Lower Bounds for the Number of Decryption Updates in Registration-Based Encryption

Mohammad Mahmoody[1⋆], Wei Qi[1⋆⋆], and Ahmadreza Rahimi[2⋆⋆⋆]

[1] University of Virginia, Charlottesville, VA, USA
[2] Max Planck Institute for Security and Privacy, Bochum, Germany

Abstract. Registration-based encryption (Garg, Hajiabadi, Mahmoody, Rahimi, TCC'18) aims to offer what identity-based encryption offers without the key-escrow problem, which refers to the ability of the private-key generator to obtain parties' decryption keys at wish. In RBE, parties generate their own secret and public keys and register their public keys to the *key curator* (KC) who updates a *compact* public parameter after each registration. The updated public parameter can then be used to securely encrypt messages to registered identities.

A major drawback of RBE, compared with IBE, is that in order to decrypt, parties might need to periodically request so-called *decryption updates* from the KC. Current RBE schemes require $\Omega(\log n)$ number of updates after n registrations, while the public parameter is of length $\mathrm{poly}(\log n)$. Clearly, it would be highly desirable to have RBEs with only, say, a constant number of updates. This leads to the following natural question: *are so many (logarithmic) updates necessary for RBE schemes, or can we decrease the frequency of updates significantly?*

In this paper, we prove an almost tight lower bound for the number of updates in RBE schemes, as long as the times that parties receive updates only depend on the registration time of the parties, which is a natural property that holds for all known RBE constructions. More generally, we prove a trade-off between the number of updates in RBEs and the length of the public parameter for any scheme with fixed update times. Indeed, we prove that for any such RBE scheme, if there are $n \geq \binom{k+d}{d+1}$ identities that receive at most d updates, the public parameter needs to be of length $\Omega(k)$. As a corollary, we find that RBE systems with fixed update times and public parameters of length $\mathrm{poly}(\log n)$, require $\Omega(\log n/\log\log n)$ decryption updates, which is optimal up to a $O(\log\log n)$ factor.

⋆ mohammad@virginia.edu Supported by NSF grants CCF-1910681 and CNS1936799.
⋆⋆ wq4sr@virginia.edu Supported by NSF grants CNS1936799.
⋆⋆⋆ ahmadreza.rahimi@mpi-sp.org

Table of Contents

1 Introduction

Identity-based encryption (IBE) [Sha84,BF01] is a powerful encryption primitive that allows a large group of identities to have a single public parameter pp in such a way that encryption to any identity id is possible solely based on the public parameter and id. The main weakness of IBE is the so-called key-escrow problem [Rog15,BF01,ARP03]. In particular, IBE schemes need a master secret key msk that is needed to generate personalized *decryption keys* $\mathsf{dk}_{\mathsf{id}}$ for each identity id, so that id can decrypt messages that are encrypted for them. This means the holder of msk, called the "private-key generator" (PKG) can decrypt all the messages, even the ones that are encrypted to parties who have not even requested their decryption keys yet!

To address the key escrow problem with IBE, Garg et al. [GHMR18] introduced a new primitive called Registration-based encryption (RBE). RBE is indeed a hybrid of IBE and the more basic primitive of public-key encryption. In RBE, every identity generates their own pair of public and secret keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$. Then, if a party id decides to "register" (i.e. join the system), they can send a request to a central party who manages the keys and is called the *key curator* (KC). KC runs a deterministic and fully transparent algorithm and updates two pieces of information: an auxiliary information aux_n as well as a *compact* public

parameter pp_n, where n shows how many people have registered in the system so far. The public parameter pp_n could be used like a public parameter of IBE to encrypt messages to any of the n identities who have registered so far. The auxiliary information aux_n will be used to facilitate the next registration (and another operation called update, which is discussed below). A key advantage of RBE over IBE is that parties own their secret keys. However, they might sometimes need extra help from the KC to decrypt ciphertexts that are encrypted to them, but perhaps using public parameters that are generated after the recipient identity is registered in the system. However, these "decryption updates" shall be needed rarely to make RBE useful.

Number of updates vs. compactness of public-parameters. If one puts no bound on the length of the public parameter pp_n, then a simple concatenation of all the public keys of the registered parties $\mathsf{pp}_n = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ can be used to trivially achieve RBE. Here the parties simply pick the public key of the receiver to encrypt their messages to them. This trivial scheme does not need any decryption updates! Hence, RBE is only meaningfully useful, if $|\mathsf{pp}_n| = o(n)$ grows sublinearly. In [GHMR18], it was suggested to keep $|\mathsf{pp}_n| = \mathrm{poly}(\kappa, \log n)$ as the default level of compactness for the public parameter and keep the number of needed decryption updates to be $O(\log n)$. The work of [GHMR18] also constructed such schemes based on indistinguishability obfuscation [BGI+01,GGH+13,JLS21] and somewhere-statistically binding hashing schemes [HW15].

At a very high level, the public parameter pp_n in [GHMR18] is the root of a Merkle tree that hashes all the public parameters of the registered identities, and so it can also be viewed as a commitment to all those public keys. This makes the job of the KC very similar to that of accumulators [BdM94,BP97,CL02]. The ciphertexts in [GHMR18] are obfuscations of programs that anticipate an "opening" into the identity's public key and output encryption of the message under such public keys. Therefore, to decrypt a message an identity id_i would need to know the "decommitment" (opening) to its public key pk_i with respect to the commitment message pp_n. When the identities register, the Merkle tree grows and decommitment needs to be updated as well. Therefore, when more parties register, the previously registered parties need to request decryption updates to keep their decommitments up to date. This approach led to $\Theta(\log n)$ number of updates. The work of Garg et al. [GHM+19] further improved the assumptions needed for constructing RBE to more standard ones (such as CDH or LWE) by, roughly speaking, substituting the obfuscation part with the powerful garbling techniques of [DG17]. Furthermore, Goyal and Vusirikala [GV20] added efficient verifiability mechanisms for membership and non-membership of identities.

All the RBE schemes so far have the same asymptotic efficiency barriers built into them: they all use the same level of $\mathrm{poly}(\kappa, \log n)$ compactness for the public parameter and require $\Theta(\log n)$ number of updates to guarantee successful decryption. In this work, we revisit these bounds and ask the following question.

How many decryption updates are needed in RBEs with public parameters of length $\mathrm{poly}(\kappa, \log n)$*? More generally, what is the trade-off between the number of updates and the length of the public parameter?*

Our main result provides an answer to the question above by proving an almost tight lower bound for the number of updates of any RBE schemes in which the update times are fixed. We say an RBE scheme has fixed update times if for every $i \le j$, it is known ahead of any actual registrations whether or not id_i (i.e., the ith registered identity) needs a decryption update after the registration of id_j.[3] Interestingly, all known constructions of RBE [GHMR18,GHM+19,GV20] have fixed update times, and it is indeed unclear whether the times for the updates can be tied to the public keys and (and the CRS) in a meaningful way.[4] See Remark 1.3 for more discussions on how fixed updates arise in the current constructions naturally, and why they are a useful property to have on their own. More generally, we prove a *trade-off* between the number of updates that are needed and the size of the public parameter.

Theorem 1.1 (Main result). *Let Π be any RBE scheme in which only d decryption updates are needed for each identity when we limit the scheme to only n identities. Further, suppose the times of the updates are only a function of the time when a party registers and the total number of parties so far, and that the length of the public parameter $|\mathsf{pp}_n|$ is non-decreasing in n.*[5] *Then,*

$$\binom{|\mathsf{pp}_n| + d}{d+1} \ge n.$$

In particular, for constant number of updates $d = O(1)$, one needs public parameters of length $|\mathsf{pp}_n| \ge \Omega(n^{1/(d+1)})$, and for public parameters of length at most $|\mathsf{pp}_n| \le \mathrm{poly}(\log n)$ one needs at least $d \ge \Omega(\log n / \mathrm{loglog}\, n)$ many updates.

See Theorem 4.1 and Corollary 4.2 for more details.

Our result leaves it open to either extend our lower bound to RBE schemes with *dynamic* update times that depend on the public keys or to invent new RBE schemes with dynamic update times that bypass our lower bound. In addition, it remains open to close the rather small gap of $1/\mathrm{loglog}\, n$ factor between our lower bound and the upper bounds of previously constructed RBE schemes.

1.1 Technical overview

We prove Theorem 1.1 by giving an explicit polynomial-time attack on any RBE scheme that does not satisfy the stated trade-off between the public parameters'

[3] More formally, there is an "update graph" G that is fixed and tells us if id_i needs an update after id_j registers or not.

[4] By "meaningful", here we mean that the novel scheme cannot be trivially turned into one with fixed update patterns, as it is not hard to come up with contrived schemes whose update times depend on the public keys.

[5] Notice that, one can always make $|\mathsf{pp}_n|$ non-decreasing using simple padding (with zeros) that prevents pp_n from shrinking when n grows.

length and the number of updates. Below, we fix n to be the number of the parties who register in the system. For simplicity, we work with registered identities $\{\mathsf{id}_1 = 1, \mathsf{id}_2 = 2, \ldots\}$ who register in this exact order.

