Registration-Based Encryption from Standard Assumptions

Sanjam Garg* Mohammad Hajiabadi† Mohammad Mahmoody‡
Ahmadreza Rahimi§ Sruthi Sekar¶

October 23, 2018

Abstract

The notion of *Registration-Based Encryption* (RBE) was recently introduced by Garg, Hajiabadi, Mahmoody, and Rahimi [TCC'18] with the goal of removing the private-key generator (PKG) from IBE. Specifically, RBE allows encrypting to identities using a (compact) master public key, like how IBE is used, with the benefit that the PKG is substituted with a weaker entity called "key curator" who has no knowledge of any secret keys. Here individuals generate their secret keys on their own and then publicly *register* their identities and their corresponding public keys to the key curator. Finally, individuals obtain "rare" decryption-key updates from the key curator as the population grows. In their work, they gave a construction of RBE schemes based on the combination of indistinguishability obfuscation and somewhere statistically binding hash functions. However, they left open the problem of constructing RBE schemes based on standard assumptions.

In this work, we resolve the above problem and construct RBE schemes based on standard assumptions (e.g., CDH or LWE). Furthermore, we show a new application of RBE in a novel context. In particular, we show that anonymous variants of RBE (which we also construct under standard assumptions) can be used for realizing abstracts forms of anonymous messaging tasks in simple scenarios in which the parties communicate by writing messages on a shared board in a synchronized way.

*sanjamg@berkeley.edu, Berkeley. Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

†mdhajiabadi@berkeley.edu, Berkeley and University of Virginia. Supported by NSF award CCF-1350939 and AFOSR Award FA9550-15-1-0274.

‡mohammad@virginia.edu, University of Virginia. Supported by NSF CAREER award CCF-1350939, and two University of Virginia's SEAS Research Innovation Awards.

§ahmadreza@virginia.edu, University of Virginia. Supported by NSF award CCF-1350939.

¶sruthisekar@iisc.ac.in, Indian Institute of Science, Bangalore.

Contents

1 Introduction

Identity based encryption, first introduced by Shamir [Sha84], and then realized based on pairings by Boneh and Franklin [BF01], allows a set of remote parties to communicate secretly by only knowing one single public key and the name of the recipient identity. Despite being a milestone in foundations of cryptography and a powerful tool for simplifying key-management, real-world uses of IBE schemes come with a major caveat: IBE schemes require a *private-key generator* (PKG) who holds the master key and uses it to generate decryption keys for the identities. Therefore, the PKG has the ability to decrypt all cipherexts. This issue, inherent to IBE by design, is known as the *key escrow* problem.

Many previous works tried to rectify the key escrow problem in IBE. These efforts include making the trust de-centralized using multiple PKGs [BF01], making the PKG *accountable* for distributing the decryption keys to unauthorized users [Goy07, GLSW08], making it hard for PKG to find out the receiver identity in a large set of identities [CCV04, Cho09, WQT18], or using Certificateless Public Key Cryptography [ARP03] as a hybrid of IBE and public-key directories. However, none of these efforts resolve the key escrow problem completely.

Motivated by entirely removing PKGs from IBE schemes, recently Garg, Hajiabadi, Mahmoody, and Rahimi [GHMR18] introduced the notion of *registration-based encryption* (RBE for short). In an RBE scheme, the PKG entity is substituted by a much weaker entity called the *key curator* (KC for short). The KC will not posses any secret keys, and all it does is to manage the set of public keys of the *registered* identities. More specifically, in an RBE scheme identities (or, rather the users corresponding to the identities) generate their own public and secret keys, and then they will register their public keys to the KC who maintains and updates a public parameter pp_n where n is the number of parties who have joined the system so far. This public parameter pp_n can be used (now and in the future) to encrypt messages to any of the n identities who have registered so far. The first key efficiency requirement of RBE schemes is that pp_n is *compact*; i.e., $\mathrm{poly}(\kappa, \log n)$ in size where κ is the security parameter. Moreover, RBE requires that the process of "identity registration" is also efficient; i.e., runs in time $\mathrm{poly}(\kappa, \log n)$. In order to connect an updated public parameter pp_n to the previously registered identities, RBE allows the identities to obtain *updates* from the KC, which (together with their own secret keys) can be used as decryption keys. The second efficiency requirement of RBE is that such updates are only needed at most $O(\log n)$ times over the lifetime of the system. In summary, RBE schemes are required to perform both "identity registration" and "update generation" in time *sublinear* in n. In particular, these two operations are required to run in time $\mathrm{poly}(\kappa, \log n)$ where κ is the security parameter.

The work of [GHMR18] showed how to construct RBE schemes based on the combination of indistinguishability obfuscation (IO) [BGI+01, GGH+13] and somewhere statistically binding hash functions (SSBH) [HW15]. Towards the goal of basing RBE schemes on more standard assumptions, [GHMR18] also showed how to construct *weakly efficient* RBE schemes with $\mathrm{poly}(\kappa, n)$ identity registration time based on standard assumptions such as CDH and LWE. The work of [GHMR18] left open the question of constructing RBE schemes (with the required registration time $\mathrm{poly}(\kappa, \log n)$) from standard cryptographic assumptions. This gap leads us to the following question, which is the main question studied in this work:

Can we base registration-based encryption on standard cryptographic assumptions?

Our results. In this work, we resolve the above question affirmatively. Namely, as the main result of this work, we construct RBE schemes with all the required compactness and efficiency requirements based on standard assumptions such as LWE, CDH, or Factoring. In particular, in our RBE scheme (based on CDH or LWE assumptions) the time it takes to register any new identity into the system is only $\mathrm{poly}(\kappa, \log n)$ where κ is the security parameter and n is the number of identities registered into the system so far.

In addition to resolving the question above, in this work we show the usefulness of RBE by demonstrating a connection between an *anonymous* variant of RBE (defined similarly to how anonymous IBE [BDCOP04] is defined) to an abstract anonymous messaging primitive that we call *anonymous board communication* (ABC for short). At a high level, (anonymous) IBE fails to achieve ABC, exactly because of the key escrow problem, which does not exist in RBE.

1.1 Technical Overview

In this subsection, we will first describe the high level ideas behind our RBE scheme based on standard assumptions. We will then describe how to add the extra property of anonymity to RBE, allowing it to be used for realizing ABC as described above.

Figure 1 shows the high level structure and the roadmap of the primitives that we use (and construct along the way) for achieving RBE from standard assumptions. The features in parentheses (i.e., "blind" and "anonymous") can be added to or removed from the figure. When they are added, Figure 1 demonstrates the way we obtain *anonymous* RBE.

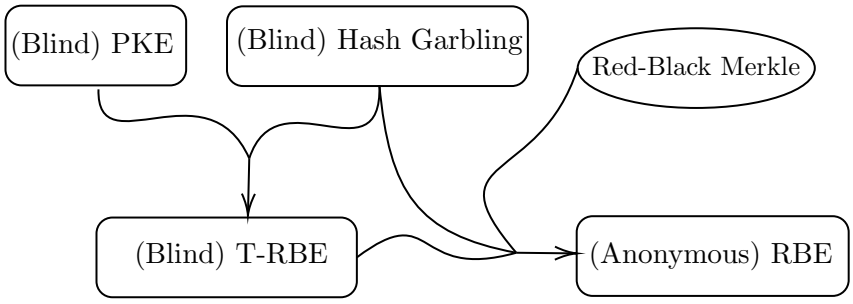

Figure 1: Roadmap

The big picture. We construct our RBE scheme based on the primitive *hash garbling* which was formally defined in [GHMR18] but was used implicitly in some prior works [CDG+17, DG17, DG17, DGHM18, BLSV18], and a new primitive "time-stamp" RBE (T-RBE for short) that we introduce in this work. T-RBE is a special case of RBE, where we use the time-stamp t_{id} of the registration time of each identity instead of their (arbitrary string identity) id. T-RBE also requires the same efficiency and compactness requirements of RBE. Since T-RBE is a special case of RBE, achieving T-RBE from standard assumptions is potentially easier, and indeed we leverage this fact in our approach.

In particular, as depicted in Figure 1, we first show how T-RBE can be constructed from public-key encryption and hash garbling schemes. Then, having T-RBE, hash garbling, and a red-black Merkle tree we show how to construct an RBE scheme. This resolves the open question of [GHMR18] where they obtained *weakly* efficient RBE schemes. Finally, we show that by substituting

each of the used primitives with a "blind" version of them as defined in [BLSV18] we can bootstrap our new RBE construction to make it anonymous. Below, we describe each of these steps, their corresponding challenges, and how we resolve them.

The weakly efficient RBE of [GHMR18]. Before describing our RBE construction, we describe the main challenge in achieving the required registration efficiency, which made the construction of [GHMR18] (based on standard assumptions) *weakly efficient*. For that, we need to quickly recall the high level structure of the (weakly-efficient) construction of [GHMR18]. At a high level, the registration algorithm in the construction of [GHMR18] leads to an auxiliary information stored at the KC that consists of some Merkle trees $\mathsf{Tree}_1, \ldots, \mathsf{Tree}_\eta$ where each of these trees in their leaves contains the ids of the already registered identities, along with their public keys pk. The encryption algorithm of this scheme requires to use the public key corresponding to the specific identity to encrypt the message. To do so, it requires to do a binary search on the tree containing the id (so as to access the corresponding public key), for which it is required that the leaves of the trees are *sorted* according to their labels (i.e., identities) from left to right. This binary search was captured by generation of a sequence of garbled circuits. Now, going back to the registration algorithm, if the execution of the registration algorithm ever leads to two trees with the same size, those two will be merged into one tree, which *again* needs to have the identities sorted. (Merging the trees is necessary to keep the public parameter compact, because the public parameter is basically the concatenation of the roots of theses trees, and so the number of trees shall remain bounded.) Hence, every time a tree merge operation is done in their registration process, the entire tree needs to be restructured based on the *newly sorted* list of the identities in the two trees. This very sorting made the construction of [GHMR18] weakly efficient.

Step one: weakening the functionality for sake of efficiency. To overcome the challenge of achieving the required efficiency for RBE, in this work we first introduce a new primitive that we call T-RBE (short form of *time-stamp* RBE) that weakens RBE for the sake of achieving efficient registration. (Looking ahead, we will later bootstrap T-RBE to RBE.) This new primitive has the same functionality as RBE, except that in a T-RBE, the identities do not register with their actual ids, but rather with the corresponding time-stamps of their registration moments. This has the immediate obvious effect that the time-stamp strings already arrive in sorted order, hence removing the need of re-structuring the trees for sorting purposes. Note that since time-stamps are taking the role of the names of identities, all the algorithms (including the the encryption algorithm) shall use the time-stamp as the identity's label (instead of the id).

We now describe how T-RBE can indeed overcome the efficiency challenge left open about RBE. Notice that the time-stamps used for registrations are already sorted based on the arrival times. One useful consequence of this phenomenon is that, if we apply the same approach of [GHMR18] for weakly-efficient RBE, the resulting T-RBE will have leaves of the trees *automatically sorted* at all times. Hence, when we want to merge two trees, we may simply hash their roots into a new root, with a guarantee that all the leaves in one sub-tree will be larger than all those in the other sub-tree. Hence, by restricting the problem to T-RBE, we would not require to restructure the trees when we merge them. Hence, the T-RBE registration overcomes the main reason for inefficiency in the RBE construction of [GHMR18] and turns out to be efficient as required. The other algorithms of the T-RBE scheme are also similar to the corresponding algorithms of the RBE in [GHMR18], with some natural changes.

Step two: bootstrapping T-RBE to RBE (using Red-Black Merkle Tree). Recall that T-RBE uses the *time-stamps* of the identities as if they were the actual ids. So, to come up with a construction of RBE with the same efficiency as T-RBE, we somehow need to find a way to connect the actual identities of the parties to their corresponding registration time-stamps. In particular, the encryption algorithm of the RBE scheme, which now takes as input an identity id as opposed to a time-stamp, would need to first obtain the corresponding time-stamp of the given id, and then run the T-RBE encryption algorithm using this time-stamp. Further, we also need to ensure that the registration and update algorithms of this RBE scheme are efficient. At a high level, we achieve this efficiency by using a red-black Merkle tree in addition to the auxiliary input maintained by our T-RBE scheme. Such a tree allows us to (indirectly) obtain the time-stamp corresponding to an identity id in an efficient way (details of which are described below), without having to store the tree in the public parameter (which is prohibitive due to the tree size).

In order to enable this indirect access to the red-black Merkle tree, we will make further use of the hash-garbling primitive. Here we give a high-level description of our RBE scheme which uses T-RBE (and hash garbling) as subroutines together with the help of a red-black Merkle tree. In this scheme, the auxiliary information aux, stored by the key curator consists of η full binary Merkle trees $\mathsf{Tree}_1, \ldots, \mathsf{Tree}_\eta$, as was in the construction of [GHMR18]. In addition, aux contains a red-black Merkle tree $\mathsf{TimeTree}$, whose leaves contain pairs of identities/time-stamps, sorted according to the identities. $\mathsf{TimeTree}$ is a key part in enabling the efficiency of our registration part.

- **The description of $\mathsf{TimeTree}$.** It is a close variant of red-black Merkle tree, where each leaf is of the form $(\mathsf{id}, t_{\mathsf{id}})$ and every non-leaf node in this tree contains the hash of its left child, the largest identity in its left sub-tree and the hash of the right child (all hashes use a hash key generated at the setup and described by a CRS). It differs from just red-black tree in the sense that each internal node also contains the largest identity in its left sub-tree. The choice of this specific data-structure is crucial for our construction. Notice that the leaves are sorted in ascending order of the identities. In addition, every node has another bit of information representing its color, which would be helpful in keeping the tree balanced using the red-black Merkle tree rotation algorithms.

