delivering 1–2 orders of magnitude speedup in key generation and registration latencies over the baselines. Accordingly, the main construction achieves a favorable practical trade-off among registration throughput, public parameter size and storage overhead.

8 Conclusions

In this work, we present a black-box pairing-based RBE construction with logarithmic registration. By maintaining the registry with MLTs, each registration updates only a root-to-leaf path. Under the balanced setting $n = B = \sqrt{N}$, the



(a) Time performance



(b) Storage overhead

Figure 4: Performance comparison for different user capacities N. Here, N denotes the maximum number of supported users capacity. (a) Running time of Setup, Gen, Reg, Enc, Upd, and Dec. (b) Storage overhead of pp, crs, aux, and ct. The proposed warm-up and main constructions are compared with the schemes of Glaeser et al. [25]; for the main construction, $B = n = \lceil \sqrt{N} \rceil$.

construction achieves $O(\log \sqrt{N})$ registration and key generation while retaining $O(\sqrt{N})$-size CRS and public parameters. We further support membership/non-membership verification and efficient batch verification through an optimized IPA-based aggregation mechanism. Our evaluation confirms that the resulting logarithmic registration and batch verification are practical at large scales.

Our construction currently supports a bounded identity space, where the maximum capacity is fixed during setup, and relies on a structured CRS. Moreover, logarithmic registration comes at the cost of logarithmic-size ciphertexts and high update latency. Addressing these limitations while preserving logarithmic registration efficiency remains an important direction for future work. Despite these limitations, our results demonstrate that logarithmic registration can be achieved in a concretely efficient pairing-based RBE construction while maintaining practical performance at large system scales. This work narrows the gap between the goal of polylogarithmic registration and the square-root registration complexity of prior efficient pairing-based constructions.

References

[1] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pages 452–473. Springer, 2003.

[2] Diego F Aranha. Relic is an efficient library for cryptography. *http://code.google.com/p/relic-toolkit/*, 2020.

[3] Kyoichi Asano, Nuttapong Attrapadung, Keisuke Hara, Keitaro Hashimoto, and Yohei Watanabe. Key revocation in registered attribute-based encryption. In *IACR*

International Conference on Public-Key Cryptography, pages 102–133. Springer, 2025.

[4] Nuttapong Attrapadung and Junichi Tomida. A modular approach to registered abe for unbounded predicates. In *Annual international cryptology conference*, pages 280–316. Springer, 2024.

[5] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 440–456. Springer, 2005.

[6] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[7] Xavier Boyen. The uber-assumption family: A unified complexity framework for bilinear groups. In *International Conference on Pairing-Based Cryptography*, pages 39–56. Springer, 2008.

[8] Pedro Branco, Russell WF Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy KY Woo. Traitor tracing without trusted authority from registered functional encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 33–66. Springer, 2024.

[9] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In *International conference on the theory and application of cryptology and information security*, pages 65–97. Springer, 2021.

[10] Jeffrey Champion, Yao-Ching Hsieh, and David J Wu. Registered abe and adaptively-secure broadcast encryption from succinct lwe. In *Annual International Cryptology Conference*, pages 3–34. Springer, 2025.

[11] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA international conference on cryptography and coding*, pages 360–363. Springer, 2001.

[12] Kelong Cong, Karim Eldefrawy, and Nigel P Smart. Optimizing registration based encryption. In *IMA International Conference on Cryptography and Coding*, pages 129–157. Springer, 2021.

[13] Pratish Datta, Tapas Pal, and Shota Yamada. Registered fe beyond predicates:(attribute-based) linear functions and more. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 65–104. Springer, 2024.

[14] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Annual international cryptology conference*, pages 537–569. Springer, 2017.

[15] Nico Döttling, Dimitris Kolonelos, Russell WF Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 417–446. Springer, 2023.

[16] Jesko Dujmovic, Giulio Malavolta, and Wei Qi. Registration-based encryption in the plain model. In *IACR International Conference on Public-Key Cryptography*, pages 194–226. Springer, 2025.