Good identity tuples for the attack. At a very high level, we show that for any RBE scheme with n parties, d update at fixed times (independently of the keys), and compact public parameter $\binom{|\mathsf{pp}_n|+d}{d+1} < n$, there exists a tuple $(i, k) \in [n]^2, i \leq k$ that is "good for the attacker" in the following sense. If one encrypts a message m for $\mathsf{id}_i = i$ using pp_k (i.e., the public parameter right after id_k registers), then the adversary can successfully decrypt the ciphertext back into m, even though it does not have the real secret key of id_i. Note that we prove this, despite the fact that the public parameter pp_k could still be "linked" with the public and secret keys of id_i (through the algorithm used by the KC). Yet, we prove that if compared to the number of updates the public parameter is not long enough, there is always a tuple (i, k) that is good for the attacker to succeed.

Before proving the existence of good tuples and explaining how the adversary actually uses them in its attack, we first outline the ideas that we develop to achieve our goals. At a high level, we use two types of ideas as follows.

- *Information theoretic* ideas will rely on the length of the public parameter.
- *Combinatorial* tools will rely on the number of decryption updates.

In the following, we explain both of these ideas and how they play their role in our attack and its analysis. In order to do that, we first go over the simplest form of RBE schemes, in which *no updates* are allowed. This allows us to explain information theoretic ideas more clearly. We then extend the attack and its analysis to RBE schemes that allow *one* updates. Even this simple case will be instructive to show the challenges that arise and the new (combinatorial) tools that become necessary to overcome these challenges. The full poofs for the general case can be found in Section 4. For simplicity of the presentation, we ignore the existence of a CRS, but our proofs extend to having CRS as well.

Breaking RBEs with no updates: information theoretic tools. Suppose an RBE scheme has *no updates*. This is not entirely impossible, as one can always concatenate the public keys and store them as one giant public parameter that grows linearly with the number of parties n. But is this linear dependence on n necessary? Here we observe, using basic information theoretic tools, that this is indeed the case. First, we define a notation for keys as random variables.

Notation. We use $\mathsf{KEY}_i = (\mathsf{PK}_i, \mathsf{SK}_i)$ to denote the public/secret keys of id_i, as random variables. We also use PP_k to denote pp_k as a random variable.

Bounding the mutual information. If $|\mathsf{pp}_n| \leq \ell$, then the (Shannon) entropy of PP_n can be at most ℓ bits.[6] Therefore, the mutual information $I(\mathsf{KEY}_{1,\ldots,n}; \mathsf{PP}_n)$

[6] See Definition 2.5 for the definition of entropy.

between PP_n and concatenation of all the keys $\mathsf{KEY}_{1,\ldots,n} = (\mathsf{KEY}_1, \ldots, \mathsf{KEY}_n)$ is also bounded by ℓ.[7] Since the keys are generated *independently* for different identities, the average mutual information between PP_n and KEY_i of a random party i is bounded $\mathbb{E}_{i \leftarrow [n]}[I(\mathsf{KEY}_i; \mathsf{PP}_n)] \leq \ell/n$. Therefore, there exists $i \in [n]$, such that $I(\mathsf{KEY}_i; \mathsf{PP}_n) \leq \ell/n$. Such pair $(i, k = n)$ will be *good* for the attacker.

From bounded mutual information to independence. If $I(\mathsf{KEY}_i; \mathsf{PP}_n) \leq \ell/n = \varepsilon$ is sufficiently small (e.g., due to the small length of the public parameter), we can use the Pinsker's inequality (see Lemma B.5) to conclude that the two distributions below are $O(\sqrt{\varepsilon})$-statistically close

$$(\mathsf{KEY}_i, \mathsf{PP}_n) \approx_{O(\sqrt{\varepsilon})} (\mathsf{KEY}_i \otimes \mathsf{PP}_n),$$

where in the left side $(\mathsf{KEY}_i, \mathsf{PP}_n)$ is the *jointly* sampled pair of PP_n and keys KEY_i for id_i, while in the right side the $\otimes$ notation indicates that KEY_i and PP_n are sampled from their corresponding true marginal distribution, but they are sampled *independently* of each other.

From independence to successful attacks. The argument above shows that due to the (almost) independence of the keys KEY_i of id_i and PP_n, if the adversary simply picks a fresh pair of fake keys $(\mathsf{PK}'_i, \mathsf{SK}'_i) = \mathsf{KEY}'_i$ for id_i and uses SK'_i to decrypt the messages encrypted for id_i, it will succeed with probability $\rho - O(\sqrt{\varepsilon})$, where ρ is the completeness of the scheme. The reason is that using the correct keys would succeed with probability ρ, and switching to fake keys will affect this probability by at most $O(\sqrt{\varepsilon})$.

The attack above on the simple RBE schemes with no updates crucially uses the fact that no decryption updates are received by the parties at any time during the course of the system. In fact, the information theoretic argument above completely breaks down even if the registered parties receive just *one* update right after they register! To see why suppose u_i is the single decryption update received by id_i at some point after they register. Then, decrypting messages that are encrypted to id_i might require both sk_i and the update u_i to succeed. Therefore, we cannot simply rely on $(\mathsf{KEY}_i, \mathsf{PP}_n) \approx_{O(\sqrt{\varepsilon})} (\mathsf{KEY}_i \otimes \mathsf{PP}_n)$, and e.g., stronger conditions that also involve u_i might be necessary.

Breaking RBEs with single immediate updates. For the simpler case that the update u_i is generated *right* after the registration of id_i, we can still use the ideas for the no-update setting and slightly more powerful information theoretic tools. First, note that the adversary needs to generate some form of (fake) u'_i to run the decryption. A natural way to do it is to generate this fake update u'_i *using the fake keys* KEY'_i that it has generated for the vulnerable party id_i (where (i, n) is a good pair as explained above). A key point is that this update u'_i cannot be generated using KEY'_i alone, and it also needs to use as input the publicly available *auxiliary information* that is stored at the key curator. This

[7] See Definition B.1 for the definition of mutual information.

public information is a function of (the CRS and) the registered public keys. Hence, u_i is a function of KEY'_i and the previously registered public keys.

The above subtle point shows that the approximate independence of KEY_i and PP_n is no longer sufficient for the attack's success, and we need to *also condition* on the previously registered (public keys). Fortunately, this is not a problem, as we can start from a stronger condition that still can be proven based on the length of the public parameter: $\mathbb{E}_{i\leftarrow[n]}[I(\mathsf{KEY}_i;\mathsf{PP}_n|\mathsf{KEY}_1,\ldots,\mathsf{KEY}_{i-1})] \leq \ell/n$. (Note that we are now conditioning on the previous keys). Therefore, there exists $i \in [n]$, such that $I(\mathsf{KEY}_i;\mathsf{PP}_n|\mathsf{KEY}_1,\ldots,\mathsf{KEY}_{i-1}) \leq \ell/n$. Therefore, we can again use a variant of Pinsker's inequality and show that $(\mathsf{KEY}_i,\mathsf{PP}_n) \approx_{O(\sqrt{\varepsilon})} (\mathsf{KEY}_i \otimes \mathsf{PP}_n)$, holds even conditioned on the previously registered keys. Such i will again make the pair $(i, k = n)$ a *good pair* for the attack.

Breaking RBEs with single updates arriving at arbitrary times. When updates can arrive at arbitrary times, the simple information theoretic arguments above break down, as we cannot simply use the fake keys of the party id_i to generate its needed decryption update. This means that we might need to go a few steps further in time and even fake the keys of the parties $\mathsf{id}_{i+1},\ldots$ to be able to generate a useful update. But this will increase the length of the random variables that we fake and that kills the small mutual information with the public parameter. At a high level, we will *group* the identities in such a way that different groups can be seen as "large identity" groups that can collectively generate the needed update for the first identity in that group.

More formally, to attack RBEs with single updates that can arrive at any moment after registration of id_i, we define the notion of a good *triple* $i \leq j \leq k$ (for the attack) such that when the triple (i, j, k) is good, according to our definition, then the pair (i, k) would be good (for the attacker) as described above; namely, id_i will become vulnerable to attacks after the kth registration. The number j with $i \leq j \leq k$ denotes *how* this attack will be done. In particular, we call (i, j, k) a good triple if it has both of the following two properties.

1. *Being useful in relation with updates.* We require that id_i will not receive *any* updates during the registrations of $\mathsf{id}_{j+1},\ldots,\mathsf{id}_k$. This means that, if we only use the updates generated for id_i till the registration of id_j, id_i can still decrypt messages that are encrypted till the registration of id_k.
2. *Being useful in relation with key independence.* We require that the concatenation of the keys of the identities $(\mathsf{KEY}_i,\mathsf{KEY}_{i+1},\ldots,\mathsf{KEY}_j)$, as one big random variable, is almost independent of PP_k, and this holds when we condition on the first $i-1$ pair of keys $(\mathsf{KEY}_1,\ldots,\mathsf{KEY}_{i-1})$.