- **How to register (Updating aux).** As we use the T-RBE as a subroutine, aux consists of the auxiliary information of T-RBE, aux_T, the $\mathsf{TimeTree}$ and a list of already registered identities. aux_T itself comprises of the Merkle trees $\mathcal{T} = \{\mathsf{Tree}_1, \cdots, \mathsf{Tree}_\eta\}$, with the time-stamps and their corresponding public keys at the leaf nodes. So to update the aux when somebody registers, we insert their identity id as well as their time-stamp t_{id} to $\mathsf{TimeTree}$, update aux_T using the T-RBE subroutine and add id to the list of registered identities. Recall that the T-RBE registration process involves creating a new Merkle tree with leaf nodes t_{id} and its corresponding public key pk and the merging the trees in $\mathcal{T}$ which are of same depth (merging only requires hashing the roots of the two trees to obtain a new tree).

- **How to encrypt.** The encryption algorithm takes as input the public parameter pp, a message m, and a receiver's identity id, and outputs a ciphertext, which is obtained by encrypting m using the T-RBE encryption under the time-stamp corresponding to id. To do this, the encryption algorithm requires to first look up id in $\mathsf{TimeTree}$ to obtain its time-stamp and then use it to encrypt m under the T-RBE encryption. However, the encryption algorithm only takes pp as public information, which is too small to contain $\mathsf{TimeTree}$, and so a "direct search" is impossible. To get around this problem, the encryption algorithm "defers" this

search process to the time of decryption: Specifically, the encryption algorithm constructs a sequence of (garbled) programs, which enable one to do a binary search on $\mathsf{TimeTree}$ (during the decryption time) to obtain the time-stamp corresponding to id and then to use it to encrypt the message using the T-RBE encryption. The ciphertext then consists of the hash garbling of these programs.

- **How to decrypt.** The decryption algorithm takes as input two paths $u = \mathsf{pth}_1, \mathsf{pth}_2$, a secret key sk and a ciphertext, which contains the garbled programs, and outputs a message (or aborts). Here, pth_1 is the path from the root of $\mathsf{TimeTree}$ to the node that contains the id of the decryptor and its time-stamp t_{id}, and pth_2 is the updated path (obtained using the update algorithm of T-RBE) required for running the T-RBE decryption algorithm. Using pth_1, the decryptor runs the hash garbled programs to obtain the ciphertext under the T-RBE scheme with time-stamp t_{id} as the output of the last program. Then, it runs the decryption algorithm of T-RBE with inputs pth_2, sk and the obtained ciphertext to get the message m.

- **How to update u (the auxiliary information required by decryptor).** With the registration as described above, we can guarantee the efficiency. The updation algorithm requires to only read a path in the $\mathsf{TimeTree}$, which leads to id and its time-stamp t_{id} and further use the updation algorithm of the T-RBE scheme. This would also be efficient. But we also need to guarantee that the number of times an id calls the updation algorithm is at most $\log(n)$ (where n is the the number of identities registered so far). This is not guaranteed if only have a single variant of the $\mathsf{TimeTree}$. This is because, each time an id registers, its addition to the $\mathsf{TimeTree}$ modifies the root hash, changing the root-to-leaf path for every other identity registered. We resolve this issue by maintaining a variant of the same $\mathsf{TimeTree}$, at the times corresponding to which each Merkle tree in $\mathcal{T}$ was last updated (which means we maintain at most $\log n$ variants of $\mathsf{TimeTree}$). This guarantees two things: firstly that an identity contained in Tree_i of $\mathcal{T}$ will definitely be contained in the corresponding i-th variant of $\mathsf{TimeTree}$ and secondly that this identity only requires to update its root-to-leaf path in the $\mathsf{TimeTree}$, when the tree Tree_i is modified. This would guarantee that the number of updates required by every id is at most $\log(n)$.

Adding anonymity. Adding the anonymity feature to our RBE scheme involves techniques which are in essence same as those used in [BLSV18]. We build an RBE scheme achieving the stronger notion of "blindness," which in turn implies the required anonymity property of an anonymous RBE scheme. While the notion of anonymity guarantees that the identity id is hidden along with the message being encrypted (similar to an anonymous IBE), the property of "blindness" gives a stronger guarantee that the ciphertext generated on a uniformly random message looks uniform. The fact that this stronger guarantee is achieved by our scheme and that it implies anonymity is shown in 5.3.

As shown in the Figure 1, we can get the blind T-RBE based on blind PKE and blind hash garbling schemes. The construction of the blind T-RBE is exactly the same as the regular T-RBE, except that instead of using a regular PKE scheme and a hash garbling scheme, we will use blind variants of these primitives and separate the corresponding ciphertexts and hash inputs into two parts. Then using blind T-RBE and blind hash garbling, we can get the desired anonymous RBE scheme (which is in fact a blind RBE).

1.2 Potential Applications of Anonymous RBE

Here we describe a possible application of anonymous RBE, which demonstrates the power of (anonymous) RBE in scenarios where other seemingly similar and powerful primitives (such as anonymous IBE [BDCOP04, BW06] or anonymous PKE [BBDP01]) seem incapable of.

Anonymous board communication (ABC). In an ABC scheme, a dynamic set of (semi-honest) parties $\{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ anonymously communicate by writing and reading on a single shared board $\mathbf{B}$ in a synchronized way. More formally, whenever a party joins the system, they update some information on the board $\mathbf{B}$. Also, the communication between $\{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ is done in synchronized cycles. In cycle t, each identity id_i has a list of messages $\mathsf{m}_{i,1}, \mathsf{m}_{i,2}, \ldots$ to be delivered to some parties $\mathsf{id}_{i,1}, \mathsf{id}_{i,2}, \ldots$. Then at the beginning of the cycle t, all of the parties write some arbitrary messages on the common board $\mathbf{B}$. After all the parties are done with writing, in the second half of the cycle t, all the parties read (their selected parts or all of) the content of the board $\mathbf{B}$. By the end of cycle t, all the parties should be able to obtain the messages intended to be sent to them. Namely, if the message m was sent to identity id (by another identity id'), by the end of cycle t, m should be obtained by id. The security goal is to keep all the senders and receivers anonymous from the perspective of any adversary who can read all the information written on the common board $\mathbf{B}$. The efficiency goal here is have parties write their messages in "small" time. In particular, we require the write time of each party to be $\mathrm{poly}(\kappa, \log n) \cdot k$ where k is the total length of the message to be sent out and κ is the security parameter.

ABC from anonymous IBE? One might try to get ABCs from anonymous IBE as follows. The public key of the anonymous IBE scheme is written on the board, and the parties use it to compose their messages for the desired receivers. During the read part of each cycle, each party will read the whole board $\mathbf{B}$ and try to decrypt messages that are sent to them. This approach provides receiver anonymity and sender efficiency, but the main issue is that the master secret key should be stored somewhere and there is only one place for storing it: on the board $\mathbf{B}$. So, anyone who has full access to the board can decrypt all the messages. On the other hand, one might try to avoid IBE and use a public-key directory storing public-keys of an (anonymous) PKE scheme at the board $\mathbf{B}$. Then the parties can try to use this information and write their messages to the desired recipients on the same board. The problem with this approach is that the writing parties need to read the whole set of public-keys from the board to obtain the desired target public keys.

ABC from anonymous RBE. Interestingly, anonymous RBE directly enables ABC and achieves what anonymous IBE and PKE seem incapable of. In particular, a specific part of the common board $\mathbf{B}$ would be dedicated to store the information required for maintaining the public parameters and the auxiliary information of the KC of the anonymous RBE scheme (and note that no master secret key exists). This way, by reading the public parameter pp_n (after n people have joined), an (anonymous-receiver) message can be composed to any recipient. In particular, if $\mathsf{m}_{i,1}, \mathsf{m}_{i,2}, \ldots$ are to be sent to parties $\mathsf{id}_{i,1}, \mathsf{id}_{i,2}, \ldots$ by party id_i, all id_i does is to generate anonymous-receiver ciphertexts containing the messages and encrypted to the right identities. The anonymity of the sender stems from the fact that RBE can be used like an IBE and anyone (including those in the system) can compose messages to any other identify in a secret way. Finally, the fact that the key curator is transparent (and all it does is curating the keys) allows the parties to join the system freely one by one and update the public parameter as required.

Relation to other works on secure messaging. We emphasize that our ABC primitive is not by any means aiming to capture practical scenarios of secure messaging that have been a source of intense work over recent years [RMS18, CGF10, CGBM15, BSJ+17, AKTZ17, UDB+15,

CGCD+17, CB95]. For example a large body of work (e.g., see [CGF10, CGBM15, CB95]) explore realistic messaging settings in which a set of distributed parties communicate over a network and aim to message each other in a way that senders, receivers, (and some more specific relations) remain secret despite the messages being sent over the network. Some of these works achieve privacy by using non-colluding servers, while ABC only uses one "server". On the other hand, our ABC occurs in a centralized setting in which all the messages by the parties are directly written to and read from the shared board. In a different direction, many works (e.g., see [BSJ+17, CGCD+17, PR18, JS18, BGB04] in the context of what is now known as "Ratcheted Key Exchange" study ways to expand shared keys to secure refreshed keys to be used in the future, and so they fundamentally differ from ABC simply because in ABC there are no keys shared between the parties. Thus, we note that the main goal of introducing ABC is to demonstrate a basic abstract messaging scenario with challenges that can be resolved immediately using (anonymous) RBE, while other powerful tools do not seem to be capable of doing the same.

2 Preliminary Definitions

In this section we describe the needed definitions. We separate the definitions of variants of RBE into those borrowed from previous work and those that are introduced in this work.

2.1 Previous Definitions about RBE

In this subsection we recall the definition of RBE, taken verbatim from [GHMR18].

Definition 2.1 (Syntax of RBE)**.** A *registration-based encryption* (RBE for short) scheme consists of PPT algorithms $(\mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ working as follows. The Reg and Upd algorithms are performed by the key curator, which we call KC for short.

- **Generating common random string.** Some of the subroutines below will need a common random string crs, which could be sampled publicly using some public randomness beacon. crs of length $\mathrm{poly}(\kappa)$ is sampled at the beginning, for the security parameter κ.
- **Key generation.** $\mathsf{Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$: The randomized algorithm Gen takes as input the security parameter 1^κ and outputs a pair of public/secret keys $(\mathsf{pk}, \mathsf{sk})$. Note that these are only *public* and *secret* keys, not the *encryption* or *decryption* keys. 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 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.** $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$: 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.** $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$: The deterministic algorithm Upd takes as input the current information pp stored at the KC and an identity id, has *read only* oracle access to aux and generates an *update* information u that can help id to decrypt its messages.

- **Decryption.** $\mathrm{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct})$: 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 $\mathsf{m} \in \{0,1\}^*$ or in $\{\perp, \mathtt{GetUpd}\}$. The special symbol $\perp$ indicates a syntax error, while $\mathtt{GetUpd}$ indicates that more recent update information (than u) might be needed for decryption.

Definition 2.2 (Completeness, compactness, and efficiency of RBE)**.** For any interactive *computationally unbounded* adversary Adv that still has a limited $\mathrm{poly}(\kappa)$ round complexity, consider the following game $\mathtt{Comp}_{\mathsf{Adv}}(\kappa)$ between Adv and a challenger Chal.

1. **Initialization.** Chal sets $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathsf{u} = \perp$, $\mathsf{ID} = \varnothing$, $\mathsf{id}^* = \perp$, $t = 0$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to Adv.
2. Till Adv continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, Adv chooses exactly one of the actions below to be performed.
   (a) **Registering new (non-target) identity.** Adv sends some $\mathsf{id} \notin \mathsf{ID}$ and pk to Chal. Chal registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} := \mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathsf{ID} := \mathsf{ID} \cup \{\mathsf{id}\}$.
   (b) **Registering the target identity.** If id^* was chosen by Adv already (i.e., $\mathsf{id}^* \neq \perp$), skip this step. Otherwise, Adv sends some $\mathsf{id}^* \notin \mathsf{ID}$ to Chal. Chal then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathrm{Gen}(1^\kappa)$, updates $\mathsf{pp} :=$ $\mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*), \mathsf{ID} := \mathsf{ID} \cup \{\mathsf{id}^*\}$, and sends pk^* to Adv.
   (c) **Encrypting for the target identity.** If $\mathsf{id}^* = \perp$ then skip this step. Otherwise, Chal sets $t = t + 1$, then Adv sends some $\mathsf{m}_t \in \{0,1\}^*$ to Chal who then sets $\mathsf{m}'_t := \mathsf{m}_t$ and sends back a corresponding ciphertext $\mathsf{ct}_t \leftarrow \mathrm{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ to Adv.
   (d) **Decryption by target identity.** Adv sends a $j \in [t]$ to Chal. Chal then lets $\mathsf{m}'_j = \mathrm{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$. If $\mathsf{m}'_j = \mathtt{GetUpd}$, then Chal obtains the update $\mathsf{u} = \mathrm{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ and then lets $\mathsf{m}'_j = \mathrm{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$.
3. The adversary Adv wins the game if there is some $j \in [t]$ for which $\mathsf{m}'_j \neq \mathsf{m}_j$.

Let $n = |\mathsf{ID}|$ be the number of identities registered till a specific moment. We require the following properties to hold for any Adv (as specified above) and for *all* the moments (and so for all the values of ID and $n = |\mathsf{ID}|$ as well) during the game $\mathtt{Comp}_{\mathsf{Adv}}(\kappa)$.