[17] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: registration-based encryption and key-value map commitments for large spaces. In *International conference on the theory and application of cryptology and information security*, pages 166–200. Springer, 2023.

[18] Cody Freitag, Brent Waters, and David J Wu. How to use (plain) witness encryption: registered abe, flexible broadcast, and more. In *Annual international cryptology conference*, pages 498–531. Springer, 2023.

[19] Rachit Garg, George Lu, Brent Waters, and David J Wu. Reducing the crs size in registered abe systems. In *Annual international cryptology conference*, pages 143–177. Springer, 2024.

[20] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: removing private-key generator from ibe. In *Theory of Cryptography Conference*, pages 689–718. Springer, 2018.

[21] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In *IACR international workshop on public key cryptography*, pages 63–93. Springer, 2019.

[22] Craig Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2006*, pages 445–464, 2006. doi:10.1007/11761679_27.

[23] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[24] Wouter Lueks Laurent Girod. petrelic is a python wrapper around relic. https://github.com/spring-epfl/petrelic, 2022.

[25] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1065–1079, 2023.

[26] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In *Annual International Cryptology Conference*, pages 621–651. Springer, 2020.

[27] Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Annual International Cryptology Conference*, pages 430–447. Springer, 2007.

[28] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436, 2008.

[29] Mohammad Hajiabadi, Mohammad Mahmoody, Wei Qi, and Sara Sarfaraz. Lower bounds on assumptions behind registration-based encryption. In *Theory of cryptography conference*, pages 306–334. Springer, 2023.

[30] Susan Hohenberger, George Lu, Brent Waters, and David J Wu. Registered attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 511–542. Springer, 2023.

[31] Aniket Kate and Ian Goldberg. Distributed private-key generators for identity-based cryptography. In *International Conference on Security and Cryptography for Networks*, pages 436–453. Springer, 2010.

[32] Michael Klooß, Russell WF Lai, Jan Niklas Siemer, and Monisha Swarnakar. Scalable registration-based encryption from lattices. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 2520–2538. IEEE, 2026.

[33] Jiguo Li, Shaobo Chen, Yang Lu, Jianting Ning, Jian Shen, and Yichen Zhang. Revocable registered attribute-based encryption with user deregistration. *IEEE Internet of Things Journal*, 12(15):31526–31535, 2025.

[34] George Lu, Brent Waters, and David J Wu. Multi-authority registered attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2025.

[35] Mohammad Mahmoody, Wei Qi, and Ahmadreza Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. In *Theory of cryptography conference*, pages 559–587. Springer, 2022.

[36] Shayeef Murshid, Ramprasad Sarkar, and Mriganka Mandal. Registered attribute-based encryption with publicly verifiable certified deletion, everlasting security, and more. *arXiv preprint arXiv:2603.07646*, 2026.

[37] Tapas Pal, Robert Schädlich, and Erkan Tairi. Registered functional encryption for pseudorandom functionalities from lattices: Registered abe for unbounded depth circuits and turing machines, and more. *Cryptology ePrint Archive*, 2025.

[38] Kenneth G Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In *International Conference on Pairing-Based Cryptography*, pages 354–375. Springer, 2008.

[39] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[40] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[41] Shravan Srinivasan, Alexander Chepurnoy, Charalampos Papamanthou, Alin Tomescu, and Yupeng Zhang. Hyperproofs: Aggregating and maintaining proofs in vector commitments. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3001–3018, 2022.

[42] Toi Tomita. Towards more efficient registration-based encryption from lwe. In *Applied Cryptography and Network Security: 24th International Conference, ACNS 2026, Stony Brook, NY, USA, June 22–25, 2026, Proceedings, Part I*, page 75. Springer, 2026.

[43] Brent Waters. Efficient identity-based encryption without random oracles. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 114–127. Springer, 2005.

[44] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Annual international cryptology conference*, pages 619–636. Springer, 2009.