If the above two conditions hold for a triple (i, j, k), then one can still use an almost identical attack to that of the simpler cases above on the target identity i as follows. The adversary simply asks a message to be encrypted to id_i using pp_k. Then, it *re-samples* fake keys $\{\mathsf{KEY}_i,\ldots,\mathsf{KEY}_j\}$ for *all* parties $\{\mathsf{id}_i,\ldots,\mathsf{id}_j\}$. It then registers all these fake keys in its head starting from the auxiliary information aux_{i-1} of the KC for the moment right before the registration of id_i. During

these fake registrations, the adversary looks for any potential (fake) decryption updates that might be generated for id_i. The adversary uses all of these fake updates and the fake secret key $\mathsf{sk}'_i \leftarrow \mathsf{SK}'_i$ for id_i and tries to decrypt the challenge ciphertext. Therefore, all we need to do is to prove good triples exist. Below, we sketch why good triples exist by relying on the fact that the public parameter is small enough compared to the number of updates. For this, we would need to introduce some useful graph theoretical notions.

DAGs of decryption update times. Let G be the following directed acyclic graph (DAG) on nodes $[n]$ in regard to an RBE scheme Π. Connect i to j, if id_i receives a decryption update right after id_j registers. Note that the number of updates d translates into an upper bound on the *out-degree* of the nodes in G. We refer to G as the *update graph* of the RBE scheme Π.

Skipping sequences in DAGs. We now identify a special type of sub-graphs of DAGs like G that can help an adversary break an RBE scheme whose graph of update times is G. Let G be a DAG modeling the update times of our RBE scheme as explained above over the vertices (identities) $[n]$. We call a sequence $\mathcal{S} = \{u_1 < u_2 < \cdots < u_\ell\} \subseteq [n]$ a *skipping sequence* if for every $t \leq \ell - 1$ and every edge $(u_t, v) \in G$ (denoting that the u_tth identity id_{u_t} gets an update when the vth identity id_v registers), it holds that $v \notin \{u_{t+1}, u_{t+1}+1, \ldots, u_\ell\}$. In other words, identity id_{u_t} will either get updates before time $\mathsf{id}_{u_{t+1}}$ registers, or after time id_{u_ℓ}, but not in between.[8] Intuitively, the sequence $\{u_1 < u_2 < \cdots < u_\ell\}$ allows us to group the identities into ℓ groups such that each group internally generate the update needed for their first member.

Skipping sequences imply good triples. Here we show that if a skipping sequence in the update graph G is long enough, it implies the existence of a good triple (i, j, k). To see why, let $\mathcal{S} = \{u_1 < u_2 < \cdots < u_\ell\} \subseteq [n]$ be a skipping sequence.

- For all $t \in [k]$, $(i = u_t, j = u_{t+1} - 1, k = u_\ell)$ satisfies the first property that a good triple needs. This is directly implied by the non-existence of update edges going from u_i to any of the vertices $u_{t+1}, u_{t+1} + 1, \ldots, u_\ell$, as guaranteed by the definition of skipping sequences.
- Partition the set of (pairs of) keys of the registered identities into bigger random variables as follows. Put the keys KEY_u of identity id_u in group $\mathcal{K}_t$ if $u_t \leq u < u_{t+1}$. This partitions the set of all keys of parties corresponding to the vertices $\{u_1, \ldots u_\ell\}$ into $\ell - 1$ groups. Using the chain rule for mutual information, we can again conclude that at least for *one* of these groups $\mathcal{K}_t$, it holds that the keys in the group $\mathcal{K}_t$ (jointly) have at most $|\mathsf{pp}_k|/\ell$ mutual information with PP_k, when we condition on the keys of all the parties who registered prior to id_{u_t}. Therefore, we can again apply Pinsker's inequality and prove that the triple $(i = u_t, j = u_{t+1} - 1, k = u_\ell)$ also satisfies the second property needed for a triple to be good.

[8] See Definition 3.1 for a formal definition.

DAGs of bounded out-degrees contain long skipping sequences. It remains to show that any DAG with a "small" out degree contains a "large" skipping sequence. Here we explain the proof for the simple case of out-degrees equal to 1. The idea, however, can be extended to arbitrary (bounded) out-degrees (see Theorem 3.2). We call a finite graph G a *forward* DAG if the vertices of G are $[n]$ and all the edges are of the form (i, j) for $i \leq j$. We use $\deg^+(u)$ to denote the out-degree of u, which is the number of nodes like v where (u, v) is an edge in G.

Claim 1.2 (Long skipping sequence in forward DAGs of out-degree at most 1*).* Let $G = (\mathcal{V}_G = [n], \mathcal{E}_G)$ be a forward DAG of size $n = \binom{k+1}{2}$ for $k \in \mathbb{N}$, and that $\deg^+(G) \leq 1$. Then, there exists a skipping sequence in G of size k.

In the following, we prove this claim. Since $n = \binom{k+1}{2} = \sum_{i=1}^{k} i$, we divide the n vertices into k *groups* $\{\mathcal{G}_i\}_{i\in[k]}$ such that when we read the vertices in the order $1, \ldots, n$, the members of the group $\mathcal{G}_i$ are immediately after the vertices of group $\mathcal{G}_{i-1}$ and the ith group $\mathcal{G}_i$ has $(k + 1 - i)$ vertices.

We say an edge (u, v) *lies in* a group $\mathcal{G}_i$ if both $u, v \in \mathcal{G}_i$. We say a group $\mathcal{G}_i$ is *green* if there exists at least one vertex in $\mathcal{G}_i$ whose out-going edge lies in $\mathcal{G}_i$ and we call any such vertex a *representative* of $\mathcal{G}_i$. Otherwise, group $\mathcal{G}_i$ is *red*.

Now we do as follows to find a skipping sequence of the size we want:

1. If all k groups are green, then we select exactly one representative r_i from each group $\mathcal{G}_i$ and construct sequence $\mathcal{S} = \{r_1 < r_2 < \cdots < r_k\}$. By construction, for $i < k$ the out-going edge (r_i, v) must lie in $\mathcal{G}_i$, which implies that $v < r_{i+1}$. Thus, $\mathcal{S}$ is a skipping sequence.
2. If red groups exist, let $\mathcal{G}_j$ be the red group with smallest j. We then construct the sequence $\mathcal{S} = \{r_1 < r_2 < \cdots < r_{j-1}\} \cup \mathcal{G}_j$ of size k.

Where did we rely on the fixed update times? In the proof sketched above, we partition the keys into groups based on the skipping sequence $\mathcal{S}$ that comes out of the update graph G. We then argued that, because this sequence is long enough, the mutual information between pp_k and one of the groups of keys *defined by* $\mathcal{S}$ is small. If we allow the graph G itself to be correlated with the public parameter pp_k, the graph itself can carry information. Alternatively, one might try to first *sample* and fix the graph G based on the execution of the system. After all, it will again be a low-out-degree graph and it *will* be guaranteed to have a long skipping sequence $\mathcal{S} = \{u_1 < \cdots < u_\ell\}$. However, if we pick a triple $(u_t, u_{t+1} - 1, u_\ell)$ as a candidate good triple, even an adversary who has the *real* keys for the identities corresponding to $\{u_t, \ldots, u_{t+1} - 1\}$ might fail to decrypt the challenge ciphertext! That is because when we change the keys, the update times might change and now identity u_t might need an update *after* time u_{t+1}.

Remark 1.3 (How to interpret the assumption of fixed update times). As mentioned before, all known constructions of RBE have fixed update times. Here we sketch the reason. Despite their differences, the RBE schemes so far consist of two components: a data structure that serves as a commitment/accumulator for identity-key pairs, and a "crypto" component that either employs IO or garbling

to achieve a form of "delayed encryption". The first component in known constructions of RBE always consists of "subsets" whose *sizes* determine the update times, while these sizes only depend on the number of identities registered so far. Moreover, fixed update times seems like a meaningful feature on its own to have for an RBE scheme, as it allows the parties in an RBE system to know when they will need updates solely based on their registration time and the total number of registered parties (without the need for failing in decryption to realize that their credentials are outdated). In fact, if the RBE system is designed in a (natural) way that KC itself takes on the role of pushing the updates then having fixed update times would be even more natural, as the KC actually does *not* have parties' secret keys to even try any decryption. As it remains open to potentially bypass our lower bound by leveraging on dynamic update times that depend on the registered keys, we point out a success story that might have some resemblance. Indeed, in the context of memory-hard functions, in which a DAG is also built on top of the input data, provable barriers against memory-hard functions have been overcome using data-*dependent* ones [BH22].

1.2 Related work

Here we review some further related work.

In addition to the works [GHMR18,GHM+19,GV20] that studied the feasibility and asymptotic efficiency of RBE, Cong, Eldefrawy, and Smart [CES21] studied the non-asymptotic practicality of implementing RBE schemes by estimating the concrete communication and computation costs of RBE and further optimizing it using alternative tools instead of Merkle trees.

Prior to RBE, other approaches have been pursued to address the key-escrow problem with IBE. One approach proposed by [BF01] was to make the PKG *decentralized* and run by multiple parties. Goyal [Goy07,GLSW08] proposed an after-the-fact approach of making PKG "accountable", by hoping to catch an irresponsible PKG in case of misuse. The works of [CCV04,Cho09,WQT18] aimed at (a related goal of) making it harder for the PKG to find out the receiver identity by hiding it in a large set of identities. Chow [Cho09] also studied ways to allow the users to interactively obtain secret keys without revealing their identities, and Emura et al. [EKW19] further formalized this approach.

The work of [ARP03] pursued another approach to mix IBE and public-key encryption by constructing "Certificateless" Public Key Cryptography. However, we shall clarify that, since the key-escrow is inherent to IBE, none of these approaches (including RBE) can really eliminate the key-escrow problem of IBE.