- **Completeness.** $\Pr[\mathsf{Adv} \text{ wins in } \mathtt{Comp}_{\mathsf{Adv}}(\kappa)] = \mathrm{negl}(\kappa)$.
- **Compactness of public parameters and updates.** $|\mathsf{pp}|, |\mathsf{u}|$ are both $\leq \mathrm{poly}(\kappa, \log n)$.
- **Efficiency of runtime of registration and update.** The running time of each invocation of Reg and Upd algorithms is at most $\mathrm{poly}(\kappa, \log n)$. (This implies the compactness property.)
- **Efficiency of the number of updates.** The *total* number of invocations of Upd for identity id^* in Step 2d of the game $\mathtt{Comp}_{\mathsf{Adv}}(\kappa)$ is at most $O(\log n)$ for every n during $\mathtt{Comp}_{\mathsf{Adv}}(\kappa)$.

Definition 2.3 (WE-RBE)**.** A *weakly efficient RBE* (or WE-RBE for short) is defined similarly to Definition 2.2, where the specified $\mathrm{poly}(\kappa, \log n)$ runtime efficiency of the registration algorithm is not required anymore, but instead we require the registration time to be $\mathrm{poly}(\kappa, n)$.

Definition 2.4 (Security of RBE)**.** For any interactive PPT adversary Adv, consider the following game $\mathtt{Sec}_{\mathsf{Adv}}(\kappa)$ between Adv and a challenger Chal. (Steps that are different from the completeness definition are denoted with purple stars ($\star\star$). Specifically, Steps 2c and 2d from Definition 2.2 are replaced by Step 3 below. Additionally, Step 3 from Definition 2.2 is replaced by Step 4 below.)

1. **Initialization.** Chal sets $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathsf{ID} = \varnothing$, $\mathsf{id}^* = \perp$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to Adv.
2. Till Adv continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, Adv chooses exactly one of the actions below to be performed.

(a) **Registering new (non-target) identity.** Adv sends some $\mathsf{id} \notin \mathsf{ID}$ and pk to Chal. Chal registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} := \mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathsf{ID} := \mathsf{ID} \cup \{\mathsf{id}\}$.

(b) **Registering the target identity.** If id^* was chosen by Adv already (i.e., $\mathsf{id}^* \neq \bot$), skip this step. Otherwise, Adv sends some $\mathsf{id}^* \notin \mathsf{ID}$ to Chal. Chal then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathrm{Gen}(1^\kappa)$, updates $\mathsf{pp} :=$ $\mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*), \mathsf{ID} := \mathsf{ID} \cup \{\mathsf{id}^*\}$, and sends pk^* to Adv.

3. ($\star\star$) **Encrypting for the target identity.** If no id^* was chosen by Adv before (i.e., $\mathsf{id}^* = \bot$) then Adv first sends some $\mathsf{id}^* \notin \mathsf{ID}$ to Chal. Next, Chal generates $\mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, b)$, where $b \leftarrow \{0,1\}$ is a random bit, lets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}^*\}$, and sends ct to Adv.
4. ($\star\star$) The adversary Adv outputs a bit b' and wins the game if $b = b'$.

We call an RBE scheme secure if $\Pr[\mathsf{Adv} \text{ wins in } \mathtt{Sec}_{\mathsf{Adv}}(\kappa)] < \frac{1}{2} + \mathrm{negl}(\kappa)$ for any PPT Adv.

2.2 New Definitions about (Anonymous) RBE

In this section we define an anonymity feature for the notion of RBE, and we will show later how to build efficient anonymous RBE from standard assumptions.

Definition 2.5 (Anonymous RBE)**.** An anonymous RBE scheme has the same syntax as that of an RBE scheme, with PPT algorithms $(\mathrm{Gen}, \mathrm{Reg}, \mathrm{Enc}, \mathrm{Upd}, \mathrm{Dec})$. 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 Adv, consider the game $\mathsf{EXP}^{\mathtt{Anon}}_{\mathsf{Adv}}(\kappa)$ between Adv and a challenger Chal as follows.

1. **Initialization.** Chal sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{ID} = \varnothing$, $\mathsf{id}_0 = \bot$, $\mathsf{id}_1 = \bot$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to Adv.

2. Till Adv continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, Adv can perform exactly one of the following actions.

   (a) **Registering new (non-target) identity.** Adv sends some $\mathsf{id} \notin \mathsf{ID}$ and pk to Chal. Chal registers $(\mathsf{id}, \mathsf{pk})$ by getting $\mathsf{pp} = \mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}, \mathsf{pk}))$ and lets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}\}$.

   (b) **Registering new target identity pair.** If id_0 or id_1 was chosen by Adv already (i.e., $\mathsf{id}_0 \neq \bot$ or $\mathsf{id}_1 \neq \bot$), skip this step. Otherwise, Adv sends challenges $\mathsf{id}_0, \mathsf{id}_1 \notin \mathsf{ID}$ to Chal. Chal first samples $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathrm{Gen}(1^\kappa)$, registers id_0 by setting $\mathsf{pp} = \mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}_0, \mathsf{pk}_0))$ and then samples $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathrm{Gen}(1^\kappa)$, registers id_1 by setting $\mathsf{pp} = \mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (\mathsf{id}_1, \mathsf{pk}_1))$. Next, Chal lets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}_0, \mathsf{id}_1\}$ and sends $\mathsf{pk}_0, \mathsf{pk}_1$ to Adv.

3. **Encrypting for the challenge identity.** If $\mathsf{id}_0, \mathsf{id}_1$ was not chosen by Adv already (i.e., $\mathsf{id}_0 = \bot, \mathsf{id}_1 = \bot$), then Adv first sends some $\mathsf{id}_0, \mathsf{id}_1 \notin \mathsf{ID}$ to Chal before continuing this step. Next, Chal samples a bit $b \in \{0,1\}$ and generates the challenge ciphertext $\mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}_b, b)$. Further, Chal sets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}_0, \mathsf{id}_1\}$ and sends ct to Adv.

4. The adversary Adv outputs a bit b' and wins the game if $b' = b$.

We call an RBE scheme an Anonymous-RBE if $\Pr[\mathsf{Adv} \text{ wins in } \mathsf{EXP}^{\mathtt{Anon}}_{\mathsf{Adv}}(\kappa)] < \frac{1}{2} + \mathrm{negl}(\kappa)$ for any PPT Adv.

As a step-stone toward building efficient (anonymous) RBE, we will first show how to build a primitive which we call *timestamp-RBE*. We define this notion formally below.

Definition 2.6 (T-RBE)**.** A *timestamp-RBE* (or T-RBE for short) has syntax exactly similar to Definition 2.1, except for one difference: we now consider the registration algorithm Reg, the encryption algorithm Enc and the update algorithm Upd to take as input the timestamp t_{id} of an identity id (binary representation of the time at which an identity registers) as input instead of the identity id. The completeness, compactness of public parameters, the efficiency of runtime of registration and update and the efficiency of the number of updates is exactly similar to Definition 2.2 and the security guarantee is similar to Definition 2.4, replacing identity id with its timestamp t_{id} in all the appropriate places. For a T-RBE, we define the notion of anonymity exactly as in Definition 2.5, replacing identity id with its timestamp t_{id} in all the appropriate places.

2.3 Blind Public Key Encryption

In this subsection we define blindness features for several cryptographic primitives, which will be used in our main constructions. We first start with the notion of blind PKE. The notion of blindness for PKE is well-studied with a few prior definitions; see, e.g., [BBDP01, Moh10]. Here we give a tailored version of this definition suitable for our later constructions.

Definition 2.7 (Blind Public Key Encryption)**.** A blind public key encryption scheme (with public parameters) has algorithms $(\mathsf{Params}, \mathrm{G}, \mathrm{E}, \mathrm{D})$ which is IND-CPA secure and satisfies the following additional security property: the function $\mathrm{E}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r})$ can be expressed as $\mathrm{E}_1(\mathsf{pp}; \mathsf{r}) || \mathrm{E}_2(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r})$ such that the distribution of $\{(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathrm{Enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r})) : \mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{G}(\mathsf{pp}), \mathsf{m} \xleftarrow{\$} \mathcal{M}, \mathsf{r} \xleftarrow{\$} \{0,1\}^*\}$ is computationally indistinguishable from
$\{(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathrm{E}_1(\mathsf{pp}; \mathsf{r}), \mathsf{subct}_2) : \mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{G}(\mathsf{pp}), m \xleftarrow{\$} \mathcal{M}, \mathsf{r} \xleftarrow{\$} \{0,1\}^*, \mathsf{subct}_2 \xleftarrow{\$} \{0,1\}^L\}$, where $L = |\mathrm{E}_2(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r})|$.

We now define a blindness notion for garbled circuits. Our blindness requirement is the same as that introduced and used by [BLSV18].

Definition 2.8 (Blind Garbled Circuits [BLSV18])**.** A garbling scheme consists of PPT algorithms $(\mathsf{Garble}, \mathsf{Eval})$ and a simulator $\mathsf{G.Sim}$ where:

1. $\mathsf{Garble}(1^\lambda, 1^l, 1^m, \mathrm{C}; \mathsf{state}) := \mathsf{Garble}_1(1^\lambda, 1^l, 1^m; \mathsf{state}) || \mathsf{Garble}_2(1^\lambda, 1^l, 1^m, \mathrm{C}; \mathsf{state})$. Garble_1 takes as input the security parameter λ, the input length l and output length m for circuit C and a random value $\mathsf{state} \in \{0,1\}^\lambda$ and outputs the labels for input wire of the Garbled circuit $\{\mathsf{lab}_{j,b}\}_{j \in [l], b \in \{0,1\}}$, where $\mathsf{lab}_{j,b} \in \{0,1\}^\lambda$ and Garble_2 takes the circuit C in addition, and outputs the garbled circuit $\tilde{\mathrm{C}}$.

2. $\mathsf{Eval}(1^\lambda, \tilde{\mathrm{C}}, \tilde{\mathsf{lab}})$ is a deterministic algorithm that takes as input the garbled circuit $\tilde{\mathrm{C}}$, along with a set of l labels $\tilde{\mathsf{lab}} = \{\mathsf{lab}_j\}_{j \in [l]}$ and outputs a string $y \in \{0,1\}^m$.

3. $\mathsf{G.Sim}(1^\lambda, 1^{|\mathrm{C}|}, 1^l, y)$ takes as input the security parameter λ, the description length of circuit C, the input length l and a string $y \in \{0,1\}^m$ and outputs a simulated garbled circuit $\tilde{\mathrm{C}}$ and labels $\tilde{\mathsf{lab}}$.

A blind garbling scheme must satisfy the following properties:

1. **Correctness.** For all circuits C, inputs x and all $(\tilde{\mathrm{C}}, \{\mathsf{lab}_{j,b}\}_{j\in[l],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(\mathrm{C})$ and $\tilde{\mathsf{lab}} = \{\mathsf{lab}_{j,x_j}\}_{j\in[l]}$, we have $\mathsf{Eval}(\tilde{\mathrm{C}}, \tilde{\mathsf{lab}}) = \mathrm{C}(x)$.

2. **Simulation Security.** For all circuits $\mathrm{C} : \{0,1\}^l \to \{0,1\}^m$ and all inputs $x \in \{0,1\}^l$, the following distributions are computationally indistinguishable:

$$\{(\tilde{\mathrm{C}}, \tilde{\mathsf{lab}}) : (\tilde{\mathrm{C}}, \{\mathsf{lab}_{j,b}\}_{j\in[l],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(\mathrm{C}), \tilde{\mathsf{lab}} = \{\mathsf{lab}_{j,x_j}\}_{j\in[l]}\}$$
$$\overset{c}{\approx} \{(\tilde{\mathrm{C}}, \tilde{\mathsf{lab}}) : (\tilde{\mathrm{C}}, \tilde{\mathsf{lab}}) \leftarrow \mathsf{G.Sim}(1^\lambda, 1^{|\mathrm{C}|}, 1^l, \mathrm{C}(x))\}$$

3. **Blindness.** $\mathsf{G.Sim}(1^\lambda, 1^{|\mathrm{C}|}, 1^l, U_m) \overset{c}{\approx} U$. The output of the simulator on a completely uniform output is indistinguishable from a uniform bit string.

We now review the notion of blind batch encryption from [BLSV18]. The notion of batch encryption is in turn similar to some notions such as hash encryption and laconic oblivious transfer [DG17, CDG+17].

Definition 2.9 (Blind Single Batch Encryption [BLSV18])**.** A blind single batch encryption scheme consists of PPT algorithms $(\mathsf{Setup}, \mathsf{H}, \mathsf{HEnc}, \mathsf{HDec})$:

1. $\mathsf{Setup}(1^\lambda, 1^l)$ takes as input the security parameter λ, a length parameter l and outputs a hash key hk.

2. $\mathsf{H}(\mathsf{hk}, x)$ takes as input a hash key hk and $x \in \{0,1\}^l$ and deterministically outputs $h \in \{0,1\}^\lambda$.

3. $\mathsf{HEnc}(\mathsf{hk}, h, i, M)$ takes as input the hash key hk, hash value h and a message matrix $M \in \{0,1\}^{1\times 2}$ and outputs a ciphertext ct, which can be written as a concatenation of two parts $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.

4. $\mathsf{HDec}(\mathsf{hk}, x, i, \mathsf{ct})$ takes as input the ciphertext ct and outputs an $m \in \{0,1\}$.

