Efficient Registration-Based Encryption

Noemi Glaeser[1,2], Dimitris Kolonelos[3,4], Giulio Malavolta[1], and Ahmadreza Rahimi[1]

[1]Max Planck Institute for Security and Privacy
[2]University of Maryland
[3]IMDEA Software Institute
[4]Universidad Politecnica de Madrid

Abstract

Registration-based encryption (RBE) was recently introduced as an alternative to identity-based encryption (IBE), to resolve the key-escrow problem: In RBE, the trusted authority is substituted with a weaker entity, called the key curator, who has no knowledge of any secret key. Users generate keys on their own and then publicly *register* their identities and their corresponding public keys to the key curator. RBE is a promising alternative to IBE, retaining many of its advantages while removing the key-escrow problem, the major drawback of IBE. Unfortunately, all existing constructions of RBE use cryptographic schemes in a non black-box way, which makes them prohibitively expensive. It has been estimated that the size of an RBE ciphertext would be in the order of terabytes (though no RBE has even been implemented).

In this work, we propose a new approach to construct RBE, from well-studied assumptions in bilinear groups. Our scheme is black-box and it is concretely highly efficient—a ciphertext is 866 bytes. To substantiate this claim, we implemented a prototype of our scheme and we show that it scales to *millions of users*. The public parameters of the scheme are in the order of kilobytes. The most expensive operation (registration) takes a handful of seconds, whereas the encryption and decryption runtimes are in the order of milliseconds. This is the first ever implementation of an RBE scheme, and demonstrates that the practical deployment of RBE is already possible with today's hardware.

1 Introduction

Public-key encryption (PKE) [RSA78, DH76] is the cornerstone of secure communication: It allows Alice and Bob to privately exchange messages via a public channel, provided that they know each other's public key. Enabling Alice to learn Bob's public key (and vice versa) requires setting up a public key infrastructure (PKI) to certify the ownership of public keys. In practice, certification authorities are complex processes and PKI turns out to be a cumbersome and error-prone mechanism to implement.

To obviate these issues, Shamir [Sha84] proposed the notion of identity-based encryption (IBE), as an alternative to PKE with a simplified key management process: An IBE scheme allows Alice to encrypt her messages to Bob knowing just Bob's identity, along with some additional public parameters. Bob can then decrypt Alice's ciphertexts using an identity-specific secret key that he obtains from the key authority. Since the seminal work of Boneh and Franklin [BF01], IBE has seen tremendous progress with numerous new schemes being proposed over the years [Coc01, Wat05, Gen06, DG17]. In contrast, the practical deployment of IBE has been comparatively slow. Perhaps the main reason to hold back the widespread adoption of IBE is the so-called *key-escrow problem*: In an IBE scheme, the key authority can generate the secret key for any identity. This means that the key authority can decrypt *any message sent through the system*. This is not an acceptable compromise for many scenarios and has generated strong criticism against the notion [Rog15].

A recent work of Garg et al. [GHMR18, GHM+19] proposes a new paradigm for handling key-escrow, where the authority does not hold *any secret information* and it is completely transparent. They put forward the notion of registration-based encryption (RBE), which we recall next.

Registration-Based Encryption. In an RBE system, there is a dedicated party, called the key curator (KC), that is responsible for maintaining the updated system parameters. Users can register to the system by contacting the KC and sending their public key and identity. In response, the KC updates the public parameters of the system and returns to the user some supplementary information, which we refer to as *the opening*, that is useful to decrypt ciphertexts. Note that this opening serves only a functional purpose and does not substitute the user's secret key: In fact, it can be made public without affecting security. As public parameters change through the course of the system evolution, users will have to contact the KC to update their opening (i.e., the KC is also responsible for answering update queries). Importantly, we do not want to update every user's key every time a new public key is registered, so the definition of RBE imposes some efficiency requirement on the frequency of the updates.

Encryption and decryption works analogously as IBE: Given the public parameters and Bob's identity, Alice can *non-interactively* encrypt a message for Bob. Bob, given his secret key and the opening, can at a later point in time decrypt Alice's message. In terms of correctness, we require that decryption returns the correct message, provided that Bob registered his public key to the system prior to Alice computing her ciphertext. We also require that the size of the public parameters and the runtime of all algorithms is sublinear in the total number of registered users N. In terms of security, we require that encrypted messages must be hidden if no key was registered for the target identity, or if the key belongs to an honest party.

RBE can be thought as a hybrid notion between IBE and a PKI: Similar to the PKI settings, RBE requires a KC to keep the parameters up to date and (additionally) to issue updates to users. However, it has the strong advantage of offering the *encryption with respect to identity* functionality, which is the main appeal of IBE. On the flip side, RBE shares some similarities with IBE, since the users have to contact the KC in order to obtain the opening. However, the crucial difference is that the opening is not sufficient to decrypt ciphertexts, without the user's secret key. In other words, RBE does not suffer from the key escrow problem of IBE. As such, RBE is viewed as a promising alternative to the PKI architecture, without sacrificing the security of the users, also in contrast to PKI, in RBE the encryptor does not need to commmunicate with the PKI to obtain user's public keys in order to send an encrypted message.

Example Application. To exemplify the difference between RBE and PKI, consider a hypothetical scenario in which a whistle-blower wants to reveal some information to a journalist, but does not want to raise any suspicion. Querying a PKI to learn the journalist's public key may leak that some leak is about to be released, and may alert authorities. On the other hand, using RBE, the whistle-blower could encrypt a message that only the journalist is able to decrypt by just knowing the public parameters (that he stores locally anyway), without the need to contact any external entity. Although one may envision augmenting the PKI with *oblivious access* to the public keys (e.g., via private information retrieval), RBE allows us to achieve this functionality *completely non-interactively* and more efficiently. We stress that this is just one example where RBE achieves better properties than PKI, and other applications of RBE were already described in prior works (see, e.g., [GHM+19]).

The Black-Box Barrier. While RBE is considered to be a promising theoretical notion for a candidate substitute of the PKI architecture, current constructions of RBE [GHMR18, GHM+19] are prohibitively expensive. While such schemes achieve asymptotically excellent parameters, the concrete numbers are still nowhere close to the realm of practicality. It has been estimated [CES21] that, for an RBE supporting 2 billion users, ciphertexts would be as large as 11 terabytes (improved to 4.5 terabytes in [CES21]). This large overhead derives from the current construction paradigm of RBE, which is based on the garbled circuit tree of [CDG+17]. In more detail, known RBE constructions use cryptographic schemes in a *non black-box* manner, i.e., the scheme does not simply use cryptographic primitives in an input/output sense, but rather need access to their code. Traditionally, non black-box cryptographic schemes have been associated with prohibitive computational costs and a large body of literature in cryptography studies the possibility of constructing advanced primitives in a black-box way. We view known constructions of RBE as a manifestation of this phenomenon. Given the potential practical impact of this notion, the current state of affairs leaves open the following outstanding question:

Can we build efficient (and black-box) RBE?

We stress that a positive answer to the question above requires a conceptual rework of the current RBE construction paradigms, to circumvent the inefficiency barrier imposed by known (non black-box) constructions.

1.1 Our Results

We summarize our contributions below.

(i) Black-box RBE for Bounded Identity Sets. We propose a new method to construct RBE, which significantly departs from existing works. The resulting RBE scheme is black-box, and its security is based on well-studied assumptions over bilinear groups, namely the hardness of the bilinear Diffie-Hellman problem and the n-power Diffie-Hellman problem. The scheme is in the common reference string model and supports identities in the set $\{1, \ldots, N\}$, for any polynomially bounded N. The common reference string and the public parameters consist of at most $O(\sqrt{N})$ group elements, whereas ciphertexts and public/secret keys are composed of $O(1)$ group elements. Consequently, the key registration runs in time $O(\sqrt{N})$, whereas encryption, decryption, and updates are very efficient, requiring only $O(1)$ group operations. During the entire life cycle of the system, each user will have received an update at most $O(\sqrt{N})$-many times.

(ii) Extensions. Starting from the base RBE construction, we present several modifications that allow us to achieve a number of properties of interest.

- Efficient Updates: We show how to reduce the number of updates ever received by each user to $O(\log(\sqrt{N}))$, at the cost of increasing the size of the ciphertexts and of the public parameters by a multiplicative factor of $O(\log(\sqrt{N}))$.
- Verifiability: We describe an efficient procedure for the KC to prove that a certain public key was indeed registered in the public parameters [GV20]. The proof consists of a single group element and can be verified in time independent of N. We also show a method to prove the converse, i.e., that a public key was not registered, with essentially the same cost.
- Anonymity: We present a modification of our base scheme that achieves the desirable property of anonymity [GHM$^+$19]. Loosely speaking, an RBE is anonymous if it is computationally hard to distinguish ciphertexts encrypted for any two distinct identities, unless one possesses the appropriate secret key.
- CCA Security: Finally, we define the notion of CCA security for RBE and discuss how a minor modification of our construction already satisfies this property.

(iii) Implementation. To demonstrate the practicality of our approach, we implemented prototypes of our base and efficient-update constructions. Here we report numbers for our efficient-update construction. Without optimizing the code, we were able to scale our scheme up to 10 million users on a personal computer. In this regime, our construction has a common reference string of size 1MB and public parameters of size 2.8KB, the size of ciphertext for encrypting a single group element will be at most 10KB, and each update consists of a single group element (49B). In terms of runtime, registering a public key takes 2.59 seconds on average. On the other hand, encrypting and decrypting a ciphertext can be done in milliseconds.

1.2 Limitations

We explicitly discuss some shortcomings of our construction, when compared with prior works [GHMR18, GHM$^+$19, GV20]. First of all, our scheme has worse *asymptotic* parameters than prior RBE schemes where the size of the public parameters is poly-logarithmic in the size of the universe of users ($= N$). In contrast, our scheme achieves only a square root compression. We, also, note that in our construction the set of identities is $\{1, \ldots, N\}$ and therefore polynomially bounded.

Furthermore, the KC has to store an *auxiliary information*, containing the openings of all users, that grows with $O(N\sqrt{N})$ in our base construction, and with $O(N \log \sqrt{N})$ in the version with efficient updates.

In prior RBE schemes, the size of the auxiliary information was linear in N. We recall that the auxiliary information can be kept public and is only needed by the KC to issue updates correctly, but it is *not stored* by users.

Another important limitation is that our scheme requires a *structured* common reference string (i.e., a private-coin trusted setup), whereas prior work required only a *transparent* common reference string (i.e., a public-coin setup). Thus, our system must be initialized in a trusted manner, or via a multi-party computation (MPC) protocol. However, we stress that this operation needs to be done *once and for all*, and after that the KC can operate in a fully transparent manner.

Overall, we believe that the above limitations do not substantially reduce the practical applicability of our approach. For concrete parameter settings, a square root compression leads to very noticeable savings. This was already advocated in [CES21], and it is further demonstrated by our prototype implementation. As for the trusted setup, MPC protocols to sample our common reference string exist [BGM17] and have been in fact run by the Zcash community [Bow18] to achieve precisely the same goal.[1]

We also stress that polynomially bounded-sized identities can be sufficient in practical applications. As a concrete example, one could set the universe of identities to be the total set of phone numbers[2], and achieve the desired functionality of "encrypting with respect to the phone-number of a user". In addition, this is particularly useful in group messaging protocols (e.g. MLS protocol [BBM+20]) where currently a PKI is used. Furthermore, our RBE can be used as a square root compressing mechanism of a PKI, by setting a counter in $\{1, \ldots, N\}$ to each public key. On the other hand, bounded identities-RBE does not suffice for applications where identities should be arbitrarily large, such as email-based encryption.

Finally, the increase in the size of auxiliary input is modest, especially in the construction with efficient updates. As a concrete example, for $N = 10^6$ the size of the auxiliary information our construction with efficient updates is $2N \log \sqrt{N} \approx 20 \cdot N$ group elements, which is not far from the information needed to be stored by a PKI (i.e., N public keys).

1.3 Related Work

The notion of RBE was proposed in [GHMR18] and was later extended with the notion of anonymity [GHM+19] and verifiability [GV20], and was also optimized in [CES21]. All of these constructions follow roughly the same template and make non black-box usage of cryptographic primitives. While asymptotically they achieve essentially optimal efficiency, the concrete computational and storage costs are prohibitively high. We also mention that in [GHM+19] the authors also introduced the notion of timestamp RBE (T-RBE), as an intermediate abstraction. This is similar to the notion of RBE for bounded identity sets, as the users are indexed by time stamps in $\{1, \ldots, N\}$. However, the crucial difference is that in T-RBE the time stamps (and therefore the identities of the users) need to register in increasing order, whereas in our RBE settings the parties can register *in any order*, therefore T-RBE is a weaker notion than RBE for bounded identity sets.

Prior to the introduction of RBE, various attempts had been made to mitigate the key-escrow problem of IBE, though none of which fully addressing it. Boneh and Franklin [BF01] first proposed the existence of multiple private key generator authorities, an idea which was later developed by follow-up works [CHSS02, PS08, KG10, LBD+04], which however only distributes the escrow problem among many parties instead of a single one. Goyal [Goy07] introduced and [GLSW08] further developed the notion of accountable IBE, in which the private key authority is made accountable, but still not eliminated key-escrow. Chow [Cho09] and later Wei et al. [WQT18] used anonymous ciphertexts so that the authority cannot tell which is corresponding secret key, in combination to large number of identities (so that a brute-force search is hard), which even so required that each identity generation had some guaranteed randomness. Finally, in [AP03] the intermediate notion of certificateless public-key cryptography was introduced, however here a party cannot encrypt solely with respect to the identity but needs some extra information.