2 Definitions and preliminaries

2.1 Registration-based encryption

In this subsection, we first define the syntax of RBE. We then present new definitions of security and completeness for RBEs that are used in our lower

bounds. Standard definitions can be found in Section A. Our security notion is weaker than (and implied by) the standard RBE security definition; in our definition, the adversary does not get any secret keys. Using this definition makes a lower bound stronger. Our completeness is stronger than (and implies) the standard completeness definition of RBEs; in our definition, the update times are fixed. It remains open to extend our lower bounds to the standard completeness definition of RBE or to find a new construction that bypasses our lower bound.

Definition 2.1 (Syntax of registration-based encryption). *Five PPT algorithms* $(\mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ *form a* registration-based encryption *(RBE for short) if they work together as follows.*

- ***Generating CRS.*** *A common random string* crs *of length* $\mathrm{poly}(\kappa)$ *is publicly sampled at the beginning, for the security parameter* κ*.*
- ***Key Generation.*** $\mathsf{Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$*: The randomized algorithm* Gen *outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. The key generation algorithm is run by any honest party locally who wants to register itself into the system.*
- ***Registration.*** $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$*: The deterministic algorithm* Reg *takes as input the CRS* 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 *access to* auxiliary information aux *which will be updated into* aux' *during the process of registration and helps with the efficiency of the registration and updates (below). The system is initialized with* $\mathsf{pp}, \mathsf{aux} = \bot$*.*
- ***Encryption.*** $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$*: The randomized algorithm* Enc *takes as input the CRS* crs*, a public parameter* pp*, a recipient identity* id*, and a plaintext message* m*, and it outputs a ciphertext* ct*.*
- ***Update.*** $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{u}$*: The deterministic algorithm* Upd *takes as input the current public parameter* pp*, an identity* id*, and a public key* pk*. It has* read only *oracle access to* aux *and generates an update information* u *that can help* id *to decrypt its messages.*
- ***Decryption.*** $\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to \mathsf{m}$*: The deterministic decryption algorithm* Dec *takes as input a secret key* sk*, an update information* u*, and a ciphertext* ct*, and it outputs a message* $\mathsf{m} \in \{0,1\}^*$ *or in* $\{\bot, \mathsf{GetUpd}\}$*. The symbol* $\bot$ *indicates a syntax error while* GetUpd *indicates that more recent update information (than* u*) might be needed for decryption.*

The Reg *and* Upd *algorithms are performed by the party called* key curator*, which we call KC for short, and* aux *can be seen as the state held by the KC.*

See Definitions A.1 and A.2 for the standard definitions of completeness and security of RBEs. Below we present new definitions that are relevant to us.

We now introduce a generalization of the security of RBE called k-corruption security. In the original security definition of RBE (see Definition A.2), the adversary samples secret keys of all non-target identities. Here we only allow it to sample the keys of up to k non-target identities. In the extreme case where $k = n - 1$ (, meaning all but the target identity is corrupted,) the definition

matches Definition A.2. In the extreme case where $k = 0$, the adversary is essentially an observer who is curious to decrypt messages sent to parties.

Definition 2.2 (k-corruption security for RBE). *Let k be a positive integer. For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathsf{Sec}_{\mathcal{A}}^{k\text{-}\mathsf{c}}(\kappa)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. ***Initialization.*** *$\mathcal{C}$ sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, (the set of non-corrupted identities) $\mathcal{D}_{\mathsf{nc}} = \emptyset$, (the set of corrupted identities) $\mathcal{D}_{\mathsf{c}} = \emptyset$, $\mathsf{id}^* = \bot$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to $\mathcal{A}$.*
2. *Till $\mathcal{A}$ continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to perform.*
   (a) ***Registering a corrupted (non-target) identity.*** *This step is allowed only if $|\mathcal{D}_{\mathsf{c}}| < k$. $\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}_{\mathsf{nc}} \cup \mathcal{D}_{\mathsf{c}}$ and pk to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by letting $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathcal{D}_{\mathsf{c}} := \mathcal{D}_{\mathsf{c}} \cup \{\mathsf{id}\}$.*
   (b) ***Registering an uncorrupted (potentially target) identity.*** *$\mathcal{A}$ sends an $\mathsf{id} \notin \mathcal{D}_{\mathsf{nc}} \cup \mathcal{D}_{\mathsf{c}}$ to $\mathcal{C}$. $\mathcal{C}$ samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, runs $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$, $\mathcal{D}_{\mathsf{nc}} := \mathcal{D}_{\mathsf{nc}} \cup \{\mathsf{id}\}$, and sends pk to $\mathcal{A}$.*
3. ***Encrypting for a target identity.*** *$\mathcal{A}$ first sends some $\mathsf{id}^* \notin \mathcal{D}_{\mathsf{c}}$ to $\mathcal{C}$. (If $\mathsf{id}^* \in \mathcal{D}_{\mathsf{nc}}$, then the adversary is targeting one of the registered uncorrupted identities, otherwise it is targeting a non-registered identity). Next $\mathcal{A}$ sends messages m_0, m_1 of equal lengths $|m_0| = |m_1|$ to the adversary. Then, $\mathcal{C}$ generates $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_b)$, where $b \leftarrow \{0,1\}$ is a random bit, and sends ct to $\mathcal{A}$. The adversary $\mathcal{A}$ outputs a bit b' and wins if $b = b'$.*

We call the scheme k-corruption secure, if for all PPT $\mathcal{A}$, it holds that

$$\mathsf{P}[\mathcal{A} \textit{ wins } \mathsf{Sec}_{\mathcal{A}}^{k\text{-}\mathsf{c}}(\kappa)] < \frac{1}{2} + \mathrm{negl}(\kappa).$$

We now formally define RBE schemes with *fixed* update times. In such schemes, when a person registers at time i, they already know the indices $j > i$ such that they would need an update when the jth identity registers. More formally, we use the following game which is similar to the game of Definition A.1, with the difference that the updates are generated as soon as they are required by an "updates graph" (DAG) G.

Definition 2.3 (Forward DAGs). *Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ be a directed acyclic graph (DAG) with vertices $\mathcal{V}_G = [n]$ (in case of being finite) or $\mathcal{V}_G = \mathbb{N}$ (in case of being infinite). We write $(i,j) \in G$ if $(i,j) \in \mathcal{E}_G$ (i.e., there is an edge from i to j in G). We call G a forward DAG, if for all $(i,j) \in G$, we have $i \leq j$.*

The definition below captures the property that by making the updates according to the graph G, namely, by giving the update to person i whenever person j registers and $(i,j) \in G$, then there will be no need for further updates. The definition is written for the setting where an "adversary" targets a specific identity (and aims to make the updates *insufficient* for it). However, the definition implies that even if there is more than one identity with honestly generated keys, their updates would never be necessary outside what graph G instructs.

Definition 2.4 (Completeness of RBE with a fixed update times). *Let G be an infinite forward DAG. For an RBE scheme and any interactive* computationally unbounded *adversary $\mathcal{A}$ that still has a limited* $\mathrm{poly}(\kappa)$ *round complexity, consider the game* $\mathsf{UpdTimes}^{G}_{\mathcal{A}}(\kappa)$ *between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.*

1. ***Initialization.*** *$\mathcal{C}$ sets* $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{u} = \bot$, $\mathcal{D} = \emptyset$, $\mathcal{S} = \emptyset$, $t = 0$, *and* $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$, *and sends the sampled* crs *to $\mathcal{A}$.*
2. *Till $\mathcal{A}$ continues (which is at most* $\mathrm{poly}(\kappa)$ *steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to perform.*
   (a) ***Registering identities.*** *$\mathcal{A}$ performs exactly one out of Step 2(a)i and Step 2(a)ii below, but regardless of this choice, $\mathcal{C}$ will continue to send the updates as described next.*
      i. ***Registering a corrupted non-target identity.*** *$\mathcal{A}$ sends some* $\mathsf{id} \notin \mathcal{D}$ *and* pk *to $\mathcal{C}$. $\mathcal{C}$ registers* $(\mathsf{id}, \mathsf{pk})$ *by letting* $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and* $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.
      ii. ***Registering the target uncorrupted identity.*** *This step is allowed only if* $\mathsf{id}^* = \bot$. *In that case, $\mathcal{A}$ sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to $\mathcal{C}$. $\mathcal{C}$ then samples* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\kappa)$, *runs* $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, *and sends* pk^* *to $\mathcal{A}$.*

   Immediately updating the target identity, if required by G. *This step is allowed only if* $\mathsf{id}^* \neq \bot$ *(otherwise this step is skipped). Suppose* id^* *was the ith registered identity, and let the identity registered in either of Step 2(a)i Step or 2(a)ii be the jth identity.*[9] *If $(i, j) \in G$ (i.e., there is an edge from i to j), then we update the decryption information* $\mathsf{u} = \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ *for the target identity.*[10]
   (b) ***Encrypting for the target identity.*** *This step is allowed only if* $\mathsf{id}^* \neq \bot$. *In that case, $\mathcal{C}$ sets $t = t + 1$. $\mathcal{A}$ sends* $\mathsf{m}_t \in \{0,1\}^*$ *to $\mathcal{C}$ who then sets* $\mathsf{m}'_t \coloneqq \mathsf{m}_t$ *and sends back a corresponding ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ *to $\mathcal{A}$.*
   (c) ***Decryption for the target identity.*** *$\mathcal{A}$ sends $j \in [t]$ to $\mathcal{C}$ who lets* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$.