A blind single batch encryption must satisfy the following properties:

1. **Correctness.** Let $\mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^l)$. For all x, i, M, taking $h = \mathsf{H}(\mathsf{hk}, x)$, $\mathsf{ct} = \mathsf{HEnc}(\mathsf{hk}, h, i, M)$, it holds that $\mathsf{HDec}(\mathsf{hk}, x, i, \mathsf{ct}) = M_{x_i}$, with probability at least $1/2 + 1/\operatorname{poly}(\lambda)$ over the randomness of HEnc.

2. **Semantic Security.** For any PPT adversary Adv the probability of winning in the following game between Adv and a challenger Chal is $1/2 + \operatorname{negl}(\lambda)$:

   (a) Adv takes as input 1^λ and sends $1^l, x \in \{0,1\}^l, i \in [l]$ to Chal.

   (b) Chal generates $\mathsf{hk} = \mathsf{Setup}(1^\lambda, 1^l)$ and sends hk to Adv.

   (c) Adv sends a pair $M^{(0)}, M^{(1)}$, such that $M^{(0)}_{x_i} = M^{(1)}_{x_i}$, to Chal.

   (d) Chal computes $h = \mathsf{H}(\mathsf{hk}, x)$, chooses $b \in_R \{0,1\}$ and sends $\mathsf{ct} = \mathsf{HEnc}(\mathsf{hk}, h, i, M^{(b)})$ to Adv.

   (e) Adv outputs a bit b' and wins if $b' = b$.

3. **Blindness.** The encryption $\mathsf{HEnc}(\mathsf{hk}, h, i, M; \mathsf{r})$ can be considered as a concatenation of $\mathsf{HEnc}_1(\mathsf{hk}; \mathsf{r}) || \mathsf{HEnc}_2(\mathsf{hk}, h, i, M; \mathsf{r})$. Further, any PPT adversary Adv the probability of winning in the following game with a Challenger Chal is at most $1/2 + \operatorname{negl}(\lambda)$:

(a) Adv takes as input 1^λ and sends $1^l, x, i$ to Chal.

(b) Chal generates $\mathsf{hk} = \mathsf{Setup}(1^\lambda, 1^l)$ and computes $h = \mathsf{H}(\mathsf{hk}, x)$. Further it samples a random $b \in \{0,1\}$, a random message matrix $M \in \{0,1\}^{1\times 2}$ and encrypts $(\mathsf{subct}_1, \mathsf{subct}_2) \leftarrow \mathsf{HEnc}(\mathsf{hk}, h, i, M)$. It generates ct as:

- If $b = 0$, the $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.
- If $b = 1$, then pick a random subct'_2 of same length as subct_2 and set $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}'_2)$.

Chal sends hk, ct to Adv.

(c) Adv outputs a bit b' and wins if $b' = b$.

3 Blind Hash Garbling

In this section we introduce and build a primitive which we call blind hash garbling, which will later be used as an ingredient in the construction of anonymous T-RBE schemes. We first define this notion below and will then show how to build it using tools defined in the previous sections.

3.1 Definition of Blind Hash Garbling

The notion of hash garbling was defined in [GHMR18]; here we review this notion and define a blindness feature for it.

Definition 3.1 (Blind Hash Garbling)**.** A blind hash garbling scheme has the following polynomial time algorithms HGen, Hash, HObf, HInp:

- $\mathrm{HGen}(1^\kappa, 1^l) \to \mathsf{hk}$. It takes as input the security parameter κ and an output length parameter 1^l for $l \le \mathrm{poly}(\kappa)$, and outputs a hash key hk.
- $\mathrm{Hash}(\mathsf{hk}, x) = y$. It takes as input hk and $x \in \{0,1\}^l$ and deterministically outputs $y \in \{0,1\}^\kappa$.
- $\mathrm{HObf}(\mathsf{hk}, \mathrm{C}, \mathsf{state}) \to \tilde{\mathrm{C}}$. It takes as input hk, a circuit C, and a secret state $\mathsf{state} \in \{0,1\}^\kappa$ and outputs a circuit $\tilde{\mathrm{C}}$.
- $\mathrm{HInp}(\mathsf{hk}, y, \mathsf{state}) \to \tilde{y}$. This takes as input hk, a value $y \in \{0,1\}^\kappa$, and secret state state and outputs $\tilde{y}$. Consider $\tilde{y}$ as concatenation of two parts $\tilde{y}_1 || \tilde{y}_2$.

A blind hash garbling scheme must satisfy the following properties:

- **Correctness.** For all κ, l, hash key $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^l)$, circuit C, input $x \in \{0,1\}^l$, $\mathsf{state} \in \{0,1\}^\kappa$,$\tilde{\mathrm{C}} \leftarrow \mathrm{HObf}(\mathsf{hk}, \mathrm{C}, \mathsf{state})$ and $\tilde{y} \leftarrow \mathrm{HInp}(\mathsf{hk}, \mathrm{Hash}(\mathsf{hk}, x), \mathsf{state})$, then $\tilde{\mathrm{C}}(\tilde{y}, x) = \mathrm{C}(x)$.
- **Security.** There exists a PPT simulator Sim such that for all κ, l and PPT (in κ) Adv we have that
$$(\mathsf{hk}, x, \tilde{y}_1, \tilde{\mathrm{C}}, \tilde{y}_2) \overset{c}{\approx} (\mathsf{hk}, x, \tilde{y}_1, \mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, \mathrm{C}(x))),$$
where $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^l)$, $(\mathrm{C}, x) \leftarrow \mathsf{Adv}(\mathsf{hk})$,$\mathsf{state} \leftarrow \{0,1\}^\kappa$, $\tilde{\mathrm{C}} \leftarrow \mathrm{HObf}(\mathsf{hk}, \mathrm{C}, \mathsf{state})$ and $(\tilde{y}_1, \tilde{y}_2) \leftarrow \mathrm{HInp}(\mathsf{hk}, \mathrm{Hash}(\mathsf{hk}, x), \mathsf{state})$.

- **Blindness.** The function $\tilde{y} = \mathsf{HInp}(\mathsf{hk}, y, \mathsf{state}; \mathsf{r})$ can be expressed as the concatenation $\mathsf{HInp}_1(\mathsf{hk}; \mathsf{r})||\mathsf{HInp}_2(\mathsf{hk}, y, \mathsf{state}; \mathsf{r}) = \tilde{y_1}||\tilde{y_2}$ such that

$$(\mathsf{hk}, x, \mathsf{HInp}_1(\mathsf{hk}; \mathsf{r}), \mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, U_{|\mathrm{C}(x)|})) \overset{c}{\approx} (\mathsf{hk}, x, \mathsf{HInp}_1(\mathsf{hk}; \mathsf{r}), U_{|\tilde{\mathrm{C}}|+|\tilde{y_2}|})$$

where $1^l, x \leftarrow \mathsf{Adv}(1^\kappa)$, $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^l)$. (It is clear that the distinguisher should not know about the random output value that was used for simulation)

3.2 Construction of a Blind Hash Garbling Scheme

We require the following building blocks to construct blind hash garbling:

- Blind single batch encryption scheme $(\mathsf{Setup}, \mathsf{H}, \mathsf{HEnc}, \mathsf{HDec})$ as Definition 2.9.
- Blind garbled circuit scheme $(\mathsf{Garble}, \mathsf{Eval})$ as in Definition 2.8.

The blind hash garbling scheme is as follows:

1. $\mathrm{HGen}(1^\kappa, 1^l)$: Generate $\mathsf{hk} \leftarrow \mathsf{Setup}(1^\kappa, 1^l)$ and output hk.
2. $\mathrm{Hash}(\mathsf{hk}, x)$:Generate $\mathsf{H}(\mathsf{hk}, x) = y$ and output y.
3. $\mathrm{HObf}(\mathsf{hk}, \mathrm{C}, \mathsf{state})$:Generate $\tilde{\mathrm{C}} \leftarrow \mathsf{Garble}_2(1^\kappa, 1^l, 1^m, \mathrm{C}; \mathsf{state})$ and output $\tilde{\mathrm{C}}[\mathsf{hk}]$ (circuit $\tilde{\mathrm{C}}$ hardwired with hk)
4. $\mathrm{HInp}(\mathsf{hk}, y, \mathsf{state})$:
   - Generate $\{\mathsf{lab}_{j,b}\}_{j\in[l],b\in\{0,1\}} \leftarrow \mathsf{Garble}_1(1^\kappa, 1^l, 1^m; \mathsf{state})$
   - Generate $\tilde{y} = \{\mathsf{HEnc}(\mathsf{hk}, y, j, [\mathsf{lab}_{j,0}\ \mathsf{lab}_{j,1}])\}_{j\in[l]} = \{\mathsf{subct}_{1,j}, \mathsf{subct}_{2,j}\}_{j\in[l]}$.

   Let $\tilde{y}_1 = \{\mathsf{subct}_{1,j}\}_{j\in[l]}$ and $\tilde{y}_2 = \{\mathsf{subct}_{2,j}\}_{j\in[l]}$. Output $\tilde{y} = (\tilde{y}_1, \tilde{y}_2)$

Theorem 3.2. *The above construction* $(\mathrm{HGen}, \mathrm{Hash}, \mathrm{HObf}, \mathrm{HInp})$ *satisfies the correctness, security and blindess properties as given in Definition 3.1.*

Proof. We now prove the above theorem.

1. **Correctness.** Consider the circuit $\tilde{\mathrm{C}}[\mathsf{hk}](\tilde{y}, x)$:
   - Recovers $\mathsf{lab}_{j,x_j} := \mathsf{HDec}(\mathsf{hk}, x, j, \tilde{y}_j)$ for each $j \in [l]$, where $\tilde{y} = \{\tilde{y}_j\}_{j\in[l]}$.
   - Outputs $\mathsf{Eval}(\tilde{\mathrm{C}}, \{\mathsf{lab}_{j,x_j}\}_{j\in[l]})$.

   Then clearly by correctness of the blind garbling circuit and the correctness of the blind single batch encryption scheme, $\tilde{\mathrm{C}}[\mathsf{hk}](\tilde{y}, x) = \mathrm{C}(x)$
2. **Security.** We define the simulator Sim as below: $\mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, \mathrm{C}(x))$:
   - Evaluate $(\tilde{\mathrm{C}}, \{\mathsf{lab}_j\}_{j\in[l]}) \leftarrow \mathsf{G.Sim}(1^\kappa, 1^{|\mathrm{C}|}, 1^l, \mathrm{C}(x))$.
   - For $j \in [l]$, let $M_j = [M_{j,0}\ M_{j,1}]$, where $M_{j,x_j} = \mathsf{lab}_j$, $M_{j,1-x_j} \in_R \{0,1\}^\kappa$
   - Evaluate $\tilde{y} = \{\mathsf{HEnc}(\mathsf{hk}, \mathsf{H}(\mathsf{hk}, x), j, M_j)\}_{j\in[l]}$. As expressed in the protocol, $\tilde{y} = (\tilde{y}_1, \tilde{y}_2)$.

- Output $(\tilde{\mathrm{C}}, \tilde{y}_2)$

Then, by simulation security of the blind garbled circuit scheme and the semantic security of the blind single batch encryption scheme, it can be shown through a sequence of hybrids that:

$$(\mathsf{hk}, x, \tilde{y}_1, \tilde{\mathrm{C}}, \tilde{y}_2) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, x, \tilde{y}_1, \mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, \mathrm{C}(x))),$$

where $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^l)$, $(\mathrm{C}, x) \leftarrow \mathsf{Adv}(\mathsf{hk})$,$\mathsf{state} \leftarrow \{0,1\}^\kappa$, $\tilde{\mathrm{C}} \leftarrow \mathrm{HObf}(\mathsf{hk}, \mathrm{C}, \mathsf{state})$ and $(\tilde{y}_1, \tilde{y}_2) \leftarrow \mathrm{HInp}(\mathsf{hk}, \mathrm{Hash}(\mathsf{hk}, x), \mathsf{state})$.

3. **Blindness.** For the simulator Sim described above, consider the distribution of $\mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, U_{|\mathrm{C}(x)|})$ for a uniformly generated output.

   - By the blindness of blind garbled circuit, $\mathsf{G.Sim}(1^\kappa, 1^{|\mathrm{C}|}, 1^l, U_{|\mathrm{C}(x)|}) \overset{\mathsf{c}}{\approx} U$.
   - Hence for each $j \in [l]$, $M_j \overset{\mathsf{c}}{\approx} U$. Thus, by blindness of the blind single batch encryption, $\tilde{y} = (\tilde{y}_1, \tilde{y}_2)$ (as described in the protocol), where $\tilde{y}_1$ can be expressed as $\mathrm{HInp}_1(\mathsf{hk}; \mathsf{r})$ and $(\mathsf{hk}, x, \tilde{y}_1, \tilde{y}_2) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, x, \tilde{y}_1, U)$.

   Hence, it follows that $(\mathsf{hk}, x, \tilde{y}_1, \mathsf{Sim}(\mathsf{hk}, x, 1^{|\mathrm{C}|}, U_{|\mathrm{C}(x)|})) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, x, \tilde{y}_1, U)$

4 Efficient Blind T-RBE

We first define and construct an efficient blind T-RBE and then use it to construct an efficient Anonymous RBE scheme.