[1] In fact our trusted setup is even simpler, it *almost* lies in the 'power-of-tau' category, and would require only the first phase of the MPC protocol to run.

[2] A rough upper bound on the total number of existing phone numbers on earth is $N = 10^{10}$, which is within reach of our approach.

1.4 Technical Overview

We provide here a rough outline of the main technical steps of our approach to construct RBE. To put our work into context, we briefly recall the main ideas behind prior constructions of RBE.

Merkle-Hashing with Encryption. The main idea behind existing RBE constructions [GHMR18] is to design a special-purpose hash function that supports an "encryption" functionality. Specifically, the KC uses such a function to hash all key-identity pairs $(\mathsf{pk}, \mathsf{id})$ in a Merkle-tree and sets the root of the tree to be the public parameters pp of the RBE system. This additional encryption functionality allows anyone that knows the root of the hash (pp) to encrypt a message m with respect to an identity id. The resulting ciphertext can be decrypted by knowing (i) the pre-image of the hash and (ii) the secret key sk corresponding to pk. Observe that the KC is only responsible for correctly hashing all the public keys, but at no point in time is required to know any secret key. In fact, the pair $(\mathsf{sk}, \mathsf{pk})$ is sampled locally by the user associated with the identity id.

The main technical step of their approach is then how to build such a "hash with encryption". This is realized using a special-purpose two-to-one hash (which can be constructed from a variety of assumption) in conjunction with garbled circuits [Yao82]. This can be then turned into a hash for arbitrarily long messages via the standard Merkle tree construction and a chain of garbled circuits, where the last outputs an encryption of the message m under the public key pk that was hashed in the tree. Decryption requires knowing the root-to-leaf path for the Merkle tree, along with sk. Note that such root-to-leaf paths may change as new public-keys are added to the hash, which is the reason why users need to receive updates. Asymptotically, they achieve RBE with close to optimal efficiency. Unfortunately this comes at the cost of running expensive cryptographic operations inside garbled circuits.

Vector Commitments with Encryption. In this work we take a different approach to bypass the use of heavy cryptographic non black-box primitives. Our main idea is to substitute Merkle trees with an *algebraic data structure* to accumulate the public keys: Vector commitments [LY10, CF13] (VCs). VCs work analogously to Merkle trees, i.e., one can commit to a vector of elements so that at a later point one can selectively open any position of the vector. However, they offer algebraic structure that enables more advanced properties, e.g., batch opening [BBF19,LM19], opening aggregation [CFG$^+$20, GRWZ20, TAB$^+$20] or functional opening [LRY16,LP20]. Our observation is that, once the verification of an opening is algebraic (e.g., a pairing operation), one can hope to directly encrypt a message with respect to this verification equation, thus avoiding the non black-box tools.

More concretely, we make use of the [LY10] VC which works over bilinear groups. The commitment to a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$ is defined as $C = \prod_{i\in[N]} h_i^{x_i}$, whereas the opening value for position i is $\Lambda_i = \prod_{j\neq i}(h_{N+1-i+j})^{x_j}$. To verify that is x_i is the value at position i one checks if

$$e(C, h_{N+1-i}) \stackrel{?}{=} e(\Lambda_i, g) \cdot e(h_i^{x_i}, h_{N+1-i})$$

where $\{h_i\}_{i\in[2N]\setminus\{N+1\}}$ are part of the public parameters (henceforth referred to as the crs). It is binding under the q-Diffie-Hellman exponent assumption [BGW05], a well-established q-type (falsifiable) assumption over bilinear groups. To turn this into an RBE, we set the position i to be the identity of the user (implicitly we have N potential users), $\mathsf{sk}_i = x_i$ to be the secret key and $\mathsf{pk}_i = h_i^{x_i}$ to be public key. Then $C = \prod_{i\in[N]} \mathsf{pk}_i$ is the public parameter that accumulates the keys of the users (this is the analogue of the Merkle-tree root). Furthermore, Λ_i is the necessary *opening* a user should hold in order to be able to decrypt, provided by the KC.

Our goal is now to describe a procedure to generate a ciphertext ct that can be decrypted by knowing (i) the secret key x_i and (ii) the opening Λ_i that satisfies the above equation. On input a message m and an identity i, our encryption algorithm outputs (r the encryption randomness):

$$\mathsf{ct} = (e(C, h_{N+1-i})^r, g^r, e(h_i, h_{N+1-i})^r \cdot m)\,.$$

If Λ_i is a valid opening, then we can derive that

$$e(h_i, h_{N+1-i})^r = (e(C, h_{N+1-i})^r e(\Lambda_i, g^r))^{x_i^{-1}}\,.$$

Therefore a user that knows x_i can decrypt the ciphertext by computing $e(h_i, h_{N+1-i})^r$ as indicated and dividing the third component to recover m. On the other hand, it can be shown that if either the secret key or the opening is missing, then the message m is hidden.

Amortizing the Public Parameters. Although encryption/decryption, as described, is performed very efficiently, the scheme as described in not yet a valid RBE. Observe that the $\mathsf{crs} = \{h_i\}$ consists of $2N-1$ group elements, where N is the number of users. Thus, this is even worse than the trivial scheme where the public parameters simply consist of the concatenation of all public keys! Our next observation is that the crs is reusable across different commitments, and can therefore be *amortized*. Specifically, we divide the users in blocks of size $n = O(\sqrt{N}) : \{1, \ldots, n\}, \{n+1, \ldots, 2n\}, \ldots$ Then we create $B = N/n = O(\sqrt{N})$ vector commitments $(C_1, C_2, \ldots, C_B)$, one for each block, using the same crs. Now the public parameters of the RBE consist of an $O(\sqrt{N})$-sized crs and $O(\sqrt{N})$ number of commitments, therefore overall have size $O(\sqrt{N})$, sublinear on the number of users.

Extensions. The construction outlined above forms the basis for our template to construct RBE. We then show how to extend the scheme to lower the number of updates received by each user to $O(\log(\sqrt{N}))$ (Section 5.1), achieve anonymity (Section 5.3), verifiability (Section 5.2), and CCA security (Section 5.4). We refer the reader to the corresponding sections for more details.

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We say that a function is negligible if it vanishes faster than any polynomial. Given a set S, we denote by $s \leftarrow S$ the uniform sampling from S. We say that an algorithm is PPT if it can be implemented by a probabilistic machine running in time polynomial in the security parameter. We denote by $[n]$ the set $\{1, \ldots, n\}$.

2.1 Bilinear Groups

Let $\mathsf{bg} := (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ be a generator of a (symmetric) bilinear group generated by g of prime order p, with an efficiently computable pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We recall a few well-known assumptions in bilinear groups.

Assumption 1 (Bilinear Diffie-Hellman)**.** Let $\mathcal{BG}$ be a bilinear group generator. The bilinear Diffie-Hellman problem is hard for $\mathcal{BG}$ if the following distributions are computationally indistinguishable:

$$(p, \mathbb{G}, \mathbb{G}_T, g, e, g^x, g^y, g^z, e(g,g)^{xyz}) \approx (p, \mathbb{G}, \mathbb{G}_T, g, e, g^x, g^y, g^z, e(g,g)^w)$$

where $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ and $(x, y, z, w) \leftarrow \mathbb{Z}_p^*$.

It is going to be convenient for us to rephrase the above assumption as the hardness of distinguishing between the distributions, which can be easily reduced to the bilinear Diffie-Hellman assumption:

$$(p, \mathbb{G}, \mathbb{G}_T, g, e, g^x, e(g,g)^y, e(g,g)^{xy}) \approx (p, \mathbb{G}, \mathbb{G}_T, g, e, g^x, e(g,g)^y, e(g,g)^z)\,.$$

Assumption 2 (n-Power Diffie-Hellman)**.** Let $\mathcal{BG}$ be a bilinear group generator. The n-power Diffie-Hellman problem is hard for $\mathcal{BG}$ if the following distributions are computationally indistinguishable:

$$\left(p, \mathbb{G}, \mathbb{G}_T, g, e, \left\{g^{x^i}\right\}_{i \in [2n] \setminus \{n+1\}}, g^y, e(g,g)^{x^{n+1} y}\right) \approx \left(p, \mathbb{G}, \mathbb{G}_T, g, e, \left\{g^{x^i}\right\}_{i \in [2n] \setminus \{n+1\}}, g^y, e(g,g)^z\right)$$

where $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ and and $(x, y, z) \leftarrow \mathbb{Z}_p^*$.

Symmetric vs Asymmetric Pairings. Throughout this work, we describe our assumptions and our schemes using symmetric pairings. However, both the assumptions and the constructions are easily generalizable

to the setting where only an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is available. We chose to present our constructions in the symmetric settings for two reasons: (i) In terms of security, symmetric pairings are more general since one cannot rely on the hardness of mapping between the two source groups. This shows that our construction template is more robust, and remains secure even if there is an efficiently computable isomorphism between the two source groups. (ii) Symmetric pairings help avoiding overloading the reader with indices, and allow for a more elegant presentation of main ideas of the construction.

3 Definitions

Here we formally define RBE, and our definitions are taken almost in verbatim from [GHM+19]. Compared with [GHM+19], we modify the syntax of RBE to allow the key generation algorithm to take as input a common reference string crs. Furthermore, we fix a bound on universe of users N ahead of time. For notational convenience, we still refer to such primitive as an RBE, but we explicitly mention that the scheme only supports *bounded identity sets*.

Definition 1 (Registration-Based Encryption (RBE))**.** A registration-based encryption for a universe of identities $\{1, \ldots, N\}$ scheme consists of five PPT algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ working as follows. The Reg and Upd algorithms are performed by the key curator, which we call KC for short.

- **Setup.** $\mathsf{Setup}(1^\lambda, N) \to \mathsf{crs}$: The setup algorithm Setup samples a common reference string crs once and for all at the beginning of the protocol.
- **Key Generation.** $\mathsf{Gen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$: The randomized algorithm Gen takes as input the common reference string crs and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The key generation algorithm is run by any honest party locally who wants to register itself into the system.
- **Registration.** $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$: The deterministic algorithm Reg takes as input the common random sting crs, current public parameter pp, a registering identity id and a public key pk (supposedly for the identity id), and it outputs pp' as the updated public parameters. The Reg algorithm uses *read* and *write* oracle access to the auxiliary information aux which will be updated into aux' during the process of registration. (The system is initialized with public parameters pp and auxiliary information aux set to $\perp$.)
- **Encryption.** $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$: The randomized algorithm Enc takes as input the common random sting crs, a public parameter pp, a recipient identity id, and a plaintext message m and outputs a ciphertext ct.
- **Update.** $\mathsf{Upd}^{[\mathsf{aux}]}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{u}$: The deterministic algorithm Upd takes as input the current parameters pp stored at the KC, an identity id, and a public key pk. It has *read only* oracle access to aux and generates an update information u that can help id to decrypt its messages.
- **Decryption.** $\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to m$: The deterministic decryption algorithm Dec takes as input a secret key sk, an update information u, and a ciphertext ct, and it outputs a message $m \in \{0,1\}^*$ or in $\{\perp, \mathsf{GetUpd}\}$. The symbol $\perp$ indicates a syntax error while GetUpd indicates that more recent update information (than u) might be needed for decryption.

Definition 2 (Completeness, compactness, and efficiency of RBE)**.** For any interactive *computationally unbounded* adversary $\mathcal{A}$ with polynomial query complexity, consider the following game $\mathsf{EXP}^{\mathtt{Comp}}_{\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. **Initialization.** $\mathcal{C}$ sets $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathsf{u} = \perp$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \perp$, $t = 0$, and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, and sends the sampled crs to $\mathcal{A}$.
2. Until $\mathcal{A}$ continues (which is at most $\mathsf{poly}(\lambda)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to be performed.

(a) **Registering new (non-target) identity**. $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}$ and pk in the support of the $\mathsf{Gen}(\mathsf{crs})$ algorithm, to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.

(b) **Registering the target identity.** If $\mathsf{id}^* \neq \bot$, skip this step. Otherwise, $\mathcal{A}$ sends some $\mathsf{id}^* \notin \mathcal{D}$ to $\mathcal{C}$. $\mathcal{C}$ then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(\mathsf{crs})$, updates $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$ and $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, and sends pk^* to $\mathcal{A}$.

(c) **Encrypting for the target identity.** If $\mathsf{id}^* = \bot$, then skip this step. Otherwise, $\mathcal{C}$ sets $t \coloneqq t + 1$. $\mathcal{A}$ sends some $m_t \in \{0,1\}^*$ to $\mathcal{C}$ who sends back a corresponding ciphertext $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_t)$ to $\mathcal{A}$.

(d) **Decryption by target identity.** $\mathcal{A}$ sends a $j \in [t]$ to $\mathcal{C}$. $\mathcal{C}$ then lets $m'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$. If $m'_j = \mathsf{GetUpd}$, then $\mathcal{C}$ obtains the update $\mathsf{u}^* = \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ and then lets $m'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}^*, \mathsf{ct}_j)$.

3. The adversary $\mathcal{A}$ wins the game if there is some $j \in [t]$ for which $m'_j \neq m_j$.

Let $Q \in \mathsf{poly}$ be an upper bound on the number of queries issued by $\mathcal{A}$ and let $\tilde{N} = |\mathcal{D}| \leq N$ at the end of the execution. We require the following properties to hold.