The adversary $\mathcal{A}$ wins above, if there is some $j \in [t]$ for which $\mathsf{m}'_j \neq \mathsf{m}_j$. *This particularly holds, e.g., if* $m'_j = \mathsf{GetUpd}$. *We say that G is an update graph for the RBE scheme, if* $\mathsf{P}[\mathcal{A}\ \textit{wins}] = \mathrm{negl}(\kappa)$. *In this case, we also say that the completeness holds with fixed update graph G.*

2.2 Information-theoretic notation and the twig lemma

Notation. We use capital letters to refer to random variables and usually use lowercase letters of the same type to refer to samples from those random variables. $x \leftarrow X$ refers to sampling x from the random variable X. For jointly distributed random variables X, Y, by XY or (X, Y) we refer to their joint samples, and by $X \otimes Y$, we refer to sampling X, Y from their marginals independently. For

[9] Note that this registered identity itself could be id^*.
[10] This update might not be really necessary, but we still run them as instructed.

jointly distributed XY and $y \in Y$, by $X|_y$ we refer to the random variable X conditioned on the sampled y. By $X|_Y$ we emphasize that X is sampled jointly with (and conditioned on) Y, even though $X|Y$ only refers to a sample from X. Using this notation $(X|_Y)Y$ means the same thing as XY. For distributed X, Y, Z, by $(X|_Z \otimes Y|_Z)Z$ we refer to sampling $z \leftarrow Z$ first, and then sampling $X|_z, Y|_z$ independently from their marginals. By $X \equiv Y$ we mean that X, Y are identically distributed. We write $\approx_\varepsilon$ to denote ε-closeness in statistical distance. By $\mathrm{Supp}(X)$ we mean the support set of the random variable (or probability distribution) X. When we use X as set, we refer to its support set. So, $X \cup Y$ means $\mathrm{Supp}(X) \cup \mathrm{Supp}(Y)$. We let $\mathsf{P}_X[x] = \mathsf{P}[X = x]$. log means logarithm in base 2 and ln means logarithm in base e.

Definition 2.5 (Shannon entropy). *The Shannon Entropy of a random variable X is defined as $H(X) = \sum_{x \in X} -\mathsf{P}_X[x] \log \mathsf{P}_X[x]$. The conditional Shannon entropy $H(X|Y)$ is defined as $\mathbb{E}_{y \leftarrow Y}[H(X|_y)]$. The entropy chain rule states that $H(XY) = H(X) + H(Y|X) = H(Y) + H(X|Y)$.*

Definition 2.6 (Statistical distance). *Let X and Y be two random variables. We define the statistical distance between these two distributions as:*

$$\mathrm{SD}(X, Y) = \sum_{z \in X \cup Y} \frac{|\mathsf{P}_X[z] - \mathsf{P}_Y[z]|}{2}$$

We prove the following lemma in Section B.

Lemma 2.7 (The twig lemma). *Let $X_0, \ldots, X_\ell, Y$ be jointly distributed random variables. Then,*

$$\mathop{\mathbb{E}}_{i \leftarrow [\ell]}[\mathrm{SD}(X_0 \ldots X_i Y, X_0 \ldots X_{i-1} X_i' Y)] \le \sqrt{\frac{H(Y) \ln 2}{2\ell}}$$

in which $Y X_i \ldots X_0$ are sampled jointly, while $Y X_i' X_{i-1}, \ldots, X_0$ is sampled by first sampling $x_0 \ldots, x_{i-1} y \leftarrow X_0 \ldots X_{i-1} Y$ and then sampling X_i' from $X_i|_{x_1 \ldots x_{i-1}}$ by ignoring the sampled y.[11] *In particular, if the length of the samples from Y are at most d, there exists $i \in [\ell]$ such that*

$$\mathrm{SD}(X_0 \ldots X_i Y, X_0 \ldots X_{i-1} X_i' Y) \le \sqrt{\frac{d \ln 2}{2\ell}}.$$

3 Skipping sequences in DAGs

In this section, we formally study skipping sequences in DAGs and prove that they emerge when the out degrees are small. In Section C, we prove that the bounds of this section are tight. This means that our approach of using skipping sequences cannot improve our lower bound of $\log n / \mathrm{loglog}\, n$ updates in RBEs. This leaves open to close the gap between our lower bound and the upper bound of $\log n$ updates for future work.

[11] Using our notation, that means $Y X_i' X_{i-1}, \ldots, X_0 \equiv (Y|_Z \otimes X_i|_Z)Z$ for $Z = X_{i-1}, \ldots, X_0$.

Intuition for skipping sequence. Intuitively, we want to find a sequence of identities whose updates are relatively independent from the next identity in the sequence. Namely, every identity (except the last one) should receive all her updates before the next identity in the sequence joins. Looking forward, we will attack an RBE scheme immediately after the last identity of the sequence joins, so it is irrelevant whether the identities will potentially receive another update after the last one joins. This intuition is formalized by the following definition.

Definition 3.1 (Skipping sequence). *Let G be a forward DAG (see Definition 2.3). We call $\mathcal{S} = \{u_1 < u_2 < \cdots < u_k\} \subseteq \mathcal{V}_G$ a* skipping sequence *if for every $i \leq k-1$ and every edge $(u_i, v) \in G$, it holds that: either $v < u_{i+1}$ or $v > u_k$ (i.e., $v \notin \{u_{i+1}, u_{i+1}+1, \ldots, u_k\}$).*

See Figure 1 for examples.

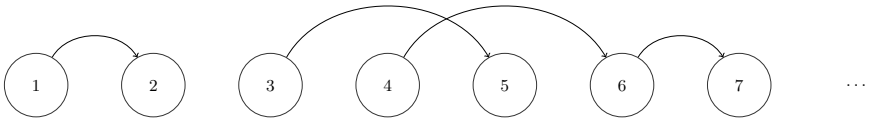

Fig. 1. Example of a forward DAG G with $\deg^+(G) = 1$. $\{1, 3, 6\}$ and $\{1, 3, 4\}$ are skipping sequences. $\{1, 3, 4, 6\}$ is not a skipping sequence, because vertex 3 has an outgoing edge to vertex 5 which is smaller than vertex 6 but larger than vertex 4.

We let $\deg^+(u) = |\{v \mid (u, v) \in G\}|$ be the *out-degree* of u and $\deg^+(G) = \max\{\deg^+(u) \mid u \in [n]\}$ to be the maximum out-degree in G.

Our main result in this subsection is the following theorem.

Theorem 3.2 (Skipping sequences from bounded out-degrees). *Let G be a forward DAG with at least $\binom{k+d}{d+1}$ vertices (for $k, d \in \mathbb{N}$) and that $\deg^+(G) \leq d$. Then, there exists a skipping sequence in G of size at least k.*

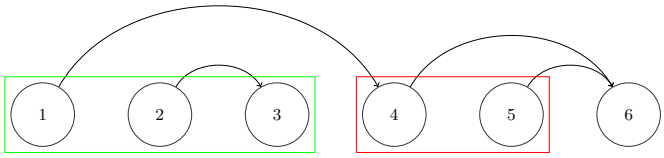

Fig. 2. Above is a forward DAG $G = ([6], \mathcal{E}_G)$ with out-degree $\deg^+(G) = 1$. Group $\mathcal{G}_1$ consists of vertices {1, 2, 3} and it is a green group because the only out-going edge from vertex 2 is (2, 3) and vertex 3 also belongs to $\mathcal{G}_1$. Namely, edge (2, 3) lies in $\mathcal{G}_1$. Vertex 2 is a representative of $\mathcal{G}_1$. Group $\mathcal{G}_2$ consists of vertices {4, 5} and it is a red group because no edge lies in it.

In Theorem C.1 we prove that the bound of Theorem 3.2 is tight.

Proof (of Theorem 3.2). We prove Theorem 3.2 by induction. Without loss of generality, we assume that G has exactly $n = \binom{k+d}{d+1}$ vertices. We use induction on the out-degree d. We first prove the base case for out-degree $d = 0$. In that case, all the k vertices of a forward DAG G with out-degree $\deg^+(G) = 0$ form a skipping sequence since none of the vertices has an out-going edge.

Assume the claim is true for forward DAGs of out-degree $d-1$. We now prove that the claim is true for forward DAGs of out-degree d. To this end, consider an arbitrary forward DAG $G = (\mathcal{V}_G, \mathcal{E}_G)$ where $|\mathcal{V}_G| = n = \binom{k+d}{d+1}$ and $\deg^+(G) \leq d$. From the hockey-stick identity we know

$$n = \binom{k+d}{d+1} = \sum_{i=1}^{k} \binom{i+d-1}{d}.$$

We divide the n vertices into k groups $\{\mathcal{G}_i\}_{i\in[k]}$ such that when we read the vertices $1, \ldots, n$, the members of the group $\mathcal{G}_i$ are exactly those after the vertices of group $\mathcal{G}_{i-1}$ and the i-th group $\mathcal{G}_i$ has $\binom{k+d-i}{d}$ vertices.