4.1 Definition

Definition 4.1 (Blind T-RBE)**.** A T-RBE scheme $(\mathrm{TGen}, \mathrm{TReg}, \mathrm{TEnc}, \mathrm{TUpd}, \mathrm{TDec})$ is said to be blind if, in addition to completeness, compactness, efficiency and security properties, as in Definition 2.6, it also satisfies the following blindness property: the function $\mathrm{TEnc}(\mathsf{crs}, \mathsf{pp}, t_{\mathsf{id}}, \mathsf{m}; \mathsf{r})$ can be expressed as the concatenation $\mathrm{TEnc}_1(\mathsf{crs}, \mathsf{pp}; \mathsf{r}) || \mathrm{TEnc}_2(\mathsf{crs}, \mathsf{pp}, t_{\mathsf{id}}, \mathsf{m}; \mathsf{r})$ such that for any PPT adversary Adv, the probability of winning in the following game with a challenger Chal is at most $1/2 + \mathrm{negl}(\kappa)$:

1. **Initialization.** Chal sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{ID} = \varnothing$, $t = 1$, $t^* = \bot$, $\mathsf{id}^* = \bot$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to Adv.

2. Till Adv continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, Adv can perform exactly one of the following actions.

   (a) **Registering new (non-target) identity.** Adv sends some $\mathsf{id} \notin \mathsf{ID}$ and pk to Chal. Chal registers $(\mathsf{id}, \mathsf{pk})$ by getting $\mathsf{pp} = \mathrm{TReg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (t, \mathsf{pk}))$ and sets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}\}$ and $t = t + 1$.

   (b) **Registering new target identity pair.** If id^* was chosen by Adv already (i.e., $\mathsf{id}^* \neq \bot$), skip this step. Otherwise, Adv sends challenge identities $\mathsf{id}^* \notin \mathsf{ID}$ to Chal. Chal first samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathrm{TGen}(1^\kappa)$, registers id^* by setting $\mathsf{pp} = \mathrm{TReg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, (t, \mathsf{pk}^*))$. Next, Chal lets $\mathsf{ID} = \mathsf{ID} \cup \{\mathsf{id}^*\}$, $t = t^*$ and $t = t + 1$, and sends t^*, $\mathsf{pk}^*, \mathsf{sk}^*$ to Adv. (Note that unlike the security property, the secret key *is* given to Adv.)

3. **Encrypting for the challenge identity.** If id^* was not chosen by Adv already (i.e., $\mathsf{id}^* \neq \perp$), then Adv first sends some $\mathsf{id}^* \notin \mathsf{ID}$ to Chal before continuing this step. Next, Chal samples a random message $\mathsf{m} \in_R \mathcal{M}$ and generates $(\mathsf{subct}_1, \mathsf{subct}_2) \leftarrow \mathrm{TEnc}(\mathsf{crs}, \mathsf{pp}, t^*, \mathsf{m}; \mathsf{r})$. It generates a bit $b \in_R \{0,1\}$ and:

   - if $b = 0$, set $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.
   - if $b = 1$, generate a random subct_2' of same length as subct_2 and set $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2')$.

   Chal sends ct to Adv (Note that the Adv does not know the random message m being encrypted).

4. The adversary Adv outputs a bit b' and wins the game if $b' = b$.

4.2 Construction of an Efficient Blind T-RBE scheme

We construct a blind T-RBE scheme $(\mathrm{TGen}, \mathrm{TReg}, \mathrm{TUpd}, \mathrm{TEnc}, \mathrm{TDec})$, as in Definition 4.1, using the following building blocks:

- Blind Hash Garbling scheme $(\mathrm{HGen}, \mathrm{Hash}, \mathrm{HObf}, \mathrm{HInp})$, where the function HInp is expressible as concatenation of function outputs of HInp_1 and HInp_2 as in Definition 3.1.
- Blind Public-key Encryption scheme $(\mathrm{G}, \mathrm{E}, \mathrm{D})$, where the function E is expressible as a concatenation of outputs of E_1 and E_2, as in Definition 2.7.

The subroutines of the T-RBE scheme are defined as follows:

- $\mathrm{TGen}(1^\kappa)$:
  1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{G}(1^\kappa)$
  2. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathrm{TReg}^{[\mathsf{aux}]}(\mathsf{pp}, t_{\mathsf{id}}, \mathsf{pk})$:
  1. aux consists of a family of Merkle trees $\mathcal{T} = \{\mathsf{Tree}_1, \cdots, \mathsf{Tree}_\eta\}$ which are constructed through the process of registration described below. It also consists of a list of timestamps corresponding to each tree in $\mathcal{T}$, TID, arranged in ascending order of timestamps of the identities.
  2. Parse $\mathsf{pp} = (\mathsf{hk}, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$, where $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^{2\kappa + \log n})$ and (rt_i, d_i) represent the root and depth of Tree_i in $\mathcal{T}$.
  3. Updating aux:
     (a) Create $\mathsf{Tree}_{\eta+1}$ with leaves t_{id} and pk and root $\mathsf{rt}_{\eta+1} = \mathrm{Hash}(\mathsf{hk}, t_{\mathsf{id}} || \mathsf{pk} || 0^\kappa)$.
     (b) If there are two trees Tree_L and Tree_R in $\mathcal{T}$ of same depth d then proceed as follows:
     - Let t_L and t_R denote the largest timestamps of Tree_L and Tree_R respectively (can be obtained by reading the last leaf of each tree). WLOG, suppose $t_L < t_R$.
     - Merge the trees, with Tree_L on the left and Tree_R on the right [1], with corresponding roots rt_L and rt_R, to obtain Tree with root $\mathsf{rt} = \mathrm{Hash}(\mathsf{hk}, \mathsf{rt}_L || \mathsf{rt}_R || t_L)$.

[1] This will guarantee that the leaf nodes are sorted in ascending order of the timestamps of the identities.

– Remove Tree_L and Tree_R from $\mathcal{T}$ and add Tree to it.

(c) $\mathsf{TID} := \mathsf{TID} \cup \{t_{\mathsf{id}}\}$.

4. Set $\mathsf{pp}' = (\mathsf{hk}, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\zeta, d_\zeta))$, where (rt_i, d_i) represent the root and depth of Tree_i in updated $\mathcal{T}$.
5. Output pp'.

- $\mathrm{TUpd}^{[\mathsf{aux}]}(\mathsf{pp}, t_{\mathsf{id}})$:

1. Parse $\mathsf{pp} = (\mathsf{hk}, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$
2. Let $\mathsf{u} = \mathsf{pth}$, the Merkle opening from the leaf node t_{id} and its sibling pk to the root rt_i (in Tree_i containing the timestamp t_{id}).
3. Output u.

- $\mathrm{TEnc}(\mathsf{pp}, t_{\mathsf{id}}, m)$:

1. Parse $\mathsf{pp} = (\mathsf{hk}, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$.
2. For each $i = 1, \cdots, \eta$:
   (a) For each $j = 1, \cdots, d_i$:
      – Sample $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\kappa$
      – Generate $\widetilde{\mathrm{P}}_{i,j} \leftarrow \mathrm{HObf}(\mathsf{hk}, \mathrm{P}_{i,j}, \mathsf{state}_{i,j})$
   (b) Obtain $\tilde{y}_{i,1} \leftarrow \mathrm{HInp}(\mathsf{hk}, \mathsf{rt}_i, \mathsf{state}_{i,1})$, where $\tilde{y}_{i,1} = \tilde{y}_{i,1}^{(0)} || \tilde{y}_{i,1}^{(1)}$
   (c) For each $j = 2, \cdots, d_i$, obtain $\tilde{y}_{i,j}^{(0)} = \mathrm{HInp}_1(\mathsf{hk}; \mathsf{r}_{i,j})$
3. Output $\mathsf{ct} = (\mathsf{pp}, \{\widetilde{\mathrm{P}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,j}^{(0)}\}_{i,j}, \{\tilde{y}_{i,1}^{(1)}\}_i, \mathrm{E}_1(\mathsf{r}))$. Let $\mathsf{subct}_1 = (\mathsf{pp}, \{\tilde{y}_{i,j}^{(0)}\}_{i,j}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_2 = (\{\widetilde{\mathrm{P}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,1}^{(1)}\}_i)$. Then $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.

The program $\mathrm{P}_{i,j}$ is as defined below:
Hardwired: $t_{\mathsf{id}}, \mathsf{hk}, \mathsf{state}_{i,j+1}, \mathsf{m}, \mathsf{r}_{i,j+1}$ (where $\mathsf{state}_{i,d_i+1} = \perp, \mathsf{r}_{i,d_i+1} = \mathsf{r}$)
Input: $a||b||t^*$

1. If $t^* = 0^\kappa$ and $a = t_{\mathsf{id}}$, output $\mathrm{E}_2(b, \mathsf{m}; \mathsf{r}_{i,j+1})$
2. If $t^* = 0^\kappa$ and $a \neq t_{\mathsf{id}}$, output $\perp$.
3. If $t_{\mathsf{id}} > t^*$, output $\mathrm{HInp}_2(\mathsf{hk}, b, \mathsf{state}_{i,j+1}; \mathsf{r}_{i,j+1})$
   Else, output $\mathrm{HInp}_2(\mathsf{hk}, a, \mathsf{state}_{i,j+1}; \mathsf{r}_{i,j+1})$

- $\mathrm{TDec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct})$:

1. Parse ct as $(\mathsf{pp}, \{\widetilde{\mathrm{P}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,j}^{(0)}\}_{i,j}, \{\tilde{y}_{i,1}^{(1)}\}_i, \mathrm{E}_1(\mathsf{r}))$.
2. Parse u as $\mathsf{pth} = (z_0 = \mathsf{rt}_{i^*}, z_1, \cdots, z_{d_{i^*}} = t_{\mathsf{id}} || \mathsf{pk} || 0^\kappa)$, the Merkle opening from leaf t_{id} to the root rt_{i^*} of Tree_{i^*} (containing t_{id}).
3. For each $j = 1, \cdots, d_{i^*} - 1$ evaluate:
   – $\tilde{y}_{i^*,j+1}^{(1)} \leftarrow \widetilde{\mathrm{P}}_{i^*,j}(\tilde{y}_{i^*,j}, z_j)$
4. Let $c_2 = \widetilde{\mathrm{P}}_{i^*,d_{i^*}}(\tilde{y}_{i^*,d_{i^*}}, z_{d_{i^*}})$. If $c_2 = \perp$, set $c = \perp$, else set $c = \mathrm{E}_1(\mathsf{r}) || c_2$.

5. If $c = \perp$, output $\texttt{GetUpd}$, else output $\mathrm{D}(\mathsf{sk}, c)$.

Theorem 4.2. *The T-RBE construction 4.2 satisfies the completeness, compactness, efficiency, security and blindness (Definition 4.1) properties.*

In the following subsections, we prove Theorem 4.2.

4.3 Proofs of Completeness, Compactness and Efficiency of the T-RBE Construction

Completeness. By the correctness of the hash garbling scheme and the Public-key Encryption scheme, completeness property follows.

Compactness of public parameters and update. Consider the public parameter $\mathsf{pp} = (\mathsf{hk}, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$. We observe that:

- The number of Merkle trees, η, in $\mathcal{T}$ at any time is at most $\log(n)$. This is because the trees are full binary trees and the size of the trees are always different (as we keep merging in the registration process).
- The hash key hk, the root and the depth of each tree are all of size κ each.

Hence, the size of pp is $O(\kappa.\log n)$.
Consider the update $\mathsf{u} = \mathsf{pth}$, the Merkle opening from leaf node t_{id} ad its sibling pk to the root rt_i. The depth of the tree is d_i and hence, there are at most $2.d_i + 1$ nodes in the Merkle opening, where $d_i \leq \kappa$. Hence, the size of u is at most $O((2.\kappa + \log n).\kappa) = O(\mathrm{poly}(\kappa, \log(n)))$.
Efficiency of runtime of registration and update. The registration process involves evaluating a hash value to create a new tree with the new identity and then merging the trees of same depth after this. We observe that:

- The number of merge operations is $O(\log n)$ as the number of trees is always logarithmic.
- Computing each hash costs $O(\kappa)$.

Hence, each invocation of the registration process takes time $O(\kappa.\log n)$. Consider a single invocation of the update. This just involves reading aux to output the Merkle opening required and this takes time $O((2.\kappa + \log n)\kappa) = O(\mathrm{poly}(\kappa, \log n))$.
Efficiency of the number of updates. Each identity would require to invoke Upd, whenever the Merkle opening for the id gets modified. This in turn happens whenever two tress are merged. Since the number of merges is at most $O(\log n)$, the total number of invocations of Upd by each identity is at most $O(\log n)$.

4.4 Proof of Security of the T-RBE construction

We prove the security assuming that there is only one tree at the time of encryption. The proof for the case of multiple trees will be the same.

Proof. Suppose that at the time of encryption, the underlying tree has root rt and depth d. For simplicity, for each $j \in [d]$, we denote the circuits $\mathrm{P}_{1,j}$ by P_j and the state used for obfuscation, $\mathsf{state}_{1,j}$ by state_j, i.e., for each $j \in [d]$

$$\mathrm{P}_j \equiv \mathrm{P}_{1,j}[t_{\mathsf{id}}, \mathsf{hk}, \mathsf{state}_{j+1}, \mathsf{m}, \mathsf{r}_{1,j+1}]$$

where all the variables are as in the encryption algorithm of the construction.
As in the construction, let the Merkle opening from the leaf node t_{id} and its sibling pk to the root rt be denoted by:

$$\mathsf{pth} = ((t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), (a_1, b_1, t_1), \cdots, (a_{d-1}, b_{d-1}, t_{d-1}), \mathsf{rt})$$

As in the decryption algorithm of the construction, we denote the hash-obfuscation of the inputs of the circuits by $\tilde{y}_j \equiv \tilde{y}_{1,j} = (\tilde{y}^{(0)}_{1,j}, \tilde{y}^{(1)}_{1,j})$ for each $j \in [d]$. Then, in the actual game, the output of the encryption algorithm is $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \tilde{y}^{(0)}_1, \cdots, \tilde{y}^{(0)}_d, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\widetilde{\mathrm{P}}_1, \cdots, \widetilde{\mathrm{P}}_d, \tilde{y}^{(1)}_1)$ (as in the protocol we have two parts of the ciphertext).
We describe the following sequence of hybrids, where we first replace the garbled versions of the programs P_j and the corresponding garbled inputs $\tilde{y}_j$ by their simulated variants, which do not use state_j, one by one.