- **Completeness.** For all computationally unbounded $\mathcal{A}$, $\Pr[\mathcal{A} \text{ wins } \mathsf{EXP}^{\mathtt{Comp}}_{\mathcal{A}}(\lambda)] = 0$.
- **Compactness of public parameters and updates.** For all queries $q \in [Q]$, let pp_q be the public parameters after the q-th query. Then $|\mathsf{pp}|$ is sublinear in $\tilde{N}$. Moreover, for all $\mathsf{id} \in \mathcal{D}$, the size of the corresponding update $|\mathsf{u}_q|$ is also sublinear in $\tilde{N}$.
- **Efficiency of runtime of registration and update.** The running time of each invocation of Reg and Upd algorithms is sublinear in $\tilde{N}$.
- **Efficiency of the number of updates.** The total number of invocations of Upd for identity id^* in Step 2(d) of the game $\mathsf{EXP}^{\mathtt{Comp}}_{\mathcal{A}}(\lambda)$ is sublinear in $\tilde{N}$.

Definition 3 (Security of RBE)**.** For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathsf{EXP}^{\mathtt{CPA}}_{\mathcal{A}}(\lambda)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. **Initialization.** $\mathcal{C}$ sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \bot$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and sends the sampled crs to $\mathcal{A}$.
2. Until $\mathcal{A}$ continues (which is at most $\mathsf{poly}(\lambda)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to be performed.

   (a) **Registering new (non-target) identity**. $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}$ and pk to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.

   (b) **Registering the target identity.** If $\mathsf{id}^* \neq \bot$, skip this step. Otherwise, $\mathcal{A}$ sends some $\mathsf{id}^* \notin \mathcal{D}$ to $\mathcal{C}$. $\mathcal{C}$ then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(\mathsf{crs})$, updates $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, and sends pk^* to $\mathcal{A}$.
3. **Encrypting for the target identity.** $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D} \setminus \{\mathsf{id}^*\}$ and two messages (m_0, m_1) and $\mathcal{C}$ generates $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$, where $b \leftarrow \{0,1\}$ is a random bit, and sends ct to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.

We call an RBE scheme secure if there exists a negligible function negl such that for all PPT adversaries $\mathcal{A}$ it holds that $\left|1/2 - \Pr[\mathcal{A} \text{ wins } \mathsf{EXP}^{\mathtt{CPA}}_{\mathcal{A}}(\lambda)]\right| \leq \mathsf{negl}(\lambda)$.

4 Our Base RBE Construction

4.1 Our Base Construction

Here we present our base RBE for bounded identity sets construction. We let N be the maximum number of users to be registered and we divide the N users in blocks of n and subsequently we get $B := \lceil \frac{N}{n} \rceil$ number of blocks. We denote by $\bar{i}$ the modular reduction $\bar{i} := (i \mod n)+1$, for any $i \in \mathbb{Z}$. As outlined in Section 1.4 the key curator (KC) has to manage B vector commitments, committing to the secret keys of the users, *without* knowing the secret keys. Each committed vector consists of n keys, thus overall $N = Bn$ keys are committed. That is, KC needs to maintain the commitments $C^{(1)}, \ldots, C^{(B)}$ and up-to-date opening values for each user, $\Lambda_1, \ldots, \Lambda_N$, so that any user can query her opening value. We will use the vector commitment from [LY10], where the commitments and the openings are of the form

$$C = \prod_{i \in [n]} h_i^{x_i} \text{ and } \Lambda_i = \prod_{j \neq i} (h_{n+1-i+j})^{x_j}$$

where $\{h_i\}_{i \in [2n] \setminus n+1}$ are part of the crs. It will be convenient to think of $i := \mathsf{id}$ as the identity of the user and $x_i := \mathsf{sk}_{\mathsf{id}}$ as the user's secret key.

Overview. For simplicity, we illustrate how the algorithms operate over the commitment $C^{(1)}$ for identities in $\{1, \ldots, n\}$, the general case follows by a straightforward generalization and it is formally described in Figure 1. The public key of the user id is set to $\mathsf{pk}_{\mathsf{id}} = (h_{\mathsf{id}})^{\mathsf{sk}_{\mathsf{id}}}$ and the commitment is simply the accumulation of all the public keys $C^{(1)} = \prod_{\mathsf{id} \in [n]} \mathsf{pk}_{\mathsf{id}}$. However, once a user joins the system, we also have to update the opening information $\{\Lambda_{\mathsf{id}}\}_{\mathsf{id} \in [n]}$, since its computation includes the secret keys of all the other users in the block:

$$\Lambda_{\mathsf{id}} = \prod_{\mathsf{id}' \neq \mathsf{id}} (h_{n+1-\mathsf{id}+\mathsf{id}'})^{\mathsf{sk}_{\mathsf{id}'}}.$$

We resolve this by letting the users compute locally all terms that are used for different openings. For example id computes $(h_{n+1-1+\mathsf{id}})^{\mathsf{sk}_{\mathsf{id}}}$ (used in Λ_1), $(h_{n+1-2+\mathsf{id}})^{\mathsf{sk}_{\mathsf{id}}}$ (used in Λ_2), and so on (excluding $(h_{n+1-\mathsf{id}+\mathsf{id}})^{\mathsf{sk}_{\mathsf{id}}}$ that is used nowhere). We call these values *helping information* and denote them by

$$\boldsymbol{\xi} = \left((h_{\mathsf{id}+n})^{\mathsf{sk}_{\mathsf{id}}}, \ldots, (h_{\mathsf{id}+1})^{\mathsf{sk}_{\mathsf{id}}} \right).$$

When a new user $\mathsf{id} \in [n]$ registers, it sends her $\mathsf{pk}_{\mathsf{id}}$ and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ to the KC. Then the KC first checks the correctness of the helping information (using pairings) and updates the corresponding commitment as $C^{(1)} \leftarrow C^{(1)} \cdot \mathsf{pk}_{\mathsf{id}}$ and all the corresponding openings in the cluster as $\Lambda_{\mathsf{id}'} \leftarrow \Lambda_{\mathsf{id}'} \cdot \xi_{\mathsf{id}'}$ for each $\mathsf{id}' \in [n] \setminus \{\mathsf{id}\}$. Afterwards, KC can delete $\boldsymbol{\xi}$.

Furthermore, we note that a user needs to keep track of all the openings throughout the evolution of the system. That is because a ciphertext may have been encrypted with respect to any instance of the public parameters (thus any $C^{(1)}$). Therefore the KC (and the user) should maintain a list of openings $L_{\mathsf{id}} = \{\Lambda_{\mathsf{id},1}, \ldots, \Lambda_{\mathsf{id},\ell}\}$, instead of a single opening, that correspond to all the opening values changes that have occurred until this point. Concretely, ℓ is the number of users that have registered in this block so far and $\Lambda_{\mathsf{id},j}$ is the corresponding openings. It follows that $\ell \leq n$.

Encryption and decryption are done (as described in Section 1.4) with respect to the verification equation of the vector commitment. This is accompanied with a mechanism to make the user aware of the fact that the opening of the vector commitment is outdated. In that case, the user simply requests an update from the KC.

$\mathsf{Setup}(1^\lambda, n) \to \mathsf{crs}:$ generate a bilinear group $\mathsf{bg} := (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ and sample $z \leftarrow \mathbb{Z}_p^*$. Sets $h_i = g^{z^i}$ for each $i \in [2n]$ and output

$$\mathsf{crs} = \left(\mathsf{bg}, \{h_i\}_{i \in [2n] \setminus \{n+1\}}\right).$$

$\mathsf{Gen}(\mathsf{crs}, \mathsf{id}) \to (\mathsf{pk}, \mathsf{sk}, \boldsymbol{\xi}):$ If $\mathsf{id} \notin [N]$ output $\perp$. Otherwise, sample $x_{\mathsf{id}} \leftarrow \mathbb{Z}_p^*$, compute $\bar{\mathsf{id}} = (\mathsf{id} \mod n) + 1$ and set

$$\mathsf{pk} = (h_{\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \qquad \mathsf{sk} = x_{\mathsf{id}}, \qquad \boldsymbol{\xi} = \left((h_{\bar{\mathsf{id}}+n})^{x_{\mathsf{id}}}, \ldots, (h_{2+n})^{x_{\mathsf{id}}}, \emptyset, (h_n)^{x_{\mathsf{id}}}, \ldots, (h_{\bar{\mathsf{id}}+1})^{x_{\mathsf{id}}}\right)$$

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \boldsymbol{\xi}) \to \mathsf{pp}':$ The key curator proceeds as follows:

- If it is the first identity to be registered then initialize $\mathsf{pp} = (1, \ldots, 1) \in \mathbb{G}^B$ and $\mathsf{aux} = (\{1\}, \ldots, \{1\}) \in \mathbb{G}^N$.
- If $\mathsf{id} \notin [N]$ output $\perp$.
- Check the correctness of $\boldsymbol{\xi}$:

$$e(\mathsf{pk}, h_n) = e(\boldsymbol{\xi}_1, g) = \cdots = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}-1}, h_{\bar{\mathsf{id}}-2}) = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}+1}, h_{\bar{\mathsf{id}}}) = \ldots = e(\boldsymbol{\xi}_n, h_{n-1})$$

  where $\bar{\mathsf{id}} = \mathsf{id} \mod n$. If the above doesn't hold output $\perp$.
- Otherwise, parse $\mathsf{pp} = \left(C^{(1)}, \ldots, C^{(B)}\right)$, compute $k = \lceil \frac{\mathsf{id}}{n} \rceil$ and set $C'^{(k)} \leftarrow C^{(k)} \cdot \mathsf{pk}$, the rest remain the same $C'^{(j)} = C^{(j)}$ for each $j \in [B] \setminus \{k\}$. Output

$$\mathsf{pp}' = (C'^{(1)}, \ldots, C'^{(B)})$$

- Parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and for each $j \in \{(k-1)n + 1, \ldots, (k-1)n + n\} \setminus \{\mathsf{id}\}$ parse $L_j = \{\Lambda_{j,1}, \ldots, \Lambda_{j,\ell}\}$. Set $L_j = L_j \cup \{\Lambda_{j,\ell+1} = \Lambda_{j,\ell} \cdot \boldsymbol{\xi}_j\}$, while the rest remain the same.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}:$ On input $m \in \mathbb{G}_T$, let $\bar{\mathsf{id}} = \mathsf{id} \mod n$ and $k = \lceil \frac{\mathsf{id}}{n} \rceil$. Sample $r \leftarrow \mathbb{Z}_p^*$ and compute:

$$\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3) = \left(C^{(k)}, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^r, g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r \cdot m\right).$$

$\mathsf{Upd}^{[\mathsf{aux}]}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}:$ parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and output L_{id}.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to m$ **or** $\mathsf{GetUpd}:$ Parse $\mathsf{u} = \{\Lambda_1, \ldots, \Lambda_\ell\}$, if there exists no $i \in [\ell]$ such that $e(\mathsf{ct}_0, h_{n+1-\bar{\mathsf{id}}}) = e(\Lambda_i, g) \cdot e((h_{\bar{\mathsf{id}}})^{\mathsf{sk}}, h_{n+1-\bar{\mathsf{id}}})$, then output GetUpd.[3]Otherwise let i be such index and return

$$\frac{\mathsf{ct}_3}{\left(e(\Lambda_i, \mathsf{ct}_2)^{-1} \cdot \mathsf{ct}_1\right)^{\mathsf{sk}^{-1}}}.$$

Figure 1: Our main RBE construction.

Correctness. We can prove the correctness for party id with key pair $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$ by expanding the above equation. For notational convenience, we assume that $\mathsf{id} \in [n]$ and thus the corresponding commitment is $C^{(1)}$. Additionally, we assume that all n identities are registered under this commitment, hence $C^{(1)} = \prod_{i \in [n]} h_i^{\mathsf{sk}_i}$.

	Setup	Gen	Reg	Enc	Upd	Dec	#Upd	Prove	Verify
[GHMR18, §4]	1	1	$\log M$	$\log M$	$\log M$	$\log M$	$\log M$	–	–
[GHM$^+$19, CES21]	1	1	$\log^2 M$	$\log^2 M$	$\log M$	$\log^2 M$	$\log M$	–	–
[GV20]	1	1	$\log^2 M$	$\log^2 M$	$\log M$	$\log M$	$\log M$	$\log^2 M$	$\log^2 M$
Sec. 4	$\sqrt{N}$	$\sqrt{N}$	$\sqrt{N}$	1	1	1	$\sqrt{N}$	1	1
Sec. 5.1	$\sqrt{N}$	$\sqrt{N}$	$\sqrt{N}\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$

	$\lvert\mathsf{crs}\rvert$	$\lvert\mathsf{pp}\rvert$	$\lvert\mathsf{aux}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{u}\rvert$	$\lvert\pi\rvert$
[GHMR18, §4], [GHM$^+$19]	–	$\log M$	M	1	1	$\log M$	$\log M$	–
[CES21]	1	$\log M$	M	1	1	$\log M$	$\log M$	–
[GV20]	1	$\log M$	M	1	1	$\log M$	$\log M$	$\log^2 M$
Sec. 4	$\sqrt{N}$	$\sqrt{N}$	$M\sqrt{N}$	1	1	1	$\sqrt{N}$	1
Sec. 5.1	$\sqrt{N}$	$\sqrt{N}\log\sqrt{N}$	$N\log\sqrt{N}$	1	1	$\log\sqrt{N}$	$\log\sqrt{N}$	$\log\sqrt{N}$

Table 1: Comparison of worst-case asymptotic complexities of this work and previous works. The parameter sizes of our scheme are given in number of group elements, so we omit factors of λ. M denotes the number of registered users, and N the capacity of the system. The cost of the Prove and Verify algorithms refers to both membership and non-membership proofs.