[45] Hoeteck Wee and David J Wu. Unbounded distributed broadcast encryption and registered abe from succinct lwe. In *Annual International Cryptology Conference*, pages 204–235. Springer, 2025.

[46] Tianwei Zhang, Xiuquan Ding, Giulio Malavolta, and Nico Döttling. Fast and compact lattice-based

registration-based encryption. *Cryptology ePrint Archive*, 2026.

[47] Ziqi Zhu, Kai Zhang, Zhili Chen, Junqing Gong, and Haifeng Qian. Black-box registered abe from lattices. *Cryptology ePrint Archive*, 2025.

[48] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered abe via predicate encodings. In *International conference on the theory and application of cryptology and information security*, pages 66–97. Springer, 2023.

A Security Definition of RBE

Definition A.1 (Completeness, Compactness, and Efficiency of RBE)**.** For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathrm{Comp}_{\mathcal{A}}^{RBE}(\lambda)$, which is executed interactively by the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$:

(1) Initialization. $\mathcal{C}$ sets $\mathsf{pp}=\perp$, $\mathsf{aux}=\perp$, $\mathbf{u}=\perp$, $\mathcal{D}=\emptyset$, $i^*=\perp$, $t=0$, and $\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda)$, then sends the sampled crs to $\mathcal{A}$.

(2) Query Phase. Adversary $\mathcal{A}$ may perform any of the following types of queries a polynomial number of times. Each query is regarded as an interaction between $\mathcal{A}$ and $\mathcal{C}$.

(a) **Registering a non-target identity**. $\mathcal{A}$ sends $(regnew, i, \mathsf{pk}_i)$ that will be registered in the system to $\mathcal{C}$. $\mathcal{C}$ checks that $i\notin\mathcal{D}$. If it fails, the process terminates. Otherwise, it runs the Reg to register (i,pk_i) into the system: $\mathsf{pp}:=\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs},\mathsf{pp},i,\mathsf{pk}_i)$, and then adds i to $\mathcal{D}$. After each query, $\mathcal{C}$ updates the parameters pp, aux and $\mathcal{D}$.

(b) **Registering target identity**. $\mathcal{A}$ sends $(regnew, i, pk_i)$ for registration to $\mathcal{C}$. $\mathcal{C}$ first checks whether $i^*=\perp$. If it fails, the operation is terminated; otherwise, $\mathcal{C}$ sets $i^*:=i$, samples a challenge key pair $(pk^*,sk^*)\leftarrow\mathsf{Gen}(\mathsf{crs},i^*)$, and then runs the Reg to update the public parameter $\mathsf{pp}:=\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs},\mathsf{pp},i^*,pk^*)$. After that, it adds i^* to $\mathcal{D}$, and finally sends pk^* to $\mathcal{A}$. (In this case, the challenger stores the secret key sk^* while updating all other parameters. Note that the adversary is allowed to make at most one such query; otherwise, the challenger will terminate the operation.)

(c) **Target identity encryptions**. $\mathcal{A}$ sends the data to be encrypted $(enctgt, m)$ to $\mathcal{C}$, and $\mathcal{C}$ checks whether $i^*\neq\perp$. If the check fails, the operation is terminated; otherwise, $\mathcal{C}$ sets $t:=t+1$, $\widetilde{m}_t:=m$, and computes the ciphertext $\mathbf{ct}_t\leftarrow\mathsf{Enc}(\mathsf{crs},\mathsf{pp},i^*,\widetilde{m}_t)$. After that, it stores $(t,\widetilde{m}_t,\mathbf{ct}_t)$ and sends the ciphertext to $\mathcal{A}$.

(d) **Target identity decryptions**. $\mathcal{A}$ sends the index to be decrypted $(dectgt, j)$ to $\mathcal{C}$, and $\mathcal{C}$ verifies that $i^*\neq\perp$ and $j\in[t]$. If the verification fails, the process terminates. Otherwise, it computes $y_j=\mathsf{Dec}(\mathsf{sk}^*,\mathbf{u},\mathbf{ct}_j)$. If $y_j=\mathsf{GetUpd}$, a new updated information $\mathbf{u}:=\mathsf{Upd}^{[\mathsf{aux}]}(\mathsf{pp},i^*)$ is generated, and then recomputes $y_j=\mathsf{Dec}(\mathsf{sk}^*,\mathbf{u},\mathbf{ct}_j)$. Finally, $\mathcal{C}$ stores (j,y_j).