- $\mathbf{Hybrid}_0$ (**encryption in real game**): The ciphertext here will be $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \tilde{y}^{(0)}_1, \cdots, \tilde{y}^{(0)}_d, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\widetilde{\mathrm{P}}_1, \cdots, \widetilde{\mathrm{P}}_d, \tilde{y}^{(1)}_1)$, are as described above.

- $\mathbf{Hybrid}_1$: We replace the first obfuscated program $\tilde{\mathrm{P}}_1$ with its simulated form. $\tilde{\mathrm{P}}_2, \cdots, \tilde{\mathrm{P}}_d$ are sampled as in the construction. Let $\tilde{\mathrm{P}}_{1,\mathsf{sim}}$ and $\tilde{y}^{(1)}_{1,\mathsf{sim}}$ be sampled as:

  $$(\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \tilde{y}^{(1)}_{1,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-1}, b_{d-1}, t_{d-1}), 1^{|\mathrm{P}_1|}, \tilde{y}^{(1)}_2)$$

  Then, the ciphertext in this hybrid is $\mathsf{ct}_1 := (\mathsf{subct}_{1,1}, \mathsf{subct}_{1,2})$, where $\mathsf{subct}_{1,1} = (\mathsf{pp}, \tilde{y}^{(0)}_1, \cdots, \tilde{y}^{(0)}_d, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{1,2} = (\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \tilde{\mathrm{P}}_2, \cdots, \tilde{\mathrm{P}}_d, \tilde{y}^{(1)}_{1,\mathsf{sim}})$.

- $\mathbf{Hybrid}_i$, for each $i \in [d-1]$: The ciphertext is $\mathsf{ct}_i := (\mathsf{subct}_{i,1}, \mathsf{subct}_{i,2})$, where $\mathsf{subct}_{i,1} = (\mathsf{pp}, \tilde{y}^{(0)}_1, \cdots, \tilde{y}^{(0)}_d, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{i,2} = (\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{P}}_{i,\mathsf{sim}}, \tilde{\mathrm{P}}_{i+1}, \cdots, \tilde{\mathrm{P}}_d, \tilde{y}^{(1)}_{1,\mathsf{sim}})$ where for each $j \in [i]$:

  $$(\tilde{\mathrm{P}}_{j,\mathsf{sim}}, \tilde{y}^{(1)}_{j,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-j}, b_{d-j}, t_{d-j}), 1^{|\mathrm{P}_j|}, \tilde{y}^{(1)}_{j+1})$$

- $\mathbf{Hybrid}_d$: The ciphertext is $\mathsf{ct}_d := (\mathsf{subct}_{d,1}, \mathsf{subct}_{d,2})$, where $\mathsf{subct}_{d,1} = (\mathsf{pp}, \tilde{y}^{(0)}_1, \cdots, \tilde{y}^{(0)}_d, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{d,2} = (\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}^{(1)}_{1,\mathsf{sim}})$ where for each $j \in [d-1]$

  $$(\tilde{\mathrm{P}}_{j,\mathsf{sim}}, \tilde{y}^{(1)}_{j,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-j}, b_{d-j}, t_{d-j}), 1^{|\mathrm{P}_j|}, \tilde{y}^{(1)}_{j+1})$$

  and

  $$(\tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}^{(1)}_{d,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), 1^{|\mathrm{P}_d|}, \mathrm{E}_2(\mathsf{pk}, \mathsf{m}; \mathsf{r}))$$

By the simulation security of the hash garbling scheme, we know that, for each $j \in [d-1]$,

$$(\mathsf{hk}, (a_{d-j}, b_{d-j}, t_{d-j}), \tilde{y}^{(0)}_j, \tilde{\mathrm{P}}_j, \tilde{y}^{(1)}_j) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (a_{d-j}, b_{d-j}, t_{d-j}), \tilde{y}^{(0)}_j, \tilde{\mathrm{P}}_{j,\mathsf{sim}}, \tilde{y}^{(1)}_{j,\mathsf{sim}})$$

and

$$(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), \tilde{y}^{(0)}_d, \tilde{\mathrm{P}}_d, \tilde{y}^{(1)}_d) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), \tilde{y}^{(0)}_d, \tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}^{(1)}_{d,\mathsf{sim}})$$

Hence, it follows that for each $i = 0, \cdots, d-1$, $\mathbf{Hybrid}_i \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{i+1}$.
Now, let $\mathbf{Hybrid}^0_i$ denote the hybrids described above with use of underlying message m_0 and

$\mathbf{Hybrid}_i^1$ for message m_1. By semantic security of the underlying public-key encryption scheme, we get:

$$\mathsf{Sim}(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), 1^{|\,\mathrm{P}_d\,|}, \mathrm{E}_2(\mathsf{pk}, \mathsf{m}_0; \mathsf{r})) \overset{\mathsf{c}}{\approx} \mathsf{Sim}(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), 1^{|\,\mathrm{P}_d\,|}, \mathrm{E}_2(\mathsf{pk}, \mathsf{m}_1; \mathsf{r}))$$

Hence, $\mathbf{Hybrid}_d^0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_d^1$. Then, it follows that $\mathbf{Hybrid}_0^0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_0^1$, which represent the actual security game with use of messages m_0 and m_1 in respective hybrids. Hence, the security of the T-RBE scheme is proved. □

4.5 Proof of Blindness of the T-RBE Construction

We prove the blindness of the scheme assuming that there is only one tree at the time of encryption for simplicity. The proof for the case of multiple trees will be the same.

Proof. Suppose that at the time of encryption, the underlying tree has root rt and depth d. For simplicity, for each $j \in [d]$, we denote the circuits $\mathrm{P}_{1,j}$ by P_j and the state used for obfuscation, $\mathsf{state}_{1,j}$ by state_j, i.e., for each $j \in [d]$

$$\mathrm{P}_j \equiv \mathrm{P}_{1,j}[t_{\mathsf{id}}, \mathsf{hk}, \mathsf{state}_{j+1}, \mathsf{m}, \mathsf{r}_{1,j+1}]$$

where all the variables are as in the encryption algorithm of the construction.
As in the construction, let the Merkle opening from the leaf node t_{id} and its sibling pk to the root rt be denoted by:

$$\mathsf{pth} = ((t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), (a_1, b_1, t_1), \cdots, (a_{d-1}, b_{d-1}, t_{d-1}), \mathsf{rt}) \ .$$

As in the decryption algorithm of the construction, we denote the hash-obfuscation of the inputs of the circuits by $\tilde{y}_j \equiv \tilde{y}_{1,j} = (\tilde{y}_{1,j}^{(0)}, \tilde{y}_{1,j}^{(1)})$ for each $j \in [d]$. Then, in the actual game, where now the message m is chosen at random, the output of the encryption algorithm is $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \tilde{y}_1^{(0)}, \cdots, \tilde{y}_d^{(0)}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\widetilde{\mathrm{P}}_1, \cdots, \widetilde{\mathrm{P}}_d, \tilde{y}_1^{(1)})$ (as in the protocol we have two parts of the ciphertext). Clearly $\mathsf{subct}_{0,1}$ is expressible as $\mathrm{TEnc}_1(\mathsf{pp}; \mathsf{r})$.
We describe the following sequence of hybrids, where we first follow the sequence of hybrids in the security proof of Section 4.4 to replace all the garbled versions of the programs P_j and the corresponding garbled inputs $\tilde{y}_j$ by their simulated variants. Then we use the blindness property of the blind hash garbling scheme and the blind public key encryption scheme to replace the simulated garbled circuits with uniform.

- $\mathbf{Hybrid}_0$ (**encryption in real blindness game**): The ciphertext in this hybrid will be $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \tilde{y}_1^{(0)}, \cdots, \tilde{y}_d^{(0)}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\widetilde{\mathrm{P}}_1, \cdots, \widetilde{\mathrm{P}}_d, \tilde{y}_1^{(1)})$, as described above.
- $\mathbf{Hybrid}_1$: The ciphertext is $\mathsf{ct}_1 := (\mathsf{subct}_{1,1}, \mathsf{subct}_{1,2})$, where $\mathsf{subct}_{1,1} = (\mathsf{pp}, \tilde{y}_1^{(0)}, \cdots, \tilde{y}_d^{(0)}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{1,2} = (\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}_{1,\mathsf{sim}}^{(1)})$ where for each $j \in [d-1]$

  $$(\tilde{\mathrm{P}}_{j,\mathsf{sim}}, \tilde{y}_{j,\mathsf{sim}}^{(1)}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-j}, b_{d-j}, t_{d-j}), 1^{|\,\mathrm{P}_j\,|}, \tilde{y}_{j+1}^{(1)})$$

  and

  $$(\tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}_{d,\mathsf{sim}}^{(1)}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), 1^{|\,\mathrm{P}_d\,|}, \mathrm{E}_2(\mathsf{pk}, \mathsf{m}; \mathsf{r})) \ .$$

- **Hybrid**$_i$ for each $i = 2, \cdots, d$: The ciphertext is $\mathsf{ct}_i := (\mathsf{subct}_{i,1}, \mathsf{subct}_{i,2})$, where $\mathsf{subct}_{i,1} = (\mathsf{pp}, \tilde{y}_1^{(0)}, \cdots, \tilde{y}_d^{(0)}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{i,2} = (\tilde{\mathrm{P}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{P}}_{d-(i-1),\mathsf{sim}}, U_{|\tilde{\mathrm{P}}_{d-(i-2)}|}, \cdots, U_{|\tilde{\mathrm{P}}_d|}, \tilde{y}_{1,\mathsf{sim}}^{(1)})$.
- **Hybrid**$_{d+1}$: The ciphertext is $\mathsf{ct}_{d+1} := (\mathsf{subct}_{d+1,1}, \mathsf{subct}_{d+1,2})$, where $\mathsf{subct}_{d+1,1} = (\mathsf{pp}, \tilde{y}_1^{(0)}, \cdots, \tilde{y}_d^{(0)}, \mathrm{E}_1(\mathsf{r}))$ and $\mathsf{subct}_{d+1,2} = (U_{|\tilde{\mathrm{P}}_1|+|\tilde{y}_1^{(1)}|}, U_{|\tilde{\mathrm{P}}_2|}, \cdots, U_{|\tilde{\mathrm{P}}_d|})$.

By the proof of security in Section 4.4, we know that $\mathbf{Hybrid}_0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_1$. Now, by blindness property of the blind public key encryption scheme, we know that:

$$(\mathsf{pk}, \mathsf{sk}, \mathrm{E}_1(\mathsf{r}), \mathrm{E}_2(\mathsf{pk}, \mathsf{m}; \mathsf{r})) \overset{\mathsf{c}}{\approx} (\mathsf{pk}, \mathsf{sk}, \mathrm{E}_1(\mathsf{r}), U) \ .$$

Hence, by the blindness property of the hash garbling scheme, it follows that:

$$(\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), \mathsf{sk}, \tilde{y}_d^{(0)}, \tilde{\mathrm{P}}_{d,\mathsf{sim}}, \tilde{y}_{d,\mathsf{sim}}^{(1)}) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (t_{\mathsf{id}}, \mathsf{pk}, 0^\kappa), \mathsf{sk}, \tilde{y}_d^{(0)}, U_{|\mathrm{P}_d|+|\tilde{y}_d^{(1)}|}) \ .$$

Hence, it follows that $\mathbf{Hybrid}_1 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_2$ even given the secret key sk.
By consecutive use of blindness property of the hash garbling scheme, it would follow that for each $i = 2, \cdots, d$, $\mathbf{Hybrid}_i \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{i+1}$ which holds even given the secret key sk.
Hence, we have that $\mathbf{Hybrid}_0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{d+1}$, even given the secret key sk, which exactly represents the blindness game. This completes the proof of blindness of the T-RBE scheme. □

5 Anonymous Registration-based Encryption

We now construct an efficient anonymous RBE scheme as in Definition 2.5.

5.1 Construction of an Efficient Anonymous RBE scheme

We now construct a RBE scheme $(\mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ using the following building blocks:

- Blind Hash Garbling scheme $(\mathsf{HGen}, \mathsf{Hash}, \mathsf{HObf}, \mathsf{HInp})$, where the function HInp is expressible as concatenation of function outputs of HInp_1 and HInp_2, as in Definition 3.1.
- Blind T-RBE scheme $(\mathsf{TGen}, \mathsf{TReg}, \mathsf{TEnc}, \mathsf{TUpd}, \mathsf{TDec})$, where the function TEnc is expressible as the concatenation of function outputs of TEnc_1 and TEnc_2, as in Definition 4.1.