Consequently, the corresponding update for id is $\Lambda = \prod_{i\in[n]\setminus\{\mathsf{id}\}} \boldsymbol{\xi}_{i,\mathsf{id}}$. We can then write:

$$\begin{aligned}
\frac{\mathsf{ct}_3}{\left(e(\Lambda,\mathsf{ct}_2)^{-1}\cdot\mathsf{ct}_1\right)^{\mathsf{sk}_{\mathsf{id}}^{-1}}} &= \frac{e(h_{\mathsf{id}},h_{n+1-\mathsf{id}})^r\cdot m}{\left(e(\Lambda,g^r)^{-1}\cdot e(C^{(1)},h_{n+1-\mathsf{id}})^r\right)^{\mathsf{sk}_{\mathsf{id}}^{-1}}} \\
&= \frac{e(h_{\mathsf{id}},h_{n+1-\mathsf{id}})^r\cdot m}{e(\prod\limits_{i\in[n]\setminus\{\mathsf{id}\}}\boldsymbol{\xi}_{i,\mathsf{id}},g^r)^{\frac{-1}{\mathsf{sk}_{\mathsf{id}}}}\cdot e(\prod\limits_{i\in[n]}h_i^{\mathsf{sk}_i},h_{n+1-\mathsf{id}})^{\frac{r}{\mathsf{sk}_{\mathsf{id}}}}} \\
&= \frac{e(h_{\mathsf{id}},h_{n+1-\mathsf{id}})^r\cdot m}{\prod\limits_{i\in[n]\setminus\{\mathsf{id}\}}e(\boldsymbol{\xi}_{i,\mathsf{id}},g)^{\frac{-r}{\mathsf{sk}_{\mathsf{id}}}}\cdot\prod\limits_{i\in[n]}e(h_i^{\mathsf{sk}_i},h_{n+1-\mathsf{id}})^{\frac{r}{\mathsf{sk}_{\mathsf{id}}}}} \\
&= \frac{m}{\prod\limits_{i\in[n]\setminus\{\mathsf{id}\}}e(h_{i+n-\mathsf{id}+1}^{\mathsf{sk}_i},g)^{\frac{-r}{\mathsf{sk}_{\mathsf{id}}}}\cdot e(h_i^{\mathsf{sk}_i},h_{n+1-\mathsf{id}})^{\frac{r}{\mathsf{sk}_{\mathsf{id}}}}} \\
&= \frac{m}{\prod\limits_{i\in[n]\setminus\{\mathsf{id}\}}e(g^{z^{i+n-\mathsf{id}+1}\mathsf{sk}_i},g)^{\frac{-r}{\mathsf{sk}_{\mathsf{id}}}}\cdot e(g^{z^i\mathsf{sk}_i},g^{z^{n+1-\mathsf{id}}})^{\frac{r}{\mathsf{sk}_{\mathsf{id}}}}} \\
&= m.
\end{aligned}$$

The scheme is also multiplicatively homomorphic over $\mathbb{G}_T$, where the homomorphic operation can be computed by multiplying two ciphertexts component-wise.

4.2 Efficiency

Setting the parameter as $n = B = \sqrt{N}$ optimizes the public information a user should store: Both crs and pp are $O(\sqrt{N})$ group elements. Below we provide an efficiency analysis adopting this choice, $n = \sqrt{N}$. However, we stress that alternative options, that optimize different aspects of the system, can always be considered. Here we give an asymptotic efficiency analysis, parametrized by the number of users N, and we show a comparison with prior works in Table 1.

[3] In a slight abuse of notation, we assume that the algorithm is also given id as input.

Regarding the computational complexity, generating the crs requires $O(\sqrt{N})$ group exponentiations and is only done once at the system setup. Similarly, the key-generation of a user takes $O(\sqrt{N})$ group exponentiations, and is only executed once by each user. Registration of a user by the KC takes $O(\sqrt{N})$ pairing checks and then 1 group multiplication to compute the new pp and $O(\sqrt{N})$ group multiplications to update the aux. Finally, Enc, Dec, and Upd are very efficient requiring $O(1)$-many group operations.[4]

As for the storage requirements, the crs and the pp stored by the KC and the users are both $O(\sqrt{N})$ group elements, while pk and sk are 1 group and field element respectively.[5] The aux stored by the Key Curator consists of at most $O(N^{3/2})$ group elements, $O(\sqrt{N})$ corresponding to each participating user. More specifically, if M users are registered then $|\mathsf{aux}| = O(M\sqrt{N})$ in the worst-case. Each user needs to make up to $O(\sqrt{N})$ Upd-queries to the KC and store the update information u of size $O(\sqrt{N})$. In Section 5.1 we show a variant of our construction that reduces both these to $O(\log(N))$ and the aux of the KC to $|\mathsf{aux}| = O(N \log \sqrt{N})$, at the cost of having public parameters of size $|\mathsf{pp}| = O(\sqrt{N} \log(N))$ instead. Finally any ciphertext consists of 2 $\mathbb{G}$-elements and 2 $\mathbb{G}_T$-elements.

We also remark that the total number of users in the system is determined by the parameter N, which is chosen ahead of time and fixed throughout the life of the system. However, we stress that N is not a hard bound: If at some point in time the total amount of identities exceed N, nothing prevents one from appending new commitments to the public parameters to increase the identity bound (say from N to $2N$). I.e., one does not have to re-initialize the system, even if the identity bound is exceeded.

4.3 Security Analysis

In the following we state and prove our main security theorem.

Theorem 1 (Security)**.** If the bilinear Diffie-Hellman and the n-Power Diffie-Hellman assumptions hold for $\mathcal{BG}$, then the scheme as described above satisfies security.

Proof. In the following, we consider an adversary having registered N' arbitrary identities, $0 \leq N' \leq N$, i.e. $|\mathcal{D}| = N'$. The target identity $\mathsf{id}^* \notin \mathcal{D}$ chosen by the adversary $\mathcal{A}$ may or may not be registered by the challenger $\mathcal{C}$, according to the definition. In concrete:

- (Step 1) The challenger first $\mathcal{C}$ samples a $\mathsf{crs} = (\mathsf{bg}, \{g^{z^i}\}_{i \in [2n] \notin \{n+1\}})$ and performs the initialization.
- (Step 2) Until $\mathcal{A}$ exits step 2 of the security definition $\mathcal{C}$ registers the identities chosen by $\mathcal{A}$. At the end N' identities have been registered.

Let $\mathcal{D} = \{\mathsf{id}_1, \ldots, \mathsf{id}_{N'}\}$ be the registered identities, observe that for each $\mathsf{id} \in \mathcal{D}$ the challenger receives $\mathsf{pk}, \boldsymbol{\xi}$ such that:

$$e(\mathsf{pk}, h_n) = e(\boldsymbol{\xi}_1, g) = \cdots = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}-1}, h_{\bar{\mathsf{id}}-2}) = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}+1}, h_{\bar{\mathsf{id}}}) = \ldots = e(\boldsymbol{\xi}_n, h_{n-1}).$$

Since the order of the group p is prime (thus the group is cyclic), we can write $\mathsf{pk} = (h_{\bar{\mathsf{id}}})^{x_{\mathsf{id}}}$, for some $x_{\mathsf{id}} \in \mathbb{Z}_p^*$. Then $\boldsymbol{\xi}_j = (h_{\bar{\mathsf{id}}+n-(j-1)})^{x_{\mathsf{id}}}$.

- (Step 3) Encrypting the target identity we get a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_b)$. Concretely:

$$\mathsf{ct} = \left(C^{(k)}, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}^*})^r, g^r, e(h_{\bar{\mathsf{id}}^*}, h_{n+1-\bar{\mathsf{id}}^*})^r \cdot m_b\right)$$

Next, in order to prove security of our construction we define the following sequence of games:

[4] For Dec we assume that the decrypter knows the correct opening for a given ciphertext, which is naturally the case if the user is up-to-date and the ciphertext recent. Otherwise its worse-case complexity is bounded by $O(\sqrt{N})$ (base construction) or by $O(\log \sqrt{N})$ (efficient updates).

[5] The helping information $\boldsymbol{\xi}$ is generated by the user once during the registration. It is not needed to later be stored neither by the KC nor the user.

$\underline{\mathsf{Game}_0}$: Is the original game of RBE security described above. Here:

$$\mathsf{ct} = \left(C^{(k)}, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}^*})^r, g^r, e(h_{\bar{\mathsf{id}}^*}, h_{n+1-\bar{\mathsf{id}}^*})^r \cdot m_b\right)$$

where $k = \lceil \frac{\mathsf{id}^*}{n} \rceil$, $\bar{\mathsf{id}}^* = \mathsf{id}^* \mod n$.

It will be useful to recall the structure of $C^{(k)}$:

1. $\underline{\text{Case 1}}$: if id^* was registered by $\mathcal{C}$ then

$$C^{(k)} = (h_{\bar{\mathsf{id}}^*})^{x_{\mathsf{id}^*}} \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} (h_{\bar{i}})^{x_i}$$

where $k = \lceil \frac{\mathsf{id}^*}{n} \rceil$ and $\mathcal{D}^{(k)} = \mathcal{D} \cap \{\mathsf{id}\}_{\mathsf{id}=(k-1)n+1}^{(k-1)n+n}$. Consequently:

$$\begin{aligned}
e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}^*}\right)^r &= e\left((h_{\bar{\mathsf{id}}^*})^{x_{\mathsf{id}^*}} \cdot \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} (h_{\bar{i}})^{x_i}, h_{n+1-\bar{\mathsf{id}}^*}\right)^r \\
&= e\left((h_{\bar{\mathsf{id}}^*})^{x_{\mathsf{id}^*}}, h_{n+1-\bar{\mathsf{id}}^*}\right)^r \cdot \\
&\qquad \cdot \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} e\left((h_{\bar{i}})^{x_i}, h_{n+1-\bar{\mathsf{id}}^*}\right)^r \\
&= e\left(g, g\right)^{z^{n+1} r x_{\mathsf{id}^*}} \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} e\left(g, g\right)^{z^{\bar{i}+n+1-\bar{\mathsf{id}}^*} x_i r} \\
&= e\left(g, g\right)^{z^{n+1} r x_{\mathsf{id}^*}} \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} e\left(g^{z^{\bar{i}+n-(\bar{\mathsf{id}}^*-1)} x_i}, g^r\right) \\
&= e\left(g, g\right)^{z^{n+1} r x_{\mathsf{id}^*}} e\left(\prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}^*}, g^r\right)
\end{aligned}$$

2. $\underline{\text{Case 2}}$: if id^* was not registered then:[6]

$$C^{(k)} = \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} (h_{\bar{i}})^{x_i}$$

where $k = \lceil \frac{\mathsf{id}^*}{n} \rceil$ and $\mathcal{D}^{(k)} = \mathcal{D} \cap \{\mathsf{id}\}_{\mathsf{id}=(k-1)n+1}^{(k-1)n+n}$ and similarly to the previous case:

$$e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}^*}\right)^r = \prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} e\left(\boldsymbol{\xi}_{i,\bar{\mathsf{id}}^*}, g^r\right)$$

$\underline{\mathsf{Game}_1}$: Is the same to Game_0, except for step 3 (the encryption step); we change the ciphertext according to the two cases below:

1. $\underline{\text{Case 1}}$: if $\mathcal{A}$ chooses to register id^* we set the ciphertext:

$$\begin{aligned}
&\mathsf{ct}_0 = C^{(k)}, \\
&\mathsf{ct}_1 = e(g,g)^{s \cdot x_{\mathsf{id}^*}} e\left(\prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}^*}, g^r\right) \\
&\mathsf{ct}_2 = g^r, \quad \mathsf{ct}_3 = e(g,g)^s m_b
\end{aligned}$$

for a random $s \leftarrow \mathbb{Z}_p^*$.

[6] We note that $\setminus\{\mathsf{id}^*\}$ in the product is written for emphasis, since $\mathcal{D}^{(k)}$ doesn't include it anyway.

2. Case 2: if $\mathcal{A}$ chooses not to register id^* then we set:

$$\mathsf{ct}_0 = C^{(k)}, \mathsf{ct}_1 = e\left(\prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}^*}}, g^r\right)$$
$$\mathsf{ct}_2 = g^r, \quad \mathsf{ct}_3 = e(g,g)^s m_b$$

for a random $s \leftarrow \mathbb{Z}_p^*$.

In both cases, the distributions induced by Game_0 and Game_1 are computationally indistinguishable assuming the n-power Diffie-Hellman assumption: according to the assumption $e(h_{\bar{\mathsf{id}^*}}, h_{n+1-\bar{\mathsf{id}^*}})^r = e\left(g,g\right)^{z^{n+1}r} \approx e\left(g,g\right)^s$ (given $\{h_i = g^{z^i}\}_{i \in [2n] \setminus \{n+1\}}$ and $\mathsf{ct}_2 = g^r$). Therefore, for case 1 ct_0, ct_2 are identical to the ones in Game_0 and ct_1 and ct_3 computationally indistinguishable, while for case 2 ct_0, ct_1, ct_2 identical and ct_3 computationally indistinguishable.

Game_2: Again, is the same to Game_1 except for step 3 and we have two cases:

1. Case 1: if $\mathcal{A}$ chooses to register id^* we set the ciphertext:

$$\mathsf{ct}_0 = C^{(k)}, \mathsf{ct}_1 = e(g,g)^w e\left(\prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}^*}}, g^r\right)$$
$$\mathsf{ct}_2 = g^r, \quad \mathsf{ct}_2 = e(g,g)^s m_b$$

for random $s \leftarrow \mathbb{Z}_p^*$ and $w \leftarrow \mathbb{Z}_p^*$.