(3) Output Phase. If there exists an index $j\in[t]$ such that $\widetilde{m}_j\neq y_j$, then $\mathcal{A}$ is deemed to have won the game.

Let $Q\in poly$ be the upper bound on the number of queries issued by $\mathcal{A}$, and let $\tilde{N}\leftarrow|\mathcal{D}|$ denote the number of registered users up to any given moment in the above game. At any point during the execution of the game $\mathrm{Comp}_{\mathcal{A}}^{RBE}$, any adversary $\mathcal{A}$ shall satisfy the following properties:

- **Completeness.** $\Pr[\mathcal{A}\text{ wins }\mathrm{Comp}_{\mathcal{A}}^{RBE}(\lambda)]\leq\mathrm{negl}(\lambda)$.
- **Compactness.** For each query $q\in[Q]$, let pp_q represent the public parameters generated after the q-th query. The size of pp_q scales sublinearly with $\tilde{N}$. Similarly, for every index $i\in\mathcal{D}$, the size of the corresponding update term $\mathbf{u}_i$ also exhibits a sublinear growth rate in $\tilde{N}$.
- **Efficiency of registration and update.** The running time of each call to the Reg and Upd algorithms is sublinear in with $\tilde{N}$.
- **Efficiency of the number of updates.** In step 2 (d) of the game $\mathrm{Comp}_{\mathcal{A}}^{RBE}(\lambda)$, the total number of calls to the Upd for the identity i^* grows sublinearly with $\tilde{N}$.

B Complexity Analysis

In this section, we provide a detailed analysis and comparison of the storage overhead, computational overhead, and update frequency of our RBE constructions and the RBE construction proposed by Glaeser et al. [25].

Storage Overhead: The warm-up construction employs a full MLT of depth $\ell=\lceil\log N\rceil$, where the common reference string crs contains $O(N)$ group elements, while the public parameter pp consists of only one group element. To support updates, the auxiliary data aux maintains a tree version for each registration. Owing to the copy-on-write mechanism, each registration introduces only $O(\log N)$ new nodes, resulting in an aggregate storage overhead of $O(N\log N)$ group elements for aux. Moreover, the user secrets and keys (sk_i, Mpk_i, pk_i) are all of constant size, whereas the update information u_i has size $O(M\log N)$. Building upon the warm-up construction, the main construction (Fig. 2) further reduces the storage overhead through chunking. Specifically, it partitions the N users into $B=N/n$ chunks of size n and reduces the depth of each MLT to $\ell=\lceil\log n\rceil$. When $n=B=\sqrt{N}$, the crs size decreases from $O(N)$ to $O(n)=O(\sqrt{N})$, while the pp size increases to $O(B)=O(\sqrt{N})$ to accommodate the chunk commitments. Meanwhile, the auxiliary storage decreases from $O(N\log N)$ to $O(N\log n)=O(N\log\sqrt{N})$. A ciphertext in the main construction consists of $\ell+3$ group elements $(ct_0,A_0,\ldots,A_\ell,C_m)$, yielding a size of $O(\log n)=O(\log\sqrt{N})$. Although this is larger than the constant-size ciphertexts in the construction of [25], this trade-off enables users to generate their keys

entirely locally while significantly reducing the registration complexity.