The subroutines of our RBE scheme are defined as follows:

- $\mathrm{Gen}(1^\kappa)$:
  1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{TGen}(1^\kappa)$
  2. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathrm{Reg}^{[\mathsf{aux}]}(\mathsf{pp}, \mathsf{id}, \mathsf{pk})$:
  1. aux consists of a Red-black Merkle tree $\mathsf{TimeTree}$ which has the identities (along with their timestamps) at the leaf nodes, sorted according to the identities. $\mathsf{TimeTree}$ is constructed through the process of registration, as described below. aux also consists of the database aux_T required by the T-RBE scheme and a list ID of identities registered so far, in ascending order. Let the Merkle trees contained in aux_T be $\mathcal{T} = \{\mathsf{Tree}_1, \cdots, \mathsf{Tree}_\eta\}$.

2. Parse pp as $(\mathsf{hk}, \mathsf{pp}_T, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$, where $\mathsf{hk} \leftarrow \mathrm{HGen}(1^\kappa, 1^{3\kappa})$, pp_T corresponds to the public parameter of the T-RBE scheme and $(\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta)$ are the roots and depth of the same $\mathsf{TimeTree}$ corresponding to the time when $\mathsf{Tree}_1, \cdots, \mathsf{Tree}_\eta$ where last updated respectively.[2]
3. Updating $\mathsf{TimeTree}$:
   (a) $\mathsf{TimeTree}$ is a Red-Black Merkle tree with each non-leaf node containing a hash of its left child, hash of its right child and the largest identity on the left subtree. The leaves of the tree contain the identities (with their timestamps, i.e., a binary representation of when the identity registered). Further, each node has an additional bit of information, indicating if it's colored red or black. The color helps in keeping the tree "approximately" balanced after each insertion [3].
   (b) Evaluate $h_{\mathsf{id}} = \mathrm{Hash}(\mathsf{hk}, \mathsf{id}||0^{2\kappa - \log n}||t_{\mathsf{id}})$.
   (c) To insert the new id at the right location, we first parse the root of $\mathsf{TimeTree}$ as $\mathsf{rt} = h_1||\mathsf{id}^*||h_2$. If $\mathsf{id} > \mathsf{id}^*$, read the right child, else read the left child. Continue traversing the path down the tree to figure out the correct insertion point of id.
   (d) The parent node of $\mathsf{id}||t_{\mathsf{id}}$ contains h_{id} along with the largest identity on its left subtree and the hash of its other child. Re-order the tree, recolor, to keep it balanced. This will take at most $\log n$ time (as it's a red-black tree). Every node that is not a leaf or a parent node of a leaf node, is of the form $h_L||id^*||h_R$, where $h_L = \mathrm{Hash}(\mathsf{hk}, h_1||id_L||h_2)$ and $h_R = \mathrm{Hash}(\mathsf{hk}, h_3||id_R||h_4)$ and id^* is the largest identity on the left subtree of this node.
4. Evaluate $\mathsf{pp}'_T \leftarrow \mathrm{TReg}^{[\mathsf{aux}]}(\mathsf{pp}_T, t_{\mathsf{id}}, \mathsf{pk})$, where t_{id} will be the binary representation of the timestamp corresponding to id (can be obtained by checking ID to see the current number of identities).
5. Suppose the registration process of T-RBE above results in a merge of trees Tree_L and Tree_R and the corresponding root hashes of $\mathsf{TimeTree}$ at the time of last update of these trees be rt_L and rt_R. Remove (rt_L, d_L) and (rt_R, d_R) from pp and add (rt, d), which is the root hash and the depth of $\mathsf{TimeTree}$ at time t_{id}. Suppose the updated root hashes and depth of $\mathsf{TimeTree}$ after all the merges in $\mathcal{T}$ be $(\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\zeta, d_\zeta)$.
6. Set $\mathsf{pp}' = (\mathsf{hk}, \mathsf{pp}'_T, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\zeta, d_\zeta))$.
7. Output pp'.

- $\mathrm{Upd}^{[\mathsf{aux}]}(\mathsf{pp}, \mathsf{id})$:
  1. Parse $\mathsf{pp} = (\mathsf{hk}, \mathsf{pp}_T, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$.
  2. Evaluate $\mathsf{u}_1 \leftarrow \mathrm{TUpd}^{[\mathsf{aux}]}(\mathsf{pp}_T, t_{\mathsf{id}})$.
  3. Set pth to be the path from the leaf node $\mathsf{id}||t_{\mathsf{id}}||0^{2\kappa - \log n}$ to the root hash rt_i of $\mathsf{TimeTree}$ at the time of last modification of the Merkle tree, Tree_i, containing id.[4]

[2] Looking ahead, we need to store the root hashes of $\mathsf{TimeTree}$ at mulitple times in order to ensure that the number of updates required by each person remains $\log n$.

[3] The main advantage of having a Red-Black Merkle tree is that after each insertion, the depth of the tree does not increase beyond $\log n$, where n is the number of people registered in the system. The balancing is not perfect, but ensures that further insertions, rearrangement after insertion to balance, searches, all take time $O(\log n)$.

[4] Note that we must store the versions of the same $\mathsf{TimeTree}$ at times corresponding to last updation of each Tree_i in $\mathcal{T}$. But there would only be $\log n$ such versions.

4. Set $\mathsf{u} = (\mathsf{u}_1, \mathsf{pth})$.
5. Output u.

- $\mathrm{Enc}(\mathsf{pp}, \mathsf{id}, \mathsf{m})$:

1. Parse pp as $(\mathsf{hk}, \mathsf{pp}_T, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$.
2. For each $i = 1, \cdots, \eta$:
   (a) For each $j = 1, \cdots, d_\eta$:
      i. Sample $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\kappa$.
      ii. Generate $\tilde{\mathrm{Q}}_{i,j} \leftarrow \mathrm{HObf}(\mathsf{hk}, \mathrm{Q}_{i,j}, \mathsf{state}_{i,j})$
   (b) Parse rt_i as $h_1 || \mathsf{id}^* || h_2$.
   (c) If $\mathsf{id} > \mathsf{id}^*$, obtain $\tilde{y}_{i,1} \leftarrow \mathrm{HInp}(\mathsf{hk}, h_2, \mathsf{state}_1)$, where $\tilde{y}_{i,1} = \tilde{y}^{(0)}_{i,1} || \tilde{y}^{(1)}_{i,1}$
   Else obtain $\tilde{y}_{i,1} \leftarrow \mathrm{HInp}(\mathsf{hk}, h_1, \mathsf{state}_1)$, where $\tilde{y}_{i,1} = \tilde{y}^{(0)}_{i,1} || \tilde{y}^{(1)}_{i,1}$.
   (d) For each $j = 2, \cdots, d_i$, obtain $\tilde{y}^{(0)}_{i,j} \leftarrow \mathrm{HInp}_1(\mathsf{hk}; \mathsf{r}_{i,j})$.
3. Output $\mathsf{ct} = (\mathsf{pp}, \{\tilde{\mathrm{Q}}_{i,j}\}_{i,j}, \{\tilde{y}^{(0)}_{i,j}\}_{i,j}, \{\tilde{y}^{(1)}_{i,1}\}_i, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$.
   Let $\mathsf{subct}_1 = (\mathsf{pp}, \{\tilde{y}^{(0)}_{i,j}\}_{i,j}, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_2 = (\{\tilde{\mathrm{Q}}_{i,j}\}_{i,j}, \{\tilde{y}^{(1)}_{i,1}\}_i)$. Then $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.

The program $\mathrm{Q}_{i,j}$ is defined as:
Hardwired: $\mathsf{hk}, \mathsf{state}_{i,j+1}, \mathsf{id}, \mathsf{m}, \mathsf{r}_{i,j+1}, \mathsf{r}, \mathsf{pp}_T$ (where $\mathsf{state}_{i,d_i+1} = \bot$, $\mathsf{r}_{i,d_i+1} = \bot$)
Input: $a || \mathsf{id}^* || b$

1. If $\mathsf{id}^* = 0^{2\kappa - \log n}$ and $a = \mathsf{id}$, output $\mathrm{TEnc}_2(\mathsf{pp}_T, b, \mathsf{m}; \mathsf{r})$
2. If $\mathsf{id}^* = 0^{2\kappa - \log n}$ and $a \neq \mathsf{id}$, output $\bot$
3. If $\mathsf{id} > \mathsf{id}^*$, output $\mathrm{HInp}_2(\mathsf{hk}, b, \mathsf{state}_{i,j+1}; \mathsf{r}_{i,j+1})$
   Else, output $\mathrm{HInp}_2(\mathsf{hk}, a, \mathsf{state}_{i,j+1}; \mathsf{r}_{i,j+1})$

- $\mathrm{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct})$:

1. Parse u as $(\mathsf{u}_1, \mathsf{pth})$, where $\mathsf{pth} = (z_0 = \mathsf{rt}_i, z_1, \cdots, z_d = \mathsf{id} || 0^{2\kappa - \log n} || t_{\mathsf{id}})$. Here rt_i is the root hash of $\mathsf{TimeTree}$ at the time when Tree_i, the tree containing id, was last updated.
2. Parse ct as $(\mathsf{pp}, \{\tilde{\mathrm{Q}}_{i,j}\}_{i,j}, \{\tilde{y}^{(0)}_{i,j}\}_{i,j}, \{\tilde{y}^{(1)}_{i,1}\}_i, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$.
3. For $j = 1, \cdots, d-1$, evaluate:
   – $\tilde{y}^{(1)}_{i,j+1} \leftarrow \tilde{\mathrm{Q}}_{i,j}(\tilde{y}_{i,j}, z_j)$
4. Let $c_2 = \tilde{\mathrm{Q}}_{i,d}(\tilde{y}_{i,d}, z_d)$. If $c_2 = \bot$, set $c = \bot$, else set $c = \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}) || c_2$.[5]
5. If $c = \bot$, output $\mathtt{GetUpd}$, else output $\mathrm{TDec}(\mathsf{sk}, \mathsf{u}_1, c)$.

Theorem 5.1. *The RBE construction 5.1 satisfies the completeness, compactness, efficiency (Definition 2.2) and the stronger security notion of an anonymous RBE (Definition 2.5) properties.*

In the following subsections, we prove Theorem 5.1.

[5] Alternately, we could have performed these operations for each i, which would be the number of trees in $\mathcal{T}$. Here, we would have obtained a value $\neq \bot$ only for one i

5.2 Completeness, Compactness and Efficiency of the RBE Construction

Completeness. By the correctness of the hash garbling scheme and the completeness of the T-RBE scheme, completeness property follows.

Compactness of public parameters and update. Consider the public parameter $\mathsf{pp} = (\mathsf{hk}, \mathsf{pp}_T, (\mathsf{rt}_1, d_1), \cdots, (\mathsf{rt}_\eta, d_\eta))$. We observe that:

- The public parameter of T-RBE, pp_T is of size $O(\text{poly}(\kappa, \log n))$.
- The hash key hk and the depth d_i are each of size κ.
- The root of the time tree rt_i is of size $3.\kappa$.
- η is at any time is atmost $\log(n)$, as the number of trees in $\mathcal{T}$ at any time is at most $\log(n)$.

Hence, the size of pp is $O(\text{poly}(\kappa, \log n))$.

Consider the update $\mathsf{u} = (\mathsf{u}_1, pth)$. By efficiency of T-RBE, size of u_1 is $O(\text{poly}(\kappa, \log n))$. pth is the path of nodes from the leaf $\mathsf{id}||t_{\mathsf{id}}||0^{2\kappa - \log n}$ to the root rt and hence of size at most $\kappa.(3\kappa) = O(\kappa)$. Hence, the size of u is
$O(\text{poly}(\kappa, \log n))$.

Efficiency of runtime of registration and update. The registration process involves running the registration of underlying T-RBE, which takes time at most $\text{poly}(\kappa, \log n)$ and inserting the new identity into the Red-black Merkle tree $\mathsf{TimeTree}$, which takes time at most $\log n$. Hence, each invocation of the registration process takes time $O(\text{poly}(\kappa, \log n))$.
Consider a single invocation of the update process. It involves a single invocation of the update algorithm of T-RBE, which takes time at most $\text{poly}(\kappa, \log n)$ and reading a single path from $\mathsf{TimeTree}$ in aux, which takes time $O((3.\kappa).\kappa)$ and an additional $\log(n)$ time to figure out the correct version of $\mathsf{TimeTree}$ to read the path from. Hence a single invocation of the update algorithm takes time $O(\text{poly}(\kappa, \log n))$.

Efficiency of the number of updates. Each identity would require to invoke Upd, whenever the Merkle tree in $\mathcal{T}$ containing id gets modified (by a merge). Unless a merge operation occurs, the identities can use the opening in the $\mathsf{TimeTree}$ corresponding to the time its Merkle tree (in $\mathcal{T}$) was last updated (even though $\mathsf{TimeTree}$ gets updated at each registration). Since the number of merges is at most $O(\log(n))$, the number of invocations of Upd by each identity is at most $O(\log n)$.

5.3 Proof of Anonymity and Security of the RBE construction

We now prove that the RBE scheme satisfies the stronger notion of security of an anonymous RBE (Definition 2.5).