2. Case 2: if $\mathcal{A}$ chooses not to register id^* then the ciphertext is identical to the one of Game_1, case 2:

$$\mathsf{ct}_0 = C^{(k)}, \mathsf{ct}_1 = e\left(\prod_{i \in \mathcal{D}^{(k)} \setminus \{\mathsf{id}^*\}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}^*}}, g^r\right)$$
$$\mathsf{ct}_2 = g^r, \quad \mathsf{ct}_3 = e(g,g)^s m_b$$

for a random $s \leftarrow \mathbb{Z}_p^*$.

In case 2, the distributions of Game_1 and Game_2 are identical. On the other hand, for case 1, the two distributions are computationally indistinguishable by the Bilinear Diffie-Hellman assumption: $e(g,g)^{s x_{\mathsf{id}^*}} \approx e(g,g)^w$. The proof is concluded by observing that in game 2, the $e(g,g)^s \cdot m_b$ is perfectly indistinguishable from a random element $M \in \mathbb{G}_T$, thus the probability of the adversary to win is exactly $1/2$. □

5 Extensions

We discuss a few extension to our main construction and we describe their efficiency tradeoffs.

5.1 More Efficient Updates

Here we describe a modified version of our scheme to make the updates more efficient and also decrease the size of the auxiliary information that is required to be stored on the KC side. To ease the understanding of the extension, we restate some of the notations compared to our construction in Figure 1. Let $t = \log(n)$; in our extension we are going to run t parallel instances of the scheme and we denote the public parameters of each instance by $\mathsf{pp} = \{\mathsf{pp}_1, \ldots, \mathsf{pp}_t\}$. Similar to the construction in Figure 1, each pp_i consists of commitments $(C_i^{(1)}, \ldots, C_i^{(B)})$. Throughout the construction, we are going to maintain the invariant that no identity id is registered under both pp_i and pp_j at any time, for any $i \neq j$. We will additionally maintain a global auxiliary information $\mathsf{aux} = (L_1, \ldots, L_N)$, along with an auxiliary variable $\boldsymbol{\alpha}_i^{(j)}$ associated with the commitment $C_i^{(j)}$.

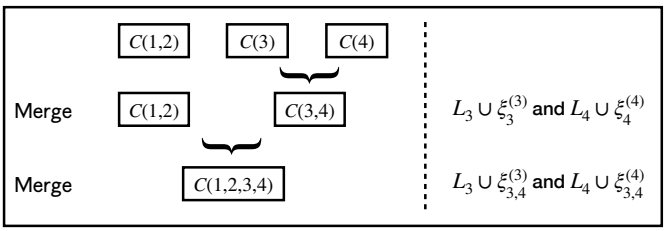

Figure 2: Merge of the commitments triggered by the registration of $\mathsf{id} = 4$. $C(I)$ denotes a commitment containing identities in the set I, whereas $\xi_I^{(i)}$ denotes the auxiliary information of the i-th party corresponding to $C(I)$.

To avoid notation clutter, in the following description we will only consider identities in the range $\{1, \ldots, n\}$. This allows us to only consider the first component $C_i^{(1)}$ of each commitment, and we consequently suppress the superscript denoting $C_i^{(1)} = C_i$ and $\boldsymbol{\alpha}_i^{(1)} = \boldsymbol{\alpha}_i$. This is without loss of generality, since other indices outside of this set are mapped to different commitments.

Registration. Whenever an identity $\mathsf{id} \in [n]$ with $(\mathsf{pk}_{\mathsf{id}}, \boldsymbol{\xi}_{\mathsf{id}})$ wants to register, the KC checks the correctness of $\boldsymbol{\xi}_{\mathsf{id}}$ (same as in Figure 1), parses the commitments $(C_1, \ldots, C_t)$, and finds the largest index i such that $C_i \neq 1$. Then it sets $C_{i+1} = \mathsf{pk}_{\mathsf{id}}$ and $\boldsymbol{\alpha}_{i+1} = \boldsymbol{\xi}_{\mathsf{id}}$. It then invokes the Merge subroutine to recursively update the public parameters and the auxiliary information. In a nutshell, the Merge subroutine maintains the invariant that the i-th commitment always has *at least* twice as many keys as the $(i+1)$-th commitment. A pictorial description is shown in Figure 2 and the pseudocode is shown below.

$\mathsf{Merge}^{[\mathsf{aux}, \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_{i+1}]}(C_i, C_{i+1})$:

- Let $J_i \subseteq [n]$ be the set of identities contained in C_i, and let $J_{i+1} \subseteq [n]$ be the set of identities contained in C_{i+1}.
- If $i = 0$ or $|J_i| \neq |J_{i+1}|$, then return $\perp$.
- Update $C_i = C_i \cdot C_{i+1}$ and $C_{i+1} = 1$.
- Parse $\mathsf{aux} = (L_1, \ldots, L_n)$.
- For all $j \in J_i$ parse $L_j = (\Lambda_{j,1}, \ldots, \Lambda_{j,\ell})$ and update $L_j \cup \{\Lambda_{j,\ell+1} = \boldsymbol{\alpha}_{i,j}\}$.
- For all $j \in J_{i+1}$ parse $L_j = (\Lambda_{j,1}, \ldots, \Lambda_{j,\ell})$ and update $L_j \cup \{\Lambda_{j,\ell+1} = \boldsymbol{\alpha}_{i+1,j}\}$.
- Compute $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i \circ \boldsymbol{\alpha}_{i+1}$, where $\circ$ denotes the component-wise product of the two vectors of group elements.
- Call $\mathsf{Merge}^{[\mathsf{aux}, \boldsymbol{\alpha}_{i-1}, \boldsymbol{\alpha}_i]}(C_{i-1}, C_i)$.

Encryption and Decryption. To encrypt a message m with respect to an identity $\mathsf{id} \in [n]$, the encrypter computes

$$\mathsf{ct} = \{\mathsf{ct}_i\}_{i \in [t]} = \\ \{C_i, e(C_i, h_{n+1-\mathsf{id}})^{r_i}, g^{r_i}, e(h_{\mathsf{id}}, h_{n+1-\mathsf{id}})^{r_i} m\}_{i \in [t]}$$

that is, the encryption is done in parallel with respect to all commitments $(C_1, \ldots, C_t)$. This way, as long as the decrypter knows the opening for *any* of the commitments, the decryption algorithm is successful. The security of the scheme can be shown by a standard hybrid argument.

Number of Updates. We claim that the total number of updates any identity id will receive will be at most $\log(n)$. The intuition behind this is quite simple, since updates must be issued only when two commitments are merged and therefore it suffices to establish that $t \leq \log(n)$. Since the size of a merged

commitment doubles, this operation can happen at most $\log(n)$ times before the number of identities loaded in a commitment equals n. More formally, let $|J_i|$ and $|J_{i+1}|$ be the number of identities in C_i and C_{i+1}, respectively. By definition, the Merge subroutine is only effective when $|J_i| = |J_{i+1}|$ (else it returns $\perp$), resulting in a commitment with $2|J_i|$ identities. Therefore for each $i \in [t]$ we have $|J_i| = 2^{t-i}$ at the end of the Merge subroutine. Since at most n identities are contained in each block, we have that

$$\sum_{i\in[t]} 2^{t-i} = \sum_{i\in[t]} |J_i| \leq n$$

which in turn implies that $t \leq \log(n)$.

Runtimes Analysis. In the following analysis, we ignore factors in the security parameter (such as the size of individual group elements) and we only consider the dependence of the algorithms with respect to the number of users. For the registration algorithm, the runtime of each recursive call is dominated by the component-product of $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_i$, which computes n group operations. Since there are at most t-many recursive calls, we can bound the total runtime by $O(n \cdot t) = O(\sqrt{N}\log\sqrt{N})$. The update algorithm sends at most t group elements to the receiver and thus its runtime is bounded by $O(\log\sqrt{N})$.

Space Analysis. Since we run t parallel instances of the public parameters, this variant introduces a multiplicative factor in $t \leq \log(n)$ and thus the size of the public parameter is bounded by $\sqrt{N}\log\sqrt{N}$. As for the auxiliary information, we have to bound the size of aux as well as the size of $\boldsymbol{\alpha}_i^{(j)}$. As argued above, each user receives at most $\log(n)$-many updates in its lifespan, so the size of aux is bounded by $N\log(n) = N\log\sqrt{N}$. On the other hand, the size of each $\boldsymbol{\alpha}_i^{(j)}$ is exactly $n-1$ and does not change throughout the execution of the protocol. In total, their size is $(n-1)Bt \leq N\log\sqrt{N}$, thus we can bound the total size of the auxiliary information by $O(N\log\sqrt{N})$.

5.2 Verifiability

In the following we discuss how to augment the construction in Section 4.1 with verifiability. Verifiable RBE [GV20] allows one to obtain a proof of registration (membership proof) or a proof of non-registration (non-membership proof) of an identity. In terms of efficiency, we require that both the proofs π and the runtime of the verification algorithm are *sublinear* in the size of the statement. The syntax of the definition of verfiable RBE is taken from [GV20].

Definition 4 (Verifiable RBE)**.** A verifiable RBE is an RBE scheme that is augmented with the PPT algorithms $(\mathsf{MProve}, \mathsf{MVerify}, \mathsf{NMProve}, \mathsf{NMVerify})$ defined as follows.

- $\mathsf{MProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \pi$: The membership proof algorithm MProve takes as input the common random sting crs, current public parameter pp, an identity id, and a public key pk and it outputs a membership proof π that certifies that id was registered in pp.
- $\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) \to 0/1$: The membership verification algorithm $\mathsf{MVerify}$ takes as input the common random sting crs, current public parameter pp, an identity id, a public key pk, and a proof π and it outputs a bit $\{0,1\}$ denoting acceptance or rejection.
- $\mathsf{NMProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}) \to \pi$: The membership proof algorithm $\mathsf{NMProve}$ takes as input the common random sting crs, current public parameter pp, and an identity id and it outputs a membership proof π that certifies that id was not registered in pp.
- $\mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) \to 0/1$: The non membership verification algorithm $\mathsf{NMVerify}$ takes as input the common random sting crs, current public parameter pp, an identity id, and a proof π and it outputs a bit $\{0,1\}$ denoting acceptance or rejection.

We refer the reader to [GV20] for definitions of completeness. In terms of efficiency, we require that both the proofs π and the runtime of the verification algorithm are *sublinear* in the size of the statement. Below we recall the formal definitions for soundness for both membership and non-membership proofs.

Definition 5 (Membership Soundness)**.** A verifiable RBE satisfies membership soundness if there exists a negligible function negl such that for all PPT adversaries $\mathcal{A}$ it holds that the advantage in the following experiment is negligibly close to 1/2:

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs})$
- $b \leftarrow \{0,1\}$
- $(\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{pk})$
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$
- The adversary wins if:
$$\mathcal{A}(\mathsf{ct}) = b \text{ and } \mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) = 1.$$

Definition 6 (Non Membership Soundness)**.** A verifiable RBE satisfies non membership soundness if there exists a negligible function negl such that for all PPT adversaries $\mathcal{A}$ it holds that the advantage in the following experiment is negligibly close to 1/2:

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
- $b \leftarrow \{0,1\}$
- $(\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs})$
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$
- The adversary wins if:
$$\mathcal{A}(\mathsf{ct}) = b \text{ and } \mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) = 1.$$

Algorithms. We describe the algorithms here. In fact we show that the opening Λ_{id} is already a valid membership/non-membership proof for the public key of id.

$\mathsf{MProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \pi$: Parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and let Λ_{id} be the last (most up to date) element of L_{id}. Return $\pi = \Lambda_{\mathsf{id}}$.

$\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) \to \{0,1\}$: Let $k = \lceil \frac{\mathsf{id}}{n} \rceil$ and parse $\mathsf{pp} = \left(C^{(1)}, \ldots, C^{(B)}\right)$. Accept if

$$e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right) \stackrel{?}{=} e\left(\pi, g\right) \cdot e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}}).$$

$\mathsf{NMProve}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}) \to \pi$: (Same as MProve)

$\mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) \to \{0,1\}$: Let $k = \lceil \frac{\mathsf{id}}{n} \rceil$ and parse $\mathsf{pp} = \left(C^{(1)}, \ldots, C^{(B)}\right)$. Accept if

$$e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right) \stackrel{?}{=} e\left(\pi, g\right).$$

In terms of efficiency, both membership and non-membership proofs consist of a single group element and the verification runtime is independent of N. Thus, the scheme satisfies verifier efficiency and proof succinctness.

Security Analysis. We state the main theorem in the following and we prove the two properties separately.

Theorem 2 (Soundness)**.** If the bilinear Diffie-Hellman and the n-Power Diffie-Hellman assumptions hold for $\mathcal{BG}$, then the scheme as described above satisfies membership and non-membership soundness.

Proof: Membership Soundness. We consider a sequence of games starting from the original Membership Soundness game of Definition 5. Each game induces a distribution for the output ciphertext; we will argue that the final distribution of ct is exactly $1/2$ and thus by a standard hybrid argument we will conclude that the initial Membership Soundness game is negligibly close to $1/2$. In the following $k = \lceil \frac{\mathsf{id}}{n} \rceil$.