Computational Overhead: Our complexity analysis mainly focuses on the main construction, where we uniformly set $n = B = \sqrt{N}$ (the corresponding complexity for the warm-up construction can be obtained by substituting N for n). In practical deployments, n and B can be adjusted according to specific system requirements and application scenarios. The initialization algorithm generates the crs once, which involves $O(n)$ group exponentiations in the main construction. During key generation, each user computes auxiliary information ξ_i of size $O(\ell) = O(\log n)$, enabling the Key Curator (KC) to update the MLT using $O(\log n)$ exponentiations without accessing the user's secret key. Registration is the core operation: the KC verifies $\ell = O(\log n)$ pairing equations, performs one group multiplication to update the chunk commitment $C^{(k)}$, and updates $\ell = O(\log n)$ nodes along the MLE path via group multiplications. Consequently, the total complexity of the registration is $O(\log n)$, significantly improving the complexity $O(\sqrt{N})$ of the construction in [25]. Encryption requires $\ell + 1$ exponentiations and one pairing operation, resulting in a complexity of $O(\log n)$; decryption similarly has a complexity of $O(\log n)$, as it requires $\ell = O(\log n)$ pairing operations to verify the opening, followed by one exponentiation.[2] An update query returns historical opening of size $O(M \log n)$ and consumes $O(M \log n)$ time.

Update Frequency: When a new user completes registration within the same chunk, only the nodes along the path from the root node to the leaf node in the MLT are modified. Therefore, in the main construction, each user triggers at most $\sqrt{N}$ updates throughout the entire system runtime, which matches the update upper bound of the basic scheme proposed in [25]. The key advantage is that, unlike the original scheme of [25], our algorithm incurs no auxiliary storage overhead of magnitude $O(\sqrt{N})$ when reaching this upper bound.

C Security Proof

We prove that our construction achieves security. The proof follows a two-step approach: First, we prove that the target polynomial f is independent of $\langle R, S, T \rangle$. Second, we show that any adversary breaking security yields an algorithm solving the decisional (R, S, T, f)-Diffie–Hellman problem of Definition 2.1, for a concrete tuple (R, S, T, f) induced by the scheme. Finally, we invoke the main theorem (Theorem. 1) to establish the hardness result of this problem in the generic group model. We denote $g_T = e(g_1, g_2)$. Throughout the proof, we treat $s_1, \ldots, s_\ell$, the user secret keys α, and the encryption randomness r as the secret variables underlying the instance. Without loss of generality, we take the first chunk as an example and $i \in [n]$, the security proofs for the other chunks are similar.

For clarity, recall Section 2.1, where $\chi_{i,k}$ is the k-th-level MLE selector for identity i: it takes the value s_k when $v_{i,k} = 1$, and $1 - s_k$ when $v_{i,k} = 0$. Its full product is $\chi_i = \prod_{k=1}^{\ell} \chi_{i,k}$.

Fix a challenge identity i with binary vector $\mathbf{v}_i = (v_{i,\ell}, \ldots, v_{i,1})$ and introduce one variable α for the secret key used if i is registered through the target-registration query. For each fixed i, the polynomial tuples used by the reduction are defined from the following algebraic instance:

- the $\mathbb{G}_1$ part of crs together with the public keys and ξ_i, whose exponents form the polynomial tuple

$$R = \Big\langle 1,\ \{s_k, 1 - s_k\}_{k \in [\ell]}, \left\{ \prod_{h=1}^{k} \chi_{u,h} \right\}_{k \in [\ell],\, u \in [0, 2^k - 1]},$$
$$\alpha\chi_i,\ \left\{ \alpha \prod_{h=1}^{\ell - k} \chi_{i,h} \right\}_{k \in [\ell]} \Big\rangle,$$

  i.e. none of these involve r;

- the $\mathbb{G}_2$ part of crs together with the ciphertext components $A_1, \ldots, A_\ell$, whose exponents form

$$S = \Big\langle 1,\ \{s_k, s_k - 1\}_{k \in [\ell]},$$
$$\left\{ \prod_{j=k}^{\ell} \chi_{u,j} \right\}_{k \in [\ell],\, u \in [0, 2^k - 1]},$$
$$\big\{ r(s_j - v_{i,j}) \big\}_{j \in [\ell]} \Big\rangle.$$

- the $\mathbb{G}_T$ exponent: $T = \langle 1, \alpha r \rangle$.