We say an edge (u, v) *lies in* a group $\mathcal{G}_i$ if both $u, v \in \mathcal{G}_i$. We say a group $\mathcal{G}_i$ is *green* if there exists at least one vertex in $\mathcal{G}_i$ all of whose outgoing edges lie in $\mathcal{G}_i$ and we call any such vertex a *representative* of $\mathcal{G}_i$. Otherwise, group $\mathcal{G}_i$ is *red*.

1. If all k groups are green, then we select exactly one representative r_i from each group $\mathcal{G}_i$ for $i \in [k]$ and construct a sequence $\mathcal{S} = \{r_1 < r_2 < \cdots < r_k\}$. In that case, we know that for $i \in [k-1]$ all out-going edges of r_i lie in $\mathcal{G}_i$. Therefore, $\mathcal{S}$ is a skipping sequence.
2. If there is at least one red group, let $\mathcal{G}_j$ be the red group with the smallest j. By the induction hypothesis, there is a skipping sequence $\mathcal{S}'$ of size $k+1-j$ in group $\mathcal{G}_j$ since it contains $\binom{k+d-j}{d}$ vertices all of which have at most $d-1$ outgoing edges that lie in the group. We then construct our desired skipping sequence as $\mathcal{S} = \{r_1 < r_2 < \cdots < r_{j-1}\} \cup \mathcal{S}'$ of size k.

□

4 Breaking RBEs with few updates

In this section, we present a trade-off between the length of the public parameter of RBEs with fixed update times and the number of such updates. At a high level, we design an attack against any RBE scheme with fixed update times assuming that the number of updates is "sufficiently small" compared to the public parameter. The key technical result of this section is Theorem 4.1 below. After stating Theorem 4.1, we first derive some corollaries about the number of updates. We then define a notion of "good tuple" (i, j, k) that can serve as a *useful advice* for breaking an RBE scheme. We then show how to break RBE schemes given a good tuple. Then, we prove Theorem 4.1 by showing that good tuples exist and how to find successful attacks without being given a good tuple.

Theorem 4.1 (Main result). *Let Π be an RBE scheme with a fixed update graph G (see Definition 2.4) and at most d updates for n registered identities; namely, $\deg^+(G) \le d$ when we limit the graph G to the first n nodes/identities. Suppose the scheme has completeness probability ρ. If $n \ge \binom{\ell+d}{d+1}$ and $|\mathsf{pp}_i| \le \alpha$ for all the first n registrations, then there is a 0-corruption (see Definition 2.2) $\mathrm{poly}(\kappa)$-time adversary who breaks Π with probability $\rho - \sqrt{\alpha \ln 2/(2\ell)} - \delta$ (i.e., advantage $\rho - 1/2 - \sqrt{\alpha \ln 2/(2\ell)} - \delta$) for arbitrarily small $\delta = 1/\mathrm{poly}(\kappa)$.*

Before proving Theorem 4.1, we derive a corollary for the extreme cases of constant number of (e.g., one) updates, and poly-logarithmic public parameters.

Corollary 4.2. *Let Π be an RBE scheme with a fixed update graph G and secure against 0-corruption. Let $\alpha_n = \max_{i\in[n]} |\mathsf{pp}_i|$ be the maximum length of the public parameter when n identities register.*

1. *If $\deg^+(G) \le d$ for a constant d, then $\alpha_n \ge \Omega(n^{1/(d+1)})$.*
2. *If $\deg^+(G) \le c \log n/\operatorname{loglog} n$ for a constant c, then $|\alpha_n| \ge \Omega(\log^{1/c} n)$.*
3. *If $|\alpha_n| \le \mathrm{poly}(\kappa, \log n)$ for security parameter n, then $\deg^+(G)$ cannot be $o(\log n/\operatorname{loglog} n)$. (I.e., there will be a constant c and an infinite sequence of n for which $\deg^+(G) \ge c \cdot \log n/\operatorname{loglog} n$.)*

Proof (of Corollary 4.2 using Theorem 4.1). First observe that by Theorem 4.1, if the scheme is complete $\rho > 0.99$, it holds that when $n = \binom{\ell+d}{d+1}$, then $\alpha_n \ge \ell/10$, as otherwise the scheme will not be secure.

1. If $\deg^+(G) \le d = O(1)$: In this case, since we have $n = \binom{\ell+d}{d+1} = \Theta(\ell^{d+1})$, therefore we get $\alpha_n \ge \Omega(\ell) \ge \Omega(n^{1/(d+1)})$.
2. If $\deg^+(G) \le d = c \log n/\operatorname{loglog} n$: Using the well-known upper bound on the binomial coefficients we get $n \le \left(\frac{(\ell+d)e}{d+1}\right)^{d+1}$. Taking logarithm, this implies

$$\log n \le (c \cdot (\log n/\operatorname{loglog} n) + 1) \cdot (\log(e\ell + ed) - \log(d+1)).$$

   It can be observed already that $\ell = \Omega(\log n)$ (for constant c), as otherwise, the right hand side will be $o(\log n)$. Therefore, it holds that $d = o(\ell)$. Therefore, we can simplify the above to the following for sufficiently large n

$$\log n \le \left(c \cdot \frac{\log n}{\operatorname{loglog} n} + 1\right) \cdot \log((e + o(1))\ell) < \left(c \cdot \frac{\log n}{\operatorname{loglog} n} + 1\right) \cdot \log(3\ell),$$

   which implies that $\operatorname{loglog} n \le c \log(3\ell)$, and so $\alpha_n \ge \Omega(\ell) \ge \Omega(\log^{1/c} n)$.
3. We use the previous item. Suppose $\alpha_n = O(\log^s n)$ for (fixed κ and) constant s, while $\deg^+(G) = o(\log n/\operatorname{loglog} n)$. Then, pick any (sufficiently small) constant c such that $1/c > s$. In this case, we still have $\deg^+(G) \le c \cdot \log n/\operatorname{loglog} n$, and $\alpha_n \ge \Omega(\log^{1/c} n)$, contradicting $\alpha_n = O(\log^s n)$.

□

Proof of Theorem 4.1. In the rest of this section, we prove Theorem 4.1 in three steps. We first define a notion of *good* tuple (i, j, k) and prove that good tuples exist. We then show that the attack can be launched successfully *if* a good tuple is given as advice. These two steps already show the existence of an efficient *non-uniform* attack. Finally, we show how to make the attack uniform by, roughly speaking, finding a good-enough advice efficiently.

4.1 Defining good tuples and proving their existence

We now define the notion of good tuples and prove that they actually exist under certain conditions. These tuples later are shown to be useful, as advice to an adversary, to break RBE schemes.

Definition 4.3 (Good tuples for forward DAGs). *Let G be a forward DAG with vertices $[n]$. For $i, j, k \in [n]$ where $i \leq j \leq k$, we say the tuple (i, j, k) is a* good tuple for G*, if the following holds:*

$$\nexists j' \in V \text{ such that } j < j' \leq k \wedge (i, j') \in G.$$

In other words, the ith registered identity will not need any updates starting from the $(j+1)$th registration till right after the kth registration.

Notation. We use the notation defined in Section 1.1 and use sans serif font for the random variables denoting the keys and public parameters. Using the same style, we also use CRS to refer to the CRS as a random variable.

Definition 4.4 (Good tuples for RBE schemes). *Let Π be any RBE scheme, and fix the first n identities to be $\{1, \ldots, n\}$. For $i, j, k \in [n]$ where $i \leq j \leq k$ we say the tuple (i, j, k) is $(1-\varepsilon)$-good for Π if the following two distributions are ε-close in statistical distance:*

$$(\mathsf{CRS}, \mathsf{KEY}_1 \ldots \mathsf{KEY}_j, \mathsf{PP}_k), \text{ and}$$

$$(\mathsf{CRS}, \mathsf{KEY}_1 \ldots \mathsf{KEY}_{i-1}, \mathsf{KEY}'_i \ldots \mathsf{KEY}'_j, \mathsf{PP}_k)$$

where in the first distribution all components are sampled jointly from an honest execution of the registration experiment in which the parties $[n]$ are registered in that order, while in the second distribution $\mathsf{KEY}'_i \ldots \mathsf{KEY}'_j$ are sampled independently of the other components.[12]

Definition 4.5 (Good tuples). *For an RBE scheme Π with fixed updates graph G and $\varepsilon < 1$, we simply call (i, j, k) a $(1-\varepsilon)$-good tuple, if it is both good for G* and *$(1-\varepsilon)$-good for Π.*

[12] Alternatively, one can pretend that there has been *true* values of $\mathsf{KEY}_i \ldots \mathsf{KEY}_j$ that were sampled jointly with $\mathsf{CRS}, \mathsf{KEY}_1 \ldots \mathsf{KEY}_{i-1}, \mathsf{PP}_k$ and were thrown out to be replaced with fresh samples at the end.

In the next subsection, we will show that *if* a good tuple is given to the attacker, it can successfully break RBEs. But to obtain such attacks, we need to at least prove that good tuples *exist* to begin with. That is exactly what the next lemma does.