$\underline{\mathsf{Game}_0}$: This is the original game of Membership Soundness. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, a key-pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs})$ according to the RBE algorithms and a random bit $b \leftarrow \{0,1\}$ and sends $(\mathsf{crs}, \mathsf{pk})$ to the adversary $\mathcal{A}$. The adversary on input $(\mathsf{crs}, \mathsf{pk})$ outputs $(\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1)$. Then a ciphertext of message m_b is computed $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$. Here:

$$\mathsf{ct} = \left(C^{(k)}, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^r, g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r m_b\right)$$

and the adversary is admissible if $\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) = 1$:

$$e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right) = e\left(\pi, g\right) \cdot e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}})$$

where pp is parsed as $\left(C^{(1)}, \ldots, C^{(B)}\right)$.[7]

$\underline{\mathsf{Game}_1}$: The game is the same to Game_0 except for the ciphertext computation. We change the way the second element of the ciphertext is computed. Namely, we set:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r) \cdot e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}})^r, g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r \cdot m_b\right)$$

Note that, since π correctly verifies, $\mathsf{MVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) = 1$, we have

$$e(\pi, g^r) \cdot e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}})^r = e(\pi, g)^r \cdot e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}})^r = e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right)^r$$

and thus the two distributions are identical.

$\underline{\mathsf{Game}_2}$: Here we change again the way the second element of the ciphertext is computed. Namely, we set:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r) \cdot e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^{r \cdot \mathsf{sk}}, g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r \cdot m_b\right).$$

To see that the two distributions are identical, note that in the RBE construction $\mathsf{pk} = h_{\bar{\mathsf{id}}}^{\mathsf{sk}}$ (note that pk, sk were honestly generated by the challenger) and thus:

$$e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^{r \cdot \mathsf{sk}} = e(h_{\bar{\mathsf{id}}}^{\mathsf{sk}}, h_{n+1-\bar{\mathsf{id}}})^r = e(\mathsf{pk}, h_{n+1-\bar{\mathsf{id}}})^r$$

$\underline{\mathsf{Game}_3}$: In this hybrid, we compute the ciphertext as:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r) \cdot e(g, g)^{s \cdot \mathsf{sk}}, g^r, e(g, g)^s \cdot m_b\right)$$

where $s \leftarrow \mathbb{Z}_p^*$ is a random field element. By the n-Power Diffie-Hellman assumption $e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r = e(g, g)^{z^{n+1} \cdot r} \approx e(g, g)^s$, therefore the distributions of ct in Game_2 and Game_3 are computationally indistinguishable.

$\underline{\mathsf{Game}_4}$: In this hybrid, we compute the ciphertext as:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r) \cdot e(g, g)^w, g^r, e(g, g)^s \cdot m_b\right)$$

where $s \leftarrow \mathbb{Z}_p^*$ and $w \leftarrow \mathbb{Z}_p^*$. Again, by the bilinear Diffie-Hellman assumption $e(g, g)^{s \cdot \mathsf{sk}} \approx e(g, g)^w$, thus the distributions induced by Game_3 and Game_4 are computationally indistinguishable. The proof is concluded by observing that $e(g, g)^s \cdot m_b$ is a uniformly distributed in $\mathbb{G}_T$, so the adversary has probability of success exactly $1/2$. □

Proof: Non-Membership Soundness. We consider the following sequence of hybrid games. In the following $k = \lceil \frac{\mathsf{id}}{n} \rceil$.

$\underline{\mathsf{Game}_0}$: This is the original game of Non-Membership Soundness. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ according to the RBE algorithm and a random bit $b \leftarrow \{0,1\}$ and sends crs to the adversary $\mathcal{A}$. The adversary on input crs outputs $(\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1)$. Then a ciphertext of message m_b is computed $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$. Here:

$$\mathsf{ct} = \left(C^{(k)}, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^r, g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r m_b\right)$$

and the adversary is admissible if $\mathsf{NMVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) = 1$:

$$e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right) = e\left(\pi, g\right)$$

where pp is parsed as $\left(C^{(1)}, \ldots, C^{(B)}\right)$.[7]

$\underline{\mathsf{Game}_1}$: Here we change the way the second element of the ciphertext is computed. Namely, we set:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r), g^r, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r \cdot m_b\right).$$

Observe that, since the proof π correctly verifies

$$e(\pi, g^r) = e(\pi, g)^r = e\left(C^{(k)}, h_{n+1-\bar{\mathsf{id}}}\right)^r$$

thus the two distributions are identical.

$\underline{\mathsf{Game}_2}$: In this hybrid, we compute the ciphertext as:

$$\mathsf{ct} = \left(C^{(k)}, e(\pi, g^r), g^r, e(g,g)^s \cdot m_b\right)$$

where $s \leftarrow \mathbb{Z}_p^*$. By the n-Powers Diffie-Hellman assumption $e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^r = e(g,g)^{z^{n+1} \cdot r} \approx e(g,g)^s$, therefore the distributions of Game_1 and Game_2 are computationally indistinguishable. The proof is concluded by observing that $e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^s \cdot m_b$ is uniformly distributed in $\mathbb{G}_T$, so the adversary has probability of success exactly $1/2$. □

5.3 Anonymity

We present a (slight) variant of our scheme of Section 4.1 that is additionally anonymous, namely, where the ciphertexts hide the identity of the decrypter. Intuitively, there are two reasons why our original scheme is not anonymous.

Firstly, in order for the decrypter to identify if her update information u is outdated (and thus needs to query the KC for Upd), we include the up-to-date $C^{(k)}$ in the ciphertext. Then the decrypter can directly check if Λ_ℓ verifies with respect to $C^{(k)}$. This clearly (partially) breaks anonymity, since the 'block' k in which the user lies is leaked. We resolve this by including in the ciphertext a hiding commitment of the message m instead, $c = \mathsf{Com}(m; o)$, together with an encryption of the randomness o, in place of $C^{(k)}$. If u is up-to-date then the two ciphertexts are decrypted normally and c opens to m with randomness o, $c = \mathsf{Com}(m; o)$. If not, then u is outdated and the user calls GetUpd.

Secondly, an adversary controlling all the users registered in a cluster k^* can, by using their secret keys and g^r, identify if the part $e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^r$ of a ciphertext for identity id is containing $C^{(k^*)}$, in case id is not registered yet. Intuitively this is because the adversary knows *all* the secret keys in $C^{(k^*)} = \prod_i h_i^{\mathsf{sk}_i}$ (since id hasn't registered yet). To resolve this we initialize all $C^{(i)}$ with a dummy public key in an extra position $n+1$, $R = h_{n+1}^t$, generated in the setup phase. Being generated in the setup phase, no adversary knows the corresponding secret.

Definition. Here we can formally define the anonymity notion for RBE schemes, we note that the definition is taken from [GHM+19] with some minor changes.

[7] If $k > B$ the adversary is considered non-admissible.

Definition 7 (Anonymous RBE)**.** An anonymous RBE scheme has the same syntax as that of an RBE scheme, with PPT algorithms $(\mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$. It satisfies the properties of completeness, compactness and efficiency as a RBE scheme and has the following stronger notion of security: For any interactive PPT adversary $\mathcal{A}$, consider the game $\mathsf{EXP}^{\mathtt{Anon}}_{\mathcal{A}}(\lambda)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

1. **Initialization.** $\mathcal{C}$ sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathcal{D} = \emptyset$, $\mathsf{id}_0 = \bot$, $\mathsf{id}_1 = \bot$, $t = 0$, $\mathsf{crs} \leftarrow U_{\mathsf{poly}(\lambda)}$ and sends the sampled crs to $\mathcal{A}$.
2. Until $\mathcal{A}$ continues (which is at most $\mathsf{poly}(\lambda)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ can perform exactly one of the following actions.
   (a) **Registering new (non-target) identity.** $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}$ and pk to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by getting $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}, \mathsf{pk}))$ and lets $\mathcal{D} = \mathcal{D} \cup \{\mathsf{id}\}$.
   (b) **Registering new target identity pair.** If id_0 or id_1 was chosen by $\mathcal{A}$ already (i.e., $\mathsf{id}_0 \neq \bot$ or $\mathsf{id}_1 \neq \bot$), skip this step. Otherwise, $\mathcal{A}$ sends challenges $\mathsf{id}_0, \mathsf{id}_1 \notin \mathcal{D}$ to $\mathcal{C}$. $\mathcal{C}$ first samples $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^\lambda)$, registers id_0 by setting $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}_0, \mathsf{pk}_0))$ and then samples $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}(1^\lambda)$, registers id_1 by setting $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}_1, \mathsf{pk}_1))$. Next, $\mathcal{C}$ lets $\mathcal{D} = \mathcal{D} \cup \{\mathsf{id}_0, \mathsf{id}_1\}$ and sends $\mathsf{pk}_0, \mathsf{pk}_1$ to $\mathcal{A}$.
3. **Encrypting for the challenge identity.** $\mathcal{A}$ sends some $(\mathsf{id}^*_0, \mathsf{id}^*_1) \notin \mathcal{D} \setminus \{\mathsf{id}_0, \mathsf{id}_1\}$ and two messages (m_0, m_1) and $\mathcal{C}$ generates $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}_b, m_b)$, where $b \leftarrow \{0, 1\}$ is a random bit, and sends ct to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

We call an RBE scheme anonymous if there exists a negligible function negl such that for all PPT adversaries $\mathcal{A}$ it holds that $\left|1/2 - \Pr[\mathcal{A} \text{ wins } \mathsf{EXP}^{\mathtt{Anon}}_{\mathcal{A}}(\lambda)]\right| \leq \mathsf{negl}(\lambda)$.

Construction. Our full construction is on Figure 3. For generality we use Com as a black-box; the reader can think of it as an ElGamal commitment.[8] For ease of presentation we highlight the differences with the original construction with blue color.

Correctness. Correctness is similar to the one of the original construction. We, additionally, argue that, under the Binding property of the commitment scheme, $c = \mathsf{Com}(m'; o')$ iff Λ_ℓ is up-to-date: if it is up-to-date then $m = m'$ and $r = r'$, if not then $m \neq m'$ and $r = r'$ (unless with negligible probability) and $c = \mathsf{Com}(m; r) \neq \mathsf{Com}(m'; r')$ (from the Binding property of Com).

Anonymity. We state and prove the following theorem.

Theorem 3 (Anonymity)**.** If the bilinear Diffie-Hellman and the n-Power Diffie-Hellman assumptions hold for $\mathcal{BG}$, then the scheme as described above satisfies anonymity.

Proof. The proof follows along the lines of the argument of our main theorem, so here we provide a sketch, only highlighting the differences in the game-hops.

$\underline{\mathsf{Game}_0}$: Is the original game of Anonymous RBE security. Here:

$$\mathsf{ct} = \left(\mathsf{Com}(m_b; o), e(C^{(k)}, h_{n+1-\mathsf{id}^*_b})^{r_1}, g^{r_1}, e(h_{\mathsf{id}^*_b}, h_{n+1-\mathsf{id}^*_b})^{r_1} m_b, , e(C^{(k)}, h_{n+1-\mathsf{id}^*_b})^{r_2}, g^{r_2}, e(h_{\mathsf{id}^*_b}, h_{n+1-\mathsf{id}^*_b})^{r_2} o\right)$$

$\underline{\mathsf{Game}_1}$: As a first step we switch to a game with a random value committed instead of m_b.

$$\mathsf{ct} = \left(\mathsf{Com}(m^*; o^*), e(C^{(k)}, h_{n+1-\mathsf{id}^*_b})^{r_1}, g^{r_1}, e(h_{\mathsf{id}^*_b}, h_{n+1-\mathsf{id}^*_b})^{r_1} m_b, , e(C^{(k)}, h_{n+1-\mathsf{id}^*_b})^{r_2}, g^{r_2}, e(h_{\mathsf{id}^*_b}, h_{n+1-\mathsf{id}^*_b})^{r_2} o\right)$$

[8] In case of domain mismatch we assume a collision-resistant hash function being applied to m and/or o.