$g_T^{\alpha r}$ is only used when this registration actually takes place. Since the adversary can choose the keys for non-target registrations on its own and can compute their public keys via crs, the public keys submitted in non-target registration queries are excluded from the R.Because the masking factor of the challenge is $C_m = e(g_1, g_2)^r \cdot m = g_T^r \cdot m$, so the target polynomial is $f = r$. We refer to (R, S, T, f) as the tuple *induced* by the challenge identity i.

Theorem 4 (Independence of the target)**.** For every challenge identity i admissible in Definition 4.2, the target polynomial $f = r$ is independent of $\langle R, S, T \rangle$ in the Type-III sense.

Proof. Suppose, for contradiction, that f is dependent. By the independence definition of Section 2.3 there exist constants $\{c_{p,q}\}$ and $\{b_k\}$ such that, as a formal polynomial identity over $\mathbb{F}_p[s_1, \ldots, s_\ell, \alpha, r]$,

$$r = \sum_{p,q} c_{p,q} R_p S_q + \sum_k b_k T_k. \qquad (*)$$

[2] For Dec, we assume that the decrypter knows the correct opening for a given ciphertext, which is naturally the case when the user's update information is current and the ciphertext is recent, as in the efficiency analysis of Glaeser et al. [25]. Otherwise, its worse-case complexity is bounded by $O(N \log N)$ (warm-up construction) or by $O(\sqrt{N} \log \sqrt{N})$ (main construction)

Every polynomial in R is free of r. In S, the only polynomials containing r are the terms $r(s_j - v_{i,j})$. The only term in T containing r is αr. Therefore, after collecting the coefficient of r in $(*)$, we obtain

$$1 = \sum_{p,j} c'_{p,j} R_p(s_j - v_{i,j}) + b\alpha \qquad (**)$$

as an identity in $\mathbb{F}_p[s_1, \ldots, s_\ell, \alpha]$. The terms not containing r in $(*)$ must cancel respectively. Now evaluate both sides of $(**)$ at the point that sets $s_k = v_{i,k}$ for all $k \in [\ell]$ and $\alpha = 0$. On the right-hand side, each factor $(s_j - v_{i,j})$ vanishes and the term $b\alpha$ vanishes as well, so the whole expression evaluates to 0; on the left-hand side it evaluates to 1. This yields $1 = 0$ in $\mathbb{F}_p$, a contradiction. Hence no relation of the form $(*)$ exists and $f = r$ is independent.

Two remarks make the mechanism explicit. The point $s_k = v_{i,k}$ is precisely the selector assignment of the challenge identity, at which every ciphertext component A_j collapses to the identity; intuitively, the randomizer r is never exposed *bare* but only through the level deficits $s_k - v_{i,k}$, none of which can be inverted inside the group. The residual $\mathbb{G}_T$ term $g_T^{\alpha r}$ ties r to α; because the admissibility constraint $i \notin \mathcal{D} \setminus \{i^*\}$ guarantees that registered non-target users cannot be treated as challenge identities by $\mathcal{A}$, and because the adversary holds no secret key α for identity i, this term supplies no way to isolate g_T^r. □

Proof of Theorem 2. Let $\mathcal{A}$ be a PPT adversary in the security game. We consider the following two games.

Game 0. This is the real security game $\mathrm{Sec}_{\mathcal{A}}^{RBE}(\lambda)$. The challenger generates crs and answers every registration query honestly. For an admissible challenge identity i^* and challenge messages (m_0, m_1), it samples $b \leftarrow \{0,1\}$ and $r \leftarrow \mathbb{Z}_p^*$, and returns

$$\mathsf{ct} = (ct_0, A_0, A_1, \ldots, A_\ell, C_m), \qquad C_m = g_T^r \cdot m_b,$$

with all remaining components generated as in Enc. Let W_0 denote the event that $\mathcal{A}$ outputs $b' = b$.