Lemma 4.6. *Let Π be an RBE scheme with a fixed update graph G. Suppose $\deg^+(G) \le d$ when limited to the first n identities, $\alpha \ge \max_{i\in[n]} |\mathsf{pp}_i|$, and $n \ge \binom{\ell+d}{d+1}$. Then, there exists an $(1-\varepsilon)$-good tuple (i,j,k) for $\varepsilon = \sqrt{\frac{\alpha \ln 2}{2\ell}}$.*

Proof (of Lemma 4.6). From Theorem 3.2 we know that there is a skipping sequence $\{s_1 < s_2 < \cdots < s_\ell\}$ in G. Below, we show that there exists $t \in [\ell]$ such that $(s_t, s_{t+1}-1, s_\ell)$ is both good for G and $(1-\varepsilon)$-good for Π. For $t = \ell$, we define $s_{\ell+1} = s_\ell + 1$ for simplicity so that the selected tuple is well defined, and it will be (s_ℓ, s_ℓ, s_ℓ). We now show that *every* such tuple is good for G, and that at least one of them is $(1-\varepsilon)$-good for Π.

Good for G. By definition of skipping sequences, for all $t \in [\ell]$, any outgoing edge (s_t, v) will either satisfy $v < s_{t+1}$ or $v > s_\ell$. (This also holds for $t = \ell$, as this condition becomes always true in that case). Therefore $(s_t, s_{t+1}-1, s_\ell)$ is good for G for all $t \in [\ell]$.

Good for Π. For $i \in [\ell - 1]$, define the random variable

$$X_0 = (\mathsf{CRS}, \mathsf{KEY}_1, \ldots, \mathsf{KEY}_{s_1-1}),$$
$$X_i = (\mathsf{KEY}_{s_i}, \mathsf{KEY}_{s_i+1}, \ldots, \mathsf{KEY}_{s_{i+1}-1}),$$
$$X_\ell = \mathsf{KEY}_{s_\ell}, Y = \mathsf{pp}_{s_\ell}.$$

Now, by the branching lemma (Lemma 2.7),

$$\mathop{\mathbb{E}}_{t\leftarrow[\ell]}[\mathrm{SD}(X_0 \ldots X_t Y, X_0 \ldots X_{t-1} X'_t Y)] \le \sqrt{\frac{H(Y)\ln 2}{2\ell}} \le \sqrt{\frac{\alpha \ln 2}{2\ell}}$$

in which X'_t is sampled independently of Y. As a result, one can fix $t \in [\ell]$ in the expectation above so that the inequality still holds. In our setting, this means $(s_t, s_{t+1}-1, s_\ell)$ is $(1-\varepsilon)$-good for Π where $\varepsilon = \sqrt{\frac{\alpha \ln 2}{2\ell}}$. □

4.2 Non-uniform attacks using good tuples as advice

We now present an attack that takes a tuple (i,j,k) that is guaranteed to be $(1-\varepsilon)$-good. (Namely, it is both good for G and $(1-\varepsilon)$-good for Π, where $\varepsilon = \sqrt{\frac{\alpha \ln 2}{2\ell}}$). Looking ahead, we will later prove that a simple modification of the attack will succeed even without the given advice.

Construction 4.7 (Attacking RBE with advice) *Let Π be an RBE scheme with update graph G. Suppose (i,j,k) is given as advice. The adversary $\mathcal{A}(i,j,k)$ proceeds as the following.*

1. *Identities: The adversary registers identities* $1, \ldots, k$*, while none are corrupted. Therefore, the adversary can reconstruct all the intermediate auxiliary information and the public parameters, including* pp_k*.*[13]
2. *Target identity: the adversary announces* i *to be the target identity.*
3. *The adversary simply picks messages* $m_0 = 0, m_1 = 1$*, one of which will be encrypted to identity* i *under the public parameter* pp_k*.*
4. *To guess which message was decrypted, the adversary does as follows.*
   (a) *Re-sample fake keys* $\mathsf{sk}'_i, \mathsf{pk}'_i, \ldots, \mathsf{pk}'_j$ *for identities* $i, i+1, \ldots, j$*.*
   (b) *Use* $\mathsf{pk}'_i, \ldots, \mathsf{pk}'_j$ *to* re-register *all the identities* $i, \ldots, j$ *starting from the auxiliary information* aux_{i-1} *that refers to the auxiliary information of the system for the moment before identity* i *registers. As mentioned above,* aux_{i-1} *is known to adversary, as it is a deterministic (efficiently computable) function of* $\mathsf{crs}, \mathsf{pk}_1, \ldots, pk_{i-1}$*.*
   (c) *Let* u' *be the fake update that is generated for the identity* i*, while (fake) registering identities* $i, \ldots, j$*.*
   (d) *Use* sk'_i *and* u' *and try to decrypt the challenge as* $m' \leftarrow \mathsf{Dec}(\mathsf{sk}'_i, \mathsf{u}', \mathsf{ct})$*.*
   (e) *If* $m' \in \{0, 1\}$*, then simply output* m'*.*

The claim below is sufficient for proving Theorem 4.1, as explained above.

Claim 4.8. The adversary of Construction 4.7, once given an $(1-\varepsilon)$-good tuple (i, j, k) where $\varepsilon = \sqrt{\frac{\alpha \ln 2}{2\ell}}$, it succeeds in winning the security game of Definition 2.2 with probability $\rho - \varepsilon$.

Proof (of Claim 4.8). We will consider two worlds $\mathsf{Real}, \mathsf{Ideal}$.

1. Real: HERE the adversary $\mathcal{A}$ behaves as described in attack 4.7.
2. Ideal: This is the world where the adversary is *given* the real keys (including the decryption key of the target identity).

Claim 4.8 directly follows from the following two Claims 4.9 and 4.10.

Claim 4.9. The adversary in Ideal wins with probability $\geq \rho$, where ρ is the completeness probability of the scheme.

Proof. Claim 4.9 directly follows from the ρ-completeness of the RBE scheme and the fact that the given advice (i, j, k) is good for G. In particular, the fact that (i, j, k) is good for G implies that the ith identity will not receive any updates between the jth registration till end of kth registration. Therefore, all the updates received before the jth registration would be enough for decrypting a ciphertext that is encrypted using the kth public parameter. □

Claim 4.10. Let p_R be the probability that the adversary successfully decrypts $m' = m_b$ in the world Real, and let P_I be the corresponding probability in the Ideal world. Then, $|p_R - p_I| \leq \varepsilon$.

[13] Note that the public keys and the CRS will still be given to the adversary.

Proof. Claim 4.10 directly follows from the fact that the given advice (i, j, k) is $(1-\varepsilon)$-good for Π. The key idea is that even though there are going to be updates generated for the ith person till the encryption happens after the kth registration, these updates are all functions of the keys of the first j parties. More formally, Since (i, j, k) is $(1-\varepsilon)$-good for Π, the following two random variables $\mathsf{D}_I, \mathsf{D}_R$ stay ε-close, in which R_E is the randomness for encryption and $\mathsf{M} \leftarrow \{0,1\}$ is the random challenge bit to be encrypted:

$$\mathsf{D}_I \equiv (\mathsf{CRS}, \mathsf{KEY}_1 \dots \mathsf{KEY}_j, \mathsf{PP}_k, \mathsf{R}_E, \mathsf{M}),$$

$$\mathsf{D}_R \equiv (\mathsf{CRS}, \mathsf{KEY}_1 \dots \mathsf{KEY}_{i-1}, \mathsf{KEY}'_i \dots \mathsf{KEY}'_j, \mathsf{PP}_k, \mathsf{R}_E, \mathsf{M})$$

Now, observe that actions of the adversary in the Real and Ideal followed by encryption of the challenge bit $b \leftarrow \mathsf{M}$ and decrypting it only differs based on whether we use D_I or D_R. In particular, all the updates generated for the ith identity generated after jth registration and before $(k+1)$st registration are deterministic functions of the first k keys and the CRS (as KC is a deterministic algorithm). By the data-processing inequality, the probability of successfully decrypting back $b \leftarrow \mathsf{M}$ in the security game will not change by more than ε across the experiments Real and Ideal. This finishes the proof of Claim 4.10. □

This finishes the proof of Claim 4.8. □

4.3 Efficient uniform attack without advice

Finally, we prove Theorem 4.1 using the results from the above subsections.

Proof (of Theorem 4.1). If one could *test* whether a given (i, j, k) is a good tuple, there would be no need to have one explicitly given as in Construction 4.7, because the adversary could enumerate all $(i \le j \le k) \in [n]^3$ cases. Unfortunately, we do not know how to test being $(1-\varepsilon)$-good for Π, even if we could test being good for G (e.g., due to knowing G explicitly). However, we can do as follows.

- **Defining good tuples for attack.** For fixed parameters (including ε) define (i, j, k) to be δ-*good* for attack, if by using (i, j, k) in Construction 4.7, the adversary wins the attack with probability at least $\rho - \varepsilon - \delta$. When $\delta = 0$, simply call (i, j, k) good for the attack.
- **There are tuples that are good for the attack.** Due to Claim 4.8, we already know that there exist tuples that are both good for G and $(1-\varepsilon)$-good for the RBE scheme. Therefore, by Claim 4.8, there exists at least one tuple that is good for the attack.
- **Finding tuples that are good for the attack.** Finally, we observe that, even though one might not be able to directly test whether a given (i, j, k) is good for G or Π, one can indeed find a tuple (i, j, k) that is guaranteed to be δ-good for the attack in time $\mathrm{poly}(\kappa/\delta)$. All the adversary does is to go over all $(i \le j \le k) \in [n]^3$ tuples, run the attack enough *in its head* for $q = \kappa/\delta$ times to approximate its probability of success within $\pm\delta$ with probability

$1 - \text{negl}(\kappa)$. Finally, the adversary simply uses the tuple (i, j, k) that leads to the maximum (estimated) probability of success. Since we already know that there is at least one tuple that is good for the attack, the adversary can always find a δ-good one this way.