$\mathsf{Setup}(1^\lambda, n) \to \mathsf{crs}$: generate a bilinear group $\mathsf{bg} := (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ and sample $z \leftarrow \mathbb{Z}_p$. Sets $h_i = g^{z^i}$ for each $i \in [2(n+1)]$. Furthermore, sample $t \leftarrow \mathbb{Z}_p^*$ and compute $R = (h_{n+1})^t$ and $\Xi = ((h_{2n+2})^t, \ldots, (h_{n+3})^t, \emptyset)$. Output

$$\mathsf{crs} = \left(\mathsf{bg}, \{h_i\}_{i \in [2(n+1)] \setminus \{n+2\}}, R, \{\Xi_i\}_{i \in [n]}\right).$$

$\mathsf{Gen}(\mathsf{crs}, \mathsf{id}) \to (\mathsf{pk}, \mathsf{sk}, \boldsymbol{\xi})$: If $\mathsf{id} \notin [N]$ output $\perp$. Otherwise, sample $x_{\mathsf{id}} \leftarrow \mathbb{Z}_p$, compute $\bar{\mathsf{id}} = \mathsf{id} \mod n$ and set

$$\mathsf{pk} = (h_{\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \qquad \mathsf{sk} = x_{\mathsf{id}}, \qquad \boldsymbol{\xi} = \left((h_{\bar{\mathsf{id}}+n})^{x_{\mathsf{id}}}, \ldots, (h_{2+n})^{x_{\mathsf{id}}}, \emptyset, (h_n)^{x_{\mathsf{id}}}, \ldots, (h_{\bar{\mathsf{id}}+1})^{x_{\mathsf{id}}}\right)$$

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \boldsymbol{\xi}) \to \mathsf{pp}'$: The key curator proceeds as follows:

- If it is the first identity to be registered then initialize $\mathsf{pp} = (R, \ldots, R) \in \mathbb{G}^B$ and $\mathsf{aux} = (\{\Xi_1\}, \ldots, \{\Xi_n\}, \ldots, \{\Xi_1\}, \ldots \{\Xi_n\}) \in \mathbb{G}^N$.
- If $\mathsf{id} \notin [N]$ output $\perp$.
- Check the correctness of $\boldsymbol{\xi}$:

  $$e(\mathsf{pk}, h_n) = e(\boldsymbol{\xi}_1, g) = \cdots = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}-1}, h_{\bar{\mathsf{id}}-2}) = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}+1}, h_{\bar{\mathsf{id}}}) = \ldots = e(\boldsymbol{\xi}_n, h_{n-1}).$$

  If the above doesn't hold output $\perp$.
- Otherwise, parse $\mathsf{pp} = \left(C^{(1)}, \ldots, C^{(B)}\right)$, compute $k = \lceil \frac{\mathsf{id}}{n} \rceil$ and set $C'^{(k)} \leftarrow C^{(k)} \cdot \mathsf{pk}$, the rest remain the same $C'^{(j)} = C^{(j)}$ for each $j \in [B] \setminus \{k\}$. Output

  $$\mathsf{pp}' = (C'^{(1)}, \ldots, C'^{(B)})$$

- Parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and for each $j \in \{(k-1)n+1, \ldots, (k-1)n+n\} \setminus \{\mathsf{id}\}$ parse $L_j = \{\Lambda_{j,1}, \ldots, \Lambda_{j,\ell}\}$. Set $L_j = L_j \cup \{\Lambda_{j,\ell+1} = \Lambda_{j,\ell} \cdot \boldsymbol{\xi}_j\}$, while the rest remain the same.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$: On input $m \in \mathbb{G}_T$, let $k = \lceil \frac{\mathsf{id}}{n} \rceil$. Sample $r \leftarrow \mathbb{Z}_p^*$, $o \leftarrow \mathcal{O}$ and compute:

$$\begin{aligned}\mathsf{ct} &= (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5, \mathsf{ct}_6) = \\ &= \left(\mathsf{Com}(m; o), e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^{r_1}, g^{r_1}, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^{r_1} m, e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}})^{r_2}, g^{r_2}, e(h_{\bar{\mathsf{id}}}, h_{n+1-\bar{\mathsf{id}}})^{r_2} o\right)\end{aligned}$$

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$: parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and output L_{id}.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to m$ **or** GetUpd : Parse $\mathsf{u} = \{\Lambda_1, \ldots, \Lambda_\ell\}$ and for each $i \in [\ell]$ compute

$$m'_i = \frac{\mathsf{ct}_3}{\left(e(\Lambda_i, \mathsf{ct}_2)^{-1} \cdot \mathsf{ct}_1\right)^{\mathsf{sk}^{-1}}} \qquad \text{and} \qquad o'_i = \frac{\mathsf{ct}_6}{\left(e(\Lambda_i, \mathsf{ct}_5)^{-1} \cdot \mathsf{ct}_4\right)^{\mathsf{sk}^{-1}}}$$

If the exist no i such that $\mathsf{Com}(m'_i; o'_i) \neq \mathsf{ct}_0$ then output GetUpd, otherwise output m'_i.

Figure 3: Our Anonymous RBE construction variant.

for random $m^* \leftarrow \mathbb{G}_T$, $o^* \leftarrow \mathcal{O}$. From the hiding property of the commitment scheme Com, the games 0 and 1 are indistinguishable.

From here the proof in the case 1, the case where $\mathcal{A}$ chooses to register id_b^*, is identical to the original proof. Hence we omit this case and focus on case 2, where the adversary $\mathcal{A}$ chooses not to register id_b^*.

$\underline{\mathsf{Game}_2}$: Here we compute:

$$\begin{aligned}
&\mathsf{ct}_0 = \mathsf{Com}(m^*; o^*),\\
&\mathsf{ct}_1 = e(g,g)^{s_1 \cdot t} e\left(\prod_{i \in \mathcal{D}^{(k)}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}_b^*}, g^{r_1}\right), \mathsf{ct}_2 = g^{r_1},\\
&\mathsf{ct}_3 = e(g,g)^{s_1} m_b,\\
&\mathsf{ct}_4 = e(g,g)^{s_2 \cdot t} e\left(\prod_{i \in \mathcal{D}^{(k)}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}_b^*}, g^{r_2}\right),\\
&\mathsf{ct}_5 = g^{r_2}, \mathsf{ct}_6 = e(g,g)^{s_2} o
\end{aligned}$$

for random $m^* \leftarrow \mathbb{G}_T$, $o^* \leftarrow \mathcal{O}$, $s_1, s_2 \leftarrow \mathbb{Z}_p^*$. The distributions induced by Game_0 and Game_1 are computationally indistinguishable assuming the n-power Diffie-Hellman assumption. To see why this holds observe that $C^{(k)} = (h_{n+1})^t \prod_{i \in \mathcal{D}^{(k)}} (h_{\bar{i}})^{x_i}$ from which we get that $e(C^{(k)}, h_{n+1-\bar{\mathsf{id}}_b^*})^r = e(g,g)^{z^{n+2} r t} \cdot e\left(\prod_{i \in \mathcal{D}^{(k)}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}_b^*}, g^r\right)$ (see the original proof for a similar argument) and $e(h_{\bar{\mathsf{id}}_b^*}, h_{n+1-\bar{\mathsf{id}}_b^*})^r = e(g,g)^{z^{n+2} r}$.

$\underline{\mathsf{Game}_3}$: Here we compute:

$$\begin{aligned}
&\mathsf{ct}_0 = \mathsf{Com}(m^*; o^*),\\
&\mathsf{ct}_1 = e(g,g)^{w_1} e\left(\prod_{i \in \mathcal{D}^{(k)}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}_b^*}, g^{r_1}\right), \mathsf{ct}_2 = g^{r_1}\\
&\mathsf{ct}_3 = e(g,g)^{s_1} m_b,\\
&\mathsf{ct}_4 = e(g,g)^{w_2} e\left(\prod_{i \in \mathcal{D}^{(k)}} \boldsymbol{\xi}_{i,\bar{\mathsf{id}}_b^*}, g^{r_2}\right)\\
&\mathsf{ct}_5 = g^{r_2}, \quad \mathsf{ct}_6 = e(g,g)^{s_2} o
\end{aligned}$$

for random $m^* \leftarrow \mathbb{G}_T$, $o^* \leftarrow \mathcal{O}$, $s_1, s_2 \leftarrow \mathbb{Z}_p^*$, $w_1, w_2 \leftarrow \mathbb{Z}_p^*$. The distributions of games 2 and 3 are computationally indistinguishable by the Bilinear Diffie-Hellman assumption. To conclude the proof note that the distribution of Game_3 is perfectly indistinguishable from a random one. □

5.4 CCA Security

The standard CCA security notion for encryption schemes captures the fact that an adversary may be able to obtain decryptions for ciphertexts of her choice and demands that even in this case she should not be able distinguish between encrypted messages. In the following we state the notion of security under chosen ciphertext attack (CCA) for RBE.

Definition 8 (CCA Security of RBE)**.** For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathsf{EXP}_{\mathcal{A}}^{\mathsf{CCA}}(\lambda)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$.

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

2. Until $\mathcal{A}$ continues (which is at most $\mathsf{poly}(\lambda)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to be performed.

   (a) **Registering new (non-target) identity**. $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}$ and pk to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.

(b) **Registering the target identity.** If $\mathsf{id}^* \neq \perp$, skip this step. Otherwise, $\mathcal{A}$ sends some $\mathsf{id}^* \notin \mathcal{D}$ to $\mathcal{C}$. $\mathcal{C}$ then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(\mathsf{crs})$, updates $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, and sends pk^* to $\mathcal{A}$.

(c) **Decryption by target identity** If $\mathsf{id}^* = \perp$, skip this step. Otherwise, $\mathcal{A}$ sends some ciphertext ct to $\mathcal{C}$. $\mathcal{C}$ then lets $m = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct})$. If $m = \mathsf{GetUpd}$, then $\mathcal{C}$ obtains the update $\mathsf{u}^* = \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ and then returns $\mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}^*, \mathsf{ct}_j)$.

3. **Encrypting for the target identity.** $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D} \setminus \{\mathsf{id}^*\}$ and two messages (m_0, m_1) and $\mathcal{C}$ generates $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$, where $b \leftarrow \{0,1\}$ is a random bit, and sends ct^* to $\mathcal{A}$.

4. The adversary $\mathcal{A}$ can continue querying the decryption oracle on ciphertexts not equal to ct^*. At some point $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.

We call an RBE scheme secure if there exists a negligible function negl such that for all PPT adversaries $\mathcal{A}$ it holds that $\left|1/2 - \Pr[\mathcal{A}\,\text{wins}\,\mathsf{EXP}^{\mathsf{CCA}}_{\mathcal{A}}(\lambda)]\right| \leq \mathsf{negl}(\lambda)$.

We sketch how our scheme can be transformed into a CCA-secure one. First, we observe that the procedure to determine whether the user's opening is up to date depends only on $h^{\mathsf{sk}}_{\mathsf{id}} = \mathsf{pk}$, which is a public information. It follows that this check is publicly computable and therefore does not leak any further information about the secret key. It remains to show how to modify the scheme to answer decryption queries. There are many generic transformations available in the literature, and the most efficient is the Fujisaki-Okamoto transformation [FO99], where the encryption algorithm is defined as

$$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, r; \mathsf{H}_1(r, m)), \mathsf{H}_2(r) \oplus m$$

where $(\mathsf{H}_1, \mathsf{H}_2)$ are two hash functions modeled as random oracles and r is a random string. Since this is a standard transformation, we omit the formal analysis and we refer the reader to [FO99] for further details.

5.5 Efficient RBE from the BDH assumption

We show an efficient construction of RBE secure under the plain Bilinear Diffie-Hellman assumption, i.e., not based on a q-type assumption. The construction is in the same spirit as the one of Section 4.1, but is instead based on the [CF13] vector commitment (which is secure under the standard CDH assumption). The scheme is slightly less efficient than our main construction. The overhead comes from the fact that the [CF13] VC has quadratic parameters, thus (after applying the square root trick) the parameters are still linear on the number of users. Fortunately, the linear-sized part of the parameters (denoted as crs) is not needed to be stored by the users, but solely by the KC.[9]

We denote the parameters that the user should store by $\mathsf{crs}_{\text{user}}$, and as will be apparent in the construction they have size $O(n)$ (instead of $O(N)$). The scheme is described in Figure 4.

Security For lack of space we omit a full formal proof. To see why the scheme is secure under the Bilinear Diffie-Hellman (BDH) assumption observe that $\mathsf{ct}_3 = e(h_i, h_i)^r m = e(g,g)^{r \cdot z^2} m$. Under the BDH assumption, $e(g,g)^{rz^2}$ is indistinguishable from $e(g,g)^s$, for a random $s \leftarrow \mathbb{Z}_p^*$.[10]

6 Implementation and Benchmarks

We implemented a Python prototype[11] of both efficient-update and basic RBE construction described in Section 5.1 and Section 4.1. We used asymmetric pairings instead of symmetric pairings in our prototype. To accommodate asymmetric pairings, the construction is modified slightly to have two copies of the CRS

[9] Each user needs to download the linear-sized parameters once at the registration phase and then delete it.

[10] In fact, this is a square-variant of the BDH assumption, but can be reduced to the plain BDH. Analogous reductions for CDH and its square-variant can be found in the literature [BDZ03, MW96].

[11] `https://anonymous.4open.science/r/Efficient-RBE-2/`

$\mathsf{Setup}(1^\lambda, n) \to (\mathsf{crs}, \mathsf{crs}_{KC})$: generate a bilinear group $\mathsf{bg} := (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{BG}(1^\lambda)$ and sample $\{z_i\}_{i\in[n]} \leftarrow \mathbb{Z}_p^*$. Set $h_i = g^{z_i}$ for each $i \in [n]$, $H_{i,j} = g^{z_i z_j}$ for each $i, j \in [n]$ and output

$$\mathsf{crs} = \left(\mathsf{bg}, \{h_i\}_{i\in[n]}, \{H_{i,j}\}_{i,j\in[n],i\neq j}\right) \quad \text{and} \quad \mathsf{crs}_{\text{user}} = \left(\mathsf{bg}, \{h_i\}_{i\in[n]}\right)$$

$\mathsf{Gen}(\mathsf{crs}, \mathsf{id}) \to (\mathsf{pk}, \mathsf{sk}, \boldsymbol{\xi})$: If $\mathsf{id} \notin [N]$ output $\perp$. Otherwise, sample $x_{\mathsf{id}} \leftarrow \mathbb{Z}_p$, compute $\bar{\mathsf{id}} = \mathsf{id} \mod n$ and set

$$\mathsf{pk} = (h_{\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \qquad \mathsf{sk} = x_{\mathsf{id}}, \qquad \boldsymbol{\xi} = \left((H_{1,\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \ldots, (H_{\bar{\mathsf{id}}-1,\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \emptyset, (H_{\bar{\mathsf{id}}+1,\bar{\mathsf{id}}})^{x_{\mathsf{id}}}, \ldots, (H_{n,\bar{\mathsf{id}}})^{x_{\mathsf{id}}}\right)$$

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \boldsymbol{\xi}) \to \mathsf{pp}'$: The key curator proceeds as follows:

- If it is the first identity to be registered then initialize $\mathsf{pp} = (1, \ldots, 1) \in \mathbb{G}^B$ and $\mathsf{aux} = (\{1\}, \ldots, \{1\}) \in \mathbb{G}^N$.
- If $\mathsf{id} \notin [N]$ output $\perp$.
- Check the correctness of $\boldsymbol{\xi}$:

  $$e(\mathsf{pk}, h_1) = e(\boldsymbol{\xi}_1, g), \ldots, e(\mathsf{pk}, h_{\bar{\mathsf{id}}-1}) = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}-1}, g), e(\mathsf{pk}, h_{\bar{\mathsf{id}}+1}) = e(\boldsymbol{\xi}_{\bar{\mathsf{id}}+1}, g), \ldots, e(\mathsf{pk}, h_n) = e(\boldsymbol{\xi}_n, g)$$

  where $\bar{\mathsf{id}} = \mathsf{id} \mod n$. If the above doesn't hold output $\perp$.
- Otherwise, parse $\mathsf{pp} = \left(C^{(1)}, \ldots, C^{(B)}\right)$, compute $k = \lceil \frac{\mathsf{id}}{n} \rceil$ and set $C'^{(k)} \leftarrow C^{(k)} \cdot \mathsf{pk}$, the rest remain the same $C'^{(j)} = C^{(j)}$ for each $j \in [B] \setminus \{k\}$. Output

  $$\mathsf{pp}' = (C'^{(1)}, \ldots, C'^{(B)})$$

- Parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and for each $j \in \{(k-1)n + 1, \ldots, (k-1)n + n\} \setminus \{\mathsf{id}\}$ parse $L_j = \{\Lambda_{j,1}, \ldots, \Lambda_{j,\ell}\}$. Set $L_j = L_j \cup \{\Lambda_{j,\ell+1} = \Lambda_{j,\ell} \cdot \boldsymbol{\xi}_j\}$, while the rest remain the same.