Game 1. This game is identical to Game 0 except that the factor g_T^r in C_m is replaced by an independently sampled uniform element $Z \leftarrow \mathbb{G}_T$; thus, the challenger sets

$$C_m = Z \cdot m_b.$$

Let W_1 denote the event that $\mathcal{A}$ outputs $b' = b$ in this game.

Since multiplication by the fixed group element m_b is a permutation of $\mathbb{G}_T$, the value $Z \cdot m_b$ is uniform and independent of b. All other ciphertext components are independent of b as well. Therefore,

$$\Pr[W_1] = \frac{1}{2}.$$

It remains to bound the difference between the two games. Suppose that

$$|\Pr[W_0] - \Pr[W_1]|$$

is non-negligible. We construct a PPT distinguisher $\mathcal{B}$ for the decisional (R,S,T,f)-Diffie–Hellman problem. Given a challenge $\big(U(\mathbf{x}), Z\big)$, where Z is either g_T^f or a uniform element of $\mathbb{G}_T$, $\mathcal{B}$ uses the components encoded in $U(\mathbf{x})$ to answer every non-target registration query by running the public Reg algorithm on the submitted (pk, ξ); this requires no discrete logarithm. If $\mathcal{A}$ makes a registration query for the target identity i, it uses the elements encoding $\alpha\chi_i$ and $\{\alpha \prod_{h=1}^{\ell-k} \chi_{i,h}\}_{k \in [\ell]}$ from R as (pk_i, ξ_i). The admissibility of the challenge identity ensures that this simulation does not require the target secret key.

When $\mathcal{A}$ announces its challenge identity i^*, $\mathcal{B}$ outputs an independent random bit and stops if $i^* \neq i$. Otherwise, admissibility guarantees that position i is either unregistered or was registered through the unique target-registration query. Let $\delta = 0$ in the former case and $\delta = 1$ in the latter. The distinguisher obtains the fresh opening proof $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$ from its maintained auxiliary state and takes from S

$$A_j = g_2^{r(s_j - v_{i,j})} \qquad (j \in [\ell]).$$

It computes

$$A_0 = (g_T^{\alpha r})^{\delta} \prod_{j=1}^{\ell} e(\pi_{i,j}, A_j).$$

The opening verification equation at position i gives

$$e(\mathsf{pp}, g_2) = e(g_1^{\alpha}, g_2)^{\delta} \prod_{j=1}^{\ell} e\Big(\pi_{i,j}, g_2^{s_j - v_{i,j}}\Big),$$

so the constructed value is exactly $A_0 = e(\mathsf{pp}, g_2)^r$.

For the challenge, $\mathcal{B}$ samples $b \leftarrow \{0,1\}$, sets

$$C_m = Z \cdot m_b,$$

and returns the resulting ciphertext to $\mathcal{A}$. When $\mathcal{A}$ outputs b', the distinguisher outputs 1 if and only if $b' = b$.

If $Z = g_T^f = g_T^r$, the simulation has the distribution of Game 0; if Z is uniform in $\mathbb{G}_T$, it has the distribution of Game 1. There is a sampling difference between the induced decisional experiment and the real scheme. The assumption samples the secret variables from $\mathbb{F}_p$, whereas the scheme samples $s_1, \ldots, s_\ell$, α, and r from $\mathbb{Z}_p^*$. Conditioning all $\ell + 2$ variables on being nonzero changes either branch by statistical distance at most $\frac{\ell+2}{p}$. Thus, after accounting for the identity guessed by $\mathcal{B}$,

$$|\Pr[W_0] - \Pr[W_1]| \leq N \cdot \left(\mathrm{Adv}_{\mathcal{B}}^{(R,S,T,f)}(\lambda) + \frac{2(\ell+2)}{p} \right).$$

Equivalently, if the induced decisional experiment samples the secret variables directly from $\mathbb{Z}_p^*$, the statistical term vanishes. Combining the preceding inequality with $\Pr[W_1] = 1/2$ gives

$$\left| \Pr\big[\mathcal{A} \text{ wins } \mathrm{Sec}_{\mathcal{A}}^{RBE}(\lambda)\big] - \frac{1}{2} \right| \leq N \cdot (\mathrm{Adv}_{\mathcal{B}}^{(R,S,T,f)}(\lambda) + \frac{2(\ell+2)}{p}).$$