Proof. We prove the security assuming that there is only one Merkle tree in $\mathcal{T}$ at the time of encryption for simplicity. The proof for the case of multiple trees will be the same. Let $\mathsf{TimeTree}$ have root rt and let the challenge identity id be at depth d in $\mathsf{TimeTree}$, at the time of encryption. For each $j \in [d]$, as in the construction, we have:

$$\mathrm{Q}_j \equiv \mathrm{Q}_{1,j}[\mathsf{hk}, \mathsf{state}_{1,j+1}, \mathsf{id}, \mathsf{m}, \mathsf{r}_{1,j+1}, \mathsf{r}, \mathsf{pp}_T]$$

where all the variables are as in the encryption algorithm of the construction.
As in the construction, let the path from the leaf node $\mathsf{id}||0^{2\kappa-\log n}||t_{\mathsf{id}}$ to the root rt be denoted by:

$$\mathsf{pth} = ((\mathsf{id}, 0^{2\kappa-\log n}, t_{\mathsf{id}}), (a_1, \mathsf{id}_1, b_1), \cdots, (a_{d-1}, \mathsf{id}_{d-1}, b_{d-1}), \mathsf{rt})$$

As in the decryption algorithm of the construction, we denote the hash-garbling of the inputs of the circuits by $\tilde{y}_j = \tilde{y}^{(0)}_{1,j}||\tilde{y}^{(1)}_{1,j}$ for each $j \in [d]$. Then, in the actual game, the output of the encryption algorithm is $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\{\tilde{\mathrm{Q}}_j\}_j, \tilde{y}^{(1)}_1)$.
We describe the following sequence of hybrids, *for uniformly drawn message* m, where we first replace the garbled versions of the programs Q_j and the corresponding garbled inputs $\tilde{y}_j$ by their simulated variants, which do not use state_j, one by one. Then, in the subsequent hybrids, we replace the simulated garbled programs and inputs by uniform, one by one. This uses the blindness of the T-RBE scheme and the blind hash garbling scheme.

- **Hybrid**$_0$ (**encryption in real game**): The ciphertext in this hybrid will be $\mathsf{ct}_0 := (\mathsf{subct}_{0,1}, \mathsf{subct}_{0,2})$, where $\mathsf{subct}_{0,1} = (\mathsf{pp}, \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{0,2} = (\tilde{\mathrm{Q}}_1, \cdots, \tilde{\mathrm{Q}}_d, \tilde{y}^{(1)}_1)$, as described above.

- **Hybrid**$_1$: We replace the first obfuscated program $\tilde{\mathrm{Q}}_1$ with its simulated form. $\tilde{\mathrm{Q}}_2, \cdots, \tilde{\mathrm{Q}}_d$ are sampled as in the construction. Let $\tilde{\mathrm{Q}}_{1,\mathsf{sim}}$ and $\tilde{y}^{(1)}_{1,\mathsf{sim}}$ be sampled as:

  $$(\tilde{\mathrm{Q}}_{1,\mathsf{sim}}, \tilde{y}^{(1)}_{1,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-1}, \mathsf{id}_{d-1}, b_{d-1}), 1^{|\mathrm{Q}_1|}, \tilde{y}^{(1)}_2)$$

  Then, the ciphertext in this hybrid is $\mathsf{ct}_1 := (\mathsf{subct}_{1,1}, \mathsf{subct}_{1,2})$, where $\mathsf{subct}_{1,1} = (\mathsf{pp}, , \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{1,2} = (\tilde{\mathrm{Q}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{Q}}_d, \tilde{y}^{(1)}_{1,\mathsf{sim}})$.

- **Hybrid**$_i$, for each $i \in [d-1]$: The ciphertext is $\mathsf{ct}_i := (\mathsf{subct}_{i,1}, \mathsf{subct}_{i,2})$, where $\mathsf{subct}_{i,1} = (\mathsf{pp}, \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{i,2} = (\tilde{\mathrm{Q}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{Q}}_{i,\mathsf{sim}}, \tilde{\mathrm{Q}}_{i+1}, \cdots, \tilde{\mathrm{Q}}_d, \tilde{y}^{(1)}_{1,\mathsf{sim}})$ where for each $j \in [i]$:

  $$(\tilde{\mathrm{Q}}_{j,\mathsf{sim}}, \tilde{y}^{(1)}_{j,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-j}, \mathsf{id}_{d-j}, b_{d-j}), 1^{|\mathrm{Q}_j|}, \tilde{y}^{(1)}_{j+1})$$

- **Hybrid**$_d$: The ciphertext is $\mathsf{ct}_d := (\mathsf{subct}_{d,1}, \mathsf{subct}_{d,2})$, where $\mathsf{subct}_{d,1} = (\mathsf{pp}, \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{d,2} = (\tilde{\mathrm{Q}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{Q}}_{d,\mathsf{sim}}, \tilde{y}^{(1)}_{1,\mathsf{sim}})$ where for each $j \in [d-1]$

  $$(\tilde{\mathrm{Q}}_{j,\mathsf{sim}}, \tilde{y}^{(1)}_{j,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (a_{d-j}, \mathsf{id}_{d-j}, b_{d-j}), 1^{|\mathrm{Q}_j|}, \tilde{y}^{(1)}_{j+1})$$

  and

  $$(\tilde{\mathrm{Q}}_{d,\mathsf{sim}}, \tilde{y}^{(1)}_{d,\mathsf{sim}}) \leftarrow \mathsf{Sim}(\mathsf{hk}, (\mathsf{id}, 0^{2\kappa-\log n}, t_{\mathsf{id}}), 1^{|\mathrm{Q}_d|}, \mathrm{TEnc}_2(\mathsf{pp}_T, t_{\mathsf{id}}, \mathsf{m}; \mathsf{r}))$$

- **Hybrid**$_i$ for each $i = d+1, \cdots, 2d-1$: The ciphertext is $\mathsf{ct}_i := (\mathsf{subct}_{i,1}, \mathsf{subct}_{i,2})$, where $\mathsf{subct}_{i,1} = (\mathsf{pp}, \{\tilde{y}^{(0)}_j\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{i,2} = (\tilde{\mathrm{Q}}_{1,\mathsf{sim}}, \cdots, \tilde{\mathrm{Q}}_{2d-i,\mathsf{sim}}, U_{|\tilde{\mathrm{Q}}_{2d-i+1}|}, \cdots, U_{|\tilde{\mathrm{Q}}_d|}, \tilde{y}^{(1)}_{1,\mathsf{sim}})$

- **Hybrid**$_{2d}$: The ciphertext is
  $\mathsf{ct}_{2d} := (\mathsf{subct}_{2d,1}, \mathsf{subct}_{2d,2})$, where $\mathsf{subct}_{2d,1} = (\mathsf{pp}, \{\tilde{y}_j^{(0)}\}_j, \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}))$ and $\mathsf{subct}_{2d,2} = (U_{|\tilde{\mathsf{Q}}_1|+|\tilde{y}_1^{(1)}|}, U_{|\tilde{\mathsf{Q}}_2|}, \cdots, U_{|\tilde{\mathsf{Q}}_d|})$

By the simulation security of the hash garbling scheme, we know that, for each $j \in [d-1]$,

$$(\mathsf{hk}, (a_{d-j}, \mathsf{id}_{d-j}, b_{d-j}), \tilde{y}_j^{(0)}, \tilde{\mathsf{Q}}_j, \tilde{y}_j^{(1)}) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (a_{d-j}, \mathsf{id}_{d-j}, b_{d-j}), \tilde{y}_j^{(0)}, \tilde{\mathsf{Q}}_{j,\mathsf{sim}}, \tilde{y}_{j,\mathsf{sim}}^{(1)})$$

and

$$(\mathsf{hk}, (\mathsf{id}, 0^{2\kappa - \log n}, t_{\mathsf{id}}), \tilde{y}_d^{(0)}, \tilde{\mathsf{Q}}_d, \tilde{y}_d^{(1)}) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (\mathsf{id}, 0^{2\kappa - \log n}, t_{\mathsf{id}}), \tilde{y}_d^{(0)}, \tilde{\mathsf{Q}}_{d,\mathsf{sim}}, \tilde{y}_{d,\mathsf{sim}}^{(1)})$$

Hence, it follows that for each $i = 0, \cdots, d-1$, $\mathbf{Hybrid}_i \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{i+1}$.
Now, as the sequence of hybrids were defined for uniformly drawn message m, by the blindness of underlying T-RBE scheme, we know that:

$$(\mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}), \mathrm{TEnc}_2(\mathsf{pp}_T, t_{\mathsf{id}}, \mathsf{m}; \mathsf{r})) \overset{\mathsf{c}}{\approx} (\mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r}), U)$$

Hence, by the blindness property of the hash garbling scheme, it follows that:

$$(\mathsf{hk}, (\mathsf{id}, 0^{2\kappa - \log n}, t_{\mathsf{id}}), \tilde{y}_d^{(0)}, \tilde{\mathsf{Q}}_{d,\mathsf{sim}}, \tilde{y}_{d,\mathsf{sim}}^{(1)}) \overset{\mathsf{c}}{\approx} (\mathsf{hk}, (\mathsf{id}, 0^{2\kappa - \log n}, t_{\mathsf{id}}), \tilde{y}_d^{(0)}, U_{|\tilde{\mathsf{Q}}_d|+|\tilde{y}_1^{(d)}|})$$

Hence, $\mathbf{Hybrid}_d \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{d+1}$.
Similarly, by consecutive application of the blindness property of the hash garbling scheme, it hodlds that for each $i = d+1, \cdots, 2d-1$, $\mathbf{Hybrid}_i \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{i+1}$.
Hence, we have that, for uniformly drawn message m,

$$\mathbf{Hybrid}_0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{2d} \tag{1}$$

We now prove that, in fact, indistinguishability in Equation 1 is stronger than the desired amomymity of the RBE scheme:
For $b = 0, 1$, let $\mathsf{ct}_0^{\mathsf{m}_b, \mathsf{id}_b} = (\mathsf{subct}_{0,1}^{\mathsf{m}_b, \mathsf{id}_b}, \mathsf{subct}_{0,2}^{\mathsf{m}_b, \mathsf{id}_b})$ and $\mathsf{ct}_{2d}^{\mathsf{m}_b, \mathsf{id}_b} = (\mathsf{subct}_{2d,1}^{\mathsf{m}_b, \mathsf{id}_b}, \mathsf{subct}_{2d,2}^{\mathsf{m}_b, \mathsf{id}_b})$ denote the ciphertexts in $\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_{2d}$ respectively for the b-th message, identity pair.

1. By security of T-RBE, it would follow that $\mathbf{Hybrid}_d^0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_d^1$, where $\mathbf{Hybrid}_i^\beta$ denote the hybrids described above with use of message m_β. Hence, it follows that $\mathbf{Hybrid}_0^0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_0^1$.

2. Using the above indistinguishability, it follows that, for a random message m, we have $\mathsf{ct}_0^{\mathsf{m}_0, \mathsf{id}_0} \overset{\mathsf{c}}{\approx} \mathsf{ct}_0^{\mathsf{m}, \mathsf{id}_0}$

3. Since $\mathbf{Hybrid}_0 \overset{\mathsf{c}}{\approx} \mathbf{Hybrid}_{2d}$ for random message m, it then follows that $\mathsf{ct}_0^{\mathsf{m}, \mathsf{id}_0} = (\mathsf{subct}_{0,1}^{\mathsf{m}, \mathsf{id}_0}, \mathsf{subct}_{0,2}^{\mathsf{m}, \mathsf{id}_0}) \overset{\mathsf{c}}{\approx} (\mathsf{subct}_{0,1}^{\mathsf{m}, \mathsf{id}_0}, U) = \mathsf{ct}_{2d}^{\mathsf{m}, \mathsf{id}_0}$

4. Since $\mathsf{subct}_{0,1}^{\mathsf{m}, \mathsf{id}_0} = \mathrm{TEnc}_1(\mathsf{pp}_T; \mathsf{r})$, is independent of identity or the message m, it then follows that $\mathsf{ct}_{2d}^{\mathsf{m}, \mathsf{id}_0} \overset{\mathsf{c}}{\approx} \mathsf{ct}_{2d}^{\mathsf{m}, \mathsf{id}_1}$.

Combining the steps 2, 3 and 4 above, it follows that $\mathsf{ct}_0^{\mathsf{m}_0, \mathsf{id}_0} \overset{\mathsf{c}}{\approx} \mathsf{ct}_0^{\mathsf{m}_1, \mathsf{id}_1}$. This exactly represents the stronger security game of the anonymous RBE. Hence, the anonymity and security of the RBE scheme is proved. □

References

[AKTZ17] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *USENIX Security Symposium. USENIX Association, Vancouver, BC*, pages 1217–1234, 2017.

[ARP03] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer, 2003.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.

[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, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM, 2004.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BSJ+17] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *Annual International Cryptology Conference*, pages 619–650. Springer, 2017.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Annual International Cryptology Conference*, pages 290–307. Springer, 2006.

[CB95] David A Cooper and Kenneth P Birman. Preserving privacy in a network of mobile computers. Technical report, Cornell University, 1995.

[CCV04] Zhaohui Cheng, Richard Comley, and Luminita Vasiu. Remove key escrow from the identity-based encryption system. In *Exploring New Frontiers of Theoretical Informatics*, pages 37–50. Springer, 2004.

[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, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CGBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. *arXiv preprint arXiv:1503.06115*, 2015.

[CGCD+17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, pages 451–466. IEEE, 2017.

[CGF10] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 340–350. ACM, 2010.

[Cho09] Sherman SM Chow. Removing escrow from identity-based encryption. In *International Workshop on Public Key Cryptography*, pages 256–276. Springer, 2009.

[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, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. Theory of Cryptography Conference (TCC), 2018. https://eprint.iacr.org/2018/919.

[GLSW08] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436. ACM, 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, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery.

[JS18] Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In *Annual International Cryptology Conference*, pages 33–62. Springer, 2018.

[Moh10] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 501–518, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[PR18] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2018.

[RMS18] Paul Rösler, Christian Mainka, and Jörg Schwenk. More is less: On the end-to-end security of group chats in signal, whatsapp, and threema. 2018.

[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, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[UDB+15] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. Sok: secure messaging. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 232–249. IEEE, 2015.

[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*, pages 1–10, 2018.