$\mathsf{Enc}(\mathsf{crs}_{\mathbf{user}}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$: On input $m \in \mathbb{G}_T$, let $\bar{\mathsf{id}} = \mathsf{id} \mod n$ and $k = \lceil \frac{\mathsf{id}}{n} \rceil$. Sample $r \leftarrow \mathbb{Z}_p$ and compute:

$$\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3) = \left(C^{(k)}, e(C^{(k)}, h_i)^r, g^r, e(h_i, h_i)^r \cdot m\right).$$

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$: parse $\mathsf{aux} = (L_1, \ldots, L_N)$ and output L_{id}.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to m$ **or** GetUpd : Parse $\mathsf{u} = \{\Lambda_1, \ldots, \Lambda_\ell\}$, if there exists no $i \in [\ell]$ such that $e(\mathsf{ct}_0, h_i) \neq e(\mathsf{u}, g) \cdot e((h_i)^{\mathsf{sk}}, h_i)$ then output GetUpd. Otherwise return

$$\frac{\mathsf{ct}_3}{\left(e(\Lambda_i, \mathsf{ct}_2)^{-1} \cdot \mathsf{ct}_1\right)^{\mathsf{sk}^{-1}}}.$$

Figure 4: Our RBE construction from the Bilinear Diffie-Hellman assumption.

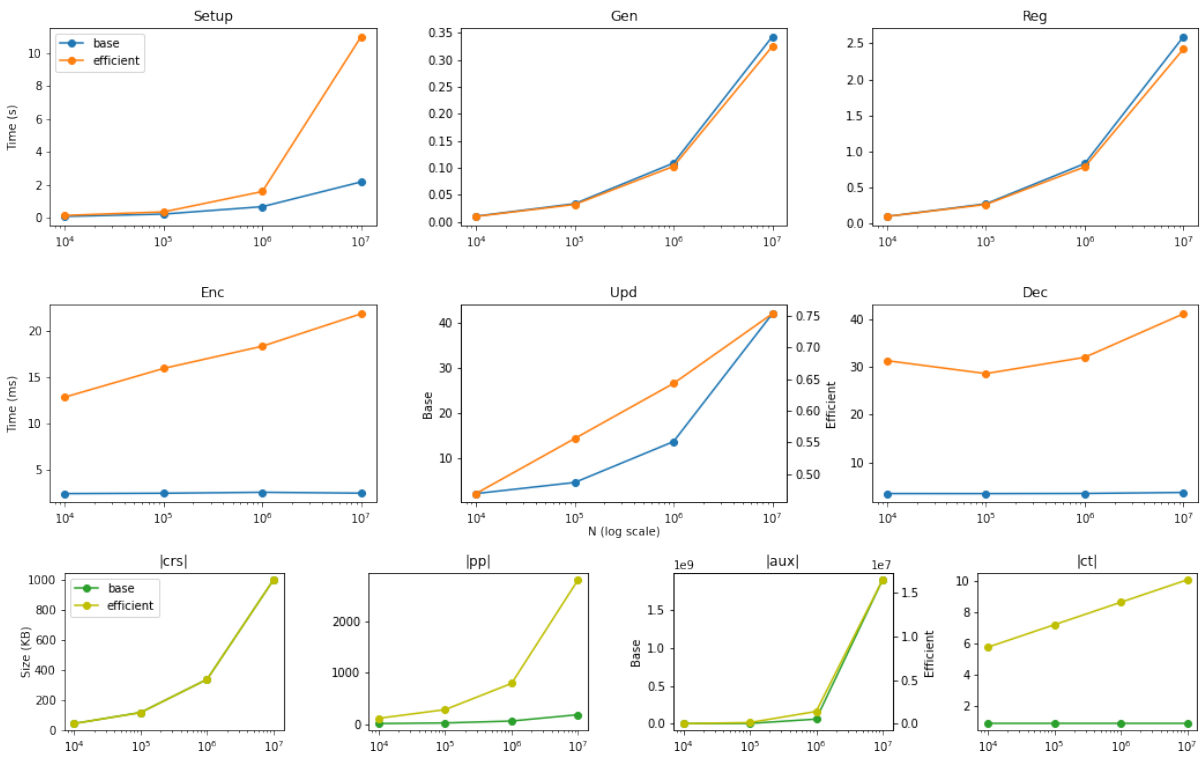

Figure 5: Benchmarks for our basic and efficient constructions. The sizes of crs, pp, aux are given are calculated assuming the full system capacity N has been registered. The size of ct is given for the worst case (see discussion in text); the sizes of pk, sk are constant at $49\mathsf{B}$ and $32\mathsf{B}$, respectively.

(one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$); all other elements (commitments and auxiliary information) are left in $\mathbb{G}_1$, and ct_2 is set to g_2^r (where g_2 is the generator of $\mathbb{G}_2$). For the choice of pairings, we used $\mathsf{BLS\text{-}381}$ pairings from [AGM+] via the $\mathsf{petrelic}$ [pet] Python wrapper; each element in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ can be represented with $49\mathsf{B}$, $97\mathsf{B}$, and $384\mathsf{B}$, respectively.

Storage. In the base RBE implementation, the size of $\mathsf{pk}, \mathsf{sk}, \mathsf{ct}$ for encrypting a single group element from $\mathbb{G}_T$ is constant: $49\mathsf{B}, 32\mathsf{B}, 866\mathsf{B}$ respectively, and size of each update that users will receive during the decryption is also constant and it is $49\mathsf{B}$. In the RBE with efficient updates, the size of pk, sk is constant and it is $49\mathsf{B}$ and $32\mathsf{B}$. However, the size of ct for the encryption of a single $\mathbb{G}_T$ element will increase by a factor of $\log \sqrt{N}$ in the worst case, which in the case of 10 million users is a total $10\mathsf{KB}$. Throughout the lifetime of the system each user will have to receive a total of $\sqrt{N}$ (base construction) or $\log \sqrt{N}$ (efficient updates) updates, each of size $49\mathsf{B}$.

Benchmarks. The run-time of each algorithm and size of the parameters are plotted for increasing values of N in Figure 5. The benchmarking for an RBE system of N users is performed in the following way: First the crs is generated for the parameter N and $n = \sqrt{N}$ identities $\mathsf{id}_1, \ldots, \mathsf{id}_n$ where $\mathsf{id}_i \in [n]$ for all $i \in [n]$ are registered in a random order. Notice that all these parties will register under a single commitment in the public parameter, and without loss of generality we take the first commitment. We also remark that since the registration/encryption/decryption of each commitment can be done in parallel and independently of the other commitments, registering all the users in the same commitment is the worst case in terms of runtimes. The registration times are taken as an average over n identities, whereas the encryption and decryption times are averages of over 100 runs of the respective algorithms. In the base construction, we allow the encrypter

	Time (s)						Size		
N	Setup	Gen	Reg	Enc	Upd	Dec	crs	pp	aux
10K	0.085	0.011	0.094	0.002	0.002	0.004	41KB	16KB	57MB
100K	0.228	0.034	0.270	0.002	0.005	0.004	115KB	28 KB	1.9GB
1M	0.678	0.109	0.829	0.003	0.014	0.004	336KB	65 KB	59GB
10M	2.181	0.343	2.595	0.002	0.042	0.004	1.0MB	188KB	1.9TB

	Time (s)						Size		
N	Setup	Gen	Reg	Enc	Upd	Dec	crs	pp	aux
10K	0.147	0.011	0.098	0.014	0.001	0.033	41KB	118KB	9.8MB
100K	0.36	0.034	0.273	0.017	0.001	0.033	115KB	286KB	124MB
1M	1.59	0.107	0.821	0.019	0.001	0.037	336KB	794KB	1.4GB
10M	10.99	0.340	2.547	0.022	0.001	0.043	1.04MB	2.8MB	16.5GB

Table 2: Benchmarks for our basic construction (top) and the construction with efficient updates (bottom).

to include the time of the encryption in the ciphertext, which allows the decryption algorithm to directly use the appropriate opening and decrypt the ciphertext in constant time. The measurements are taken on a 16-core Intel i7 10700K 3.8GHz with 128GB of RAM.

In Table 2 we report the measurements for our benchmarks plotted in Section 6.

7 Conclusions and Open Problems

In this work, we proposed a new method for constructing RBE. The resulting RBE scheme is black-box, practically efficient, and its security is based on standard assumptions over bilinear groups. For the first time, we show that RBE can be implemented in systems scaling to millions of users. Our work opens the doors to the practical deployment of RBE, and we hope it will inspire researchers to explore new applications and build new systems based on RBE. On the theoretical side, our work leaves open the problem of matching the asymptotics of previous RBE constructions [GHMR18, GHM+19], without resorting to non black-box use of cryptography. Another interesting direction is to construct an efficient RBE with a public-coin (a.k.a. transparent) setup.

Acknowledgments. The authors wish to thank Peter Schwabe for his valuable suggestions on the implementation of our prototype. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy - EXC 2092 CASA – 390781972.

References

[AGM+] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[AP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, Heidelberg, November / December 2003.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586. Springer, Heidelberg, August 2019.

[BBM+20] R Barnes, B Beurdouche, J Millican, E Omara, K Cohn-Gordon, and R Robert. The messaging layer security (mls) protocol draft-ietf-mls-protocol-09. Technical report, Internet-draft, September, 2020.

[BDZ03] Feng Bao, Robert H Deng, and Huafei Zhu. Variations of diffie-hellman problem. In *International conference on information and communications security*, pages 301–312. Springer, 2003.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Heidelberg, August 2001.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, Heidelberg, August 2005.

[Bow18] Sean Bowe. Completion of the sapling mpc. `https://electriccoin.co/blog/completion-of-the-sapling-mpc/`, 2018.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65. Springer, Heidelberg, August 2017.

[CES21] 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.

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 55–72. Springer, Heidelberg, February / March 2013.

[CFG+20] Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 3–35. Springer, Heidelberg, December 2020.

[Cho09] Sherman S. M. Chow. Removing escrow from identity-based encryption. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 256–276. Springer, Heidelberg, March 2009.

[CHSS02] Liqun Chen, Keith Harrison, David Soldera, and Nigel P Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *International Conference on Infrastructure Security*, pages 260–275. Springer, 2002.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding*, pages 360–363, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569. Springer, Heidelberg, August 2017.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, Heidelberg, August 1999.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, Heidelberg, May / June 2006.

[GHM+19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93. Springer, Heidelberg, April 2019.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718. Springer, Heidelberg, November 2018.

[GLSW08] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 427–436. ACM Press, October 2008.

[Goy07] Vipul Goyal. Reducing trust in the PKG in identity based cryptosystems. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 430–447. Springer, Heidelberg, August 2007.

[GRWZ20] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 2007–2023. ACM Press, November 2020.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651. Springer, Heidelberg, August 2020.

[KG10] Aniket Kate and Ian Goldberg. Distributed private-key generators for identity-based cryptography. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 436–453. Springer, Heidelberg, September 2010.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Secure key issuing in id-based cryptography. In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation-Volume 32*, pages 69–74. Citeseer, 2004.

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 530–560. Springer, Heidelberg, August 2019.

[LP20] Helger Lipmaa and Kateryna Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 686–716. Springer, Heidelberg, December 2020.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 499–517. Springer, Heidelberg, February 2010.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, Heidelberg, August 1996.

[pet] petrelic is a Python wrapper around RELIC. `https://github.com/spring-epfl/petrelic`.

[PS08] Kenneth G. Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 354–375. Springer, Heidelberg, September 2008.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. `https://ia.cr/2015/1162`.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, Heidelberg, August 1984.

[TAB+20] Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Heidelberg, September 2020.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, Heidelberg, May 2005.

[WQT18] Quanyun Wei, Fang Qi, and Zhe Tang. Remove key escrow from the bf and gentry identity-based encryption with non-interactive key generation. *Telecommunication Systems*, 69(2):253–262, 2018.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.