We now make this conclusion concrete in the generic bilinear group model. Theorem 4 shows that $f = r$ is independent of $\langle R, S, T \rangle$. The maximal degrees of the induced tuple are $d_R = \ell + 1$, $d_S = \ell$, $d_T = 2$, and $d_f = 1$, and hence

$$d = \max\{d_R + d_S, d_T, d_f\} = 2\ell + 1.$$

Therefore, by Theorem 1, any distinguisher $\mathcal{B}$ making at most q generic group and pairing queries satisfies

$$\mathrm{Adv}_{\mathcal{B}}^{(R,S,T,f)}(\lambda) \leq \frac{(q+2c+2)^2(2\ell+1)}{2p},$$

where $c = \ell + 2$ is the number of secret variables in the induced instance. Substituting this bound into the game-hopping inequality above yields

$$\left|\Pr\left[\mathcal{A} \text{ wins } \mathrm{Sec}_{\mathcal{A}}^{RBE}(\lambda)\right] - \frac{1}{2}\right| \leq N \cdot \left(\frac{(q+2c+2)^2(2\ell+1)}{2p} + \frac{2(\ell+2)}{p}\right).$$

Since $\ell = \lceil \log n \rceil$, $c = \ell + 2$, N and q are polynomials in λ, and $p = \Theta(2^{\lambda})$, the right-hand side is negligible. Therefore, this construction is secure. □

D Proof of Verification Soundness

Proof sketch. We first consider membership soundness. Let

$$Mpk_i = g_1^{a_i}$$

be the membership public key generated together with the challenge RBE public key. If MVerify accepts, the first verification equation gives

$$e(pk_i, g_2) = e\left(Mpk_i, g_2^{\chi_{\bar{i}}(\mathbf{s})}\right) = e\left(g_1^{a_i \chi_{\bar{i}}(\mathbf{s})}, g_2\right).$$

By the non-degeneracy of the pairing, this implies

$$pk_i = g_1^{a_i \chi_{\bar{i}}(\mathbf{s})}.$$

Thus, pk_i is the RBE public key derived from a_i for identity i. The second verification equation shows that π_i is a valid opening for the value represented by Mpk_i at position i:

$$e\left(C^{(\tau)}/Mpk_i, g_2\right) = \prod_{t=1}^{\ell} e\left(\pi_{i,t}, g_2^{s_t - \bar{v}_{i,t}}\right).$$

By the binding property of Hyperproofs, the same position cannot be opened to a different value. Thus, the opening equation binds the value committed at coordinate $\bar{i}$ to a_i, while the first verification equation ensures that the submitted public key is $pk_i = g_1^{a_i \chi_{\bar{i}}(\mathbf{s})}$, namely, the public key induced by the same value a_i for identity i. Together, the two equations establish that Mpk_i and pk_i are consistent with the value committed at coordinate $\bar{i}$. For encryption randomness r, raising the opening equation to the power r gives

$$A_0 = e(g_1, g_2)^{a_i r} \prod_{t=1}^{\ell} e(\pi_{i,t}, A_t).$$

Hence, an accepting proof provides exactly the relation needed to simulate A_0, even when $C^{(\tau)}$ and π_i are chosen by the adversary. The remainder follows the same hybrid argument as the proof of Theorem 2: replacing $e(g_1, g_2)^r$ in C_m with a random element makes the ciphertext independent of the challenge bit, while distinguishing the two cases contradicts the decisional (R, S, T, f)-Diffie–Hellman assumption. For non-membership, an accepting proof opens position i to zero. Its verification equation similarly allows A_0 to be simulated, but without the term $e(g_1, g_2)^{a_i r}$. The same argument as in Theorem 2 then establishes message hiding for the unregistered identity. This proves both membership and non-membership soundness.