Putting this together, the adversary first finds a $(\delta/2)$-good tuple for the attack with probability $1 - \text{negl}(\kappa)$, and then plugs it into Construction 4.7 and runs the attack against the challenger. The overall probability of adversary's success, this way, will be at least $\rho - \varepsilon - \delta/2 - \text{negl}(\kappa) \geq \rho - \varepsilon - \delta$. □

4.4 Extensions

In this subsection, we show some extensions to our main result Theorem 4.1.

Allowing update times to depend on identities. We first observe that the lower bound of Theorem 4.1 holds even if the update graph G of the scheme can depend on the registered identities and the CRS (but not on the keys).

It is easier to see that the update graph G can depend on the name of the registered identities, as the adversary simply picks the identities to be $\mathsf{id}_i = i$ and fixes them throughout the attack.

Allowing dependence on the CRS is slightly more subtle. In summary, this dependence is allowed, because the CRS is sampled and fixed *before* the keys are sampled. In more detail, the main observation is that for *every* $\mathsf{crs} \leftarrow \mathsf{CRS}$, the following hold.

1. The update graph G_{crs} is fixed (with bounded out-degree). Therefore, there will be a skipping sequence in G_{crs}, as proved in Theorem 3.2.
2. Definition 4.4 of $(1 - \varepsilon)$-good can be adapted for any fixed crs.
3. Therefore, Lemma 4.6 can be stated and proved (using the same exact proof) for the fixed crs, showing that a good tuple exists conditioned on crs.
4. Tuples that are good conditioned on the fixed crs can be used exactly as before to break the RBE scheme.

Allowing frequent updates for some identities. Theorem 4.1 is stated for schemes in which *all* parties receive up to d updates. However, a closer look at the proof reveals that all we need is a fixed update graph[14] such that there are *at least* $n \geq \binom{\ell+d}{d+1}$ identities who receive at most d updates and that $|\mathsf{pp}_i| \leq \alpha$ for all the first i registrations. The conclusion of Theorem 4.1 holds as stated.

To see why the above mentioned extension holds, all we have to show is that any update graph G with $n \geq \binom{\ell+d}{d+1}$ vertices of out-degree at most d has a tuple that is $(1 - \varepsilon)$-good. In order to show such tuples exist, all we have to do is to show that sufficiently large skipping sequences exist in DAGs with sufficiently many nodes with bounded out-degrees. Although we can prove such a result by adapting the proof of Theorem 3.2, we prove this extension through a black-box use of Theorem Theorem 3.2.

[14] As discussed before, this graph can depend on the identities and/or the CRS.

Theorem 4.11 (Skipping sequences from sufficiently many nodes of bounded out-degrees). *Let G be a forward DAG and there exists $\mathcal{S} \subseteq \mathcal{V}_G$ such that: $|\mathcal{S}| = \binom{k+d}{d+1}$ (for $k, d \in \mathbb{N}$) and $\deg^+(u) \leq d$ for all $u \in \mathcal{S}$. Then, there exists a skipping sequence in G of size at least k.*

Proof. Let $\mathcal{S} = \{v_1 < v_2 \cdots < v_n\}$. Construct a DAG $G_{\mathcal{S}}$ as follows. $G_{\mathcal{S}}$ has n vertices. For the convenience of presentation, we keep the labels of the vertices of $G_{\mathcal{S}}$ as $\{v_1 < v_2 \cdots < v_n\}$ and do not rename them to $[n]$. For any edge $(v_i, v) \in G$, let $j \in [n]$ be the largest number such that $v_j \leq v$ (note that j always exits and it could be the same as i), and then add the edge (v_i, v_j) to $G_{\mathcal{S}}$.

Observe that the out-degrees $G_{\mathcal{S}}$ remain at most d. That is because we do not add any outgoing edges to any vertex (although we do add incoming edges to some). Therefore, by Theorem 3.2, there is a skipping sequence $\mathcal{U} = \{u_1 < \cdots < u_k\}$ in the graph $G_{\mathcal{S}}$. We claim that the same sequence $\mathcal{U}$ is also skipping in G. Below, we prove this by contradiction.

Suppose $\mathcal{U}$ is *not* skipping in G. This means that there is $t \in [k-1]$ and an edge $(u_t, v) \in G$ such that $u_{t+1} \leq v \leq u_k$. By the definition of $G_{\mathcal{S}}$, the edge $(u_t, v) \in G$ will generate an edge $(u_t, v_j) \in G_{\mathcal{S}}$ for $u_{t+1} \leq v_j \leq u_k$, which contradicts the assumption that $\mathcal{U}$ is a skipping sequence in $G_{\mathcal{S}}$. □

Putting all the extensions above together, we obtain the following theorem.

Theorem 4.12 (Extension of the main result). *Let Π be an RBE scheme with completeness probability ρ whose update graph G is fixed for any fixed sequence $\mathsf{id}_1, \mathsf{id}_2, \ldots$ of identities and fixed CRS crs. Moreover, suppose the update graph for the fixed crs and the fixed set of n identities has at least $\binom{\ell+d}{d+1}$ identities who receive at most d updates, and that $|\mathsf{pp}_i| \leq \alpha$ for all the n registrations. Then there is a 0-corruption $\mathrm{poly}(\kappa)$-time adversary who breaks Π with probability $\rho - \sqrt{\alpha \ln 2/(2\ell)} - \delta$ for arbitrarily small $\delta = 1/\mathrm{poly}(\kappa)$.*

Handling schemes with an amortized bound on the number of updates. The extension above allows us to use an amortized (i.e., average-case) upper bound on the number of updates as well. For example, if the expected number of updates is d among n registered parties, then by an averaging argument, for at least $n/2$ of the parties, the number of received updates is at most $2d$. As a result, if the public parameter remains at most α bits and $n \geq 2 \cdot \binom{\ell+2d}{2d+1}$, then the adversary can break the RBE scheme with probability $\rho - \sqrt{\alpha \ln 2/(2\ell)} - \delta$.

Acknowledgements We thank Sanjam Garg, Mohammad Hajiabadi, and Saeed Mahloujifar for useful discussions. We also thank the anonymous reviewers of TCC 2022 for useful suggestions, including the extension of the main result allowing some identities to receive frequent updates.

References

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. 2, 10

BdM94. Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 3

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. 2, 10

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. 3

BH22. Jeremiah Blocki and Blake Holman. Sustained space and cumulative complexity trade-offs for data-dependent memory-hard functions. Cryptology ePrint Archive, Paper 2022/832, 2022. https://eprint.iacr.org/2022/832. 10

BP97. Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 480–494, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 3

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. 10

CES21. Kelong Cong, Karim Eldefrawy, and Nigel P Smart. Optimizing registration based encryption. In *IMA International Conference on Cryptography and Coding*, pages 129–157. Springer, 2021. 10

Cho09. Sherman SM Chow. Removing escrow from identity-based encryption. In *International Workshop on Public Key Cryptography*, pages 256–276. Springer, 2009. 10

CL02. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 61–76, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. 3

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. 3

EKW19. Keita Emura, Shuichi Katsumata, and Yohei Watanabe. Identity-based encryption with security against the KGC: a formal model and its instantiation from lattices. In *European symposium on research in computer security*, pages 113–133. Springer, 2019. 10

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. 3

GHM+19. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. 3, 4, 10

GHMR18. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 2, 3, 4, 10, 27

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. 10

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. 10

GV20. Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 3, 4, 10

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. 3

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021. 3

Rog15. Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. https://eprint.iacr.org/2015/1162. 2

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. 2

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. 10

Appendix

A Completeness and security of RBE schemes

Definition A.1 (Completeness, compactness, and efficiency of RBE). *Consider the following game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$ *between a challenger* $\mathcal{C}$ *and an interactive* computationally unbounded *adversary* $\mathcal{A}$ *who is yet limited to* $\mathrm{poly}(\kappa)$ *rounds of interaction.*

1. ***Initialization.*** $\mathcal{C}$ *sets* $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{u} = \bot$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \bot$, $t = 0$, *and* $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$, *and sends the sampled* crs *to* $\mathcal{A}$.
2. *Till* $\mathcal{A}$ *continues (which is at most* $\mathrm{poly}(\kappa)$ *steps), proceed as follows. At every iteration,* $\mathcal{A}$ *chooses exactly one of the actions below to perform.*
   (a) ***Registering a corrupted (non-target) identity.*** $\mathcal{A}$ *sends some* $\mathsf{id} \notin \mathcal{D}$ *and* pk *to* $\mathcal{C}$. $\mathcal{C}$ *registers* $(\mathsf{id}, \mathsf{pk})$ *by letting* $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and* $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.
   (b) ***Registering the (uncorrupted) target identity.*** *This step is allowed only if* $\mathsf{id}^* = \bot$. *In that case,* $\mathcal{A}$ *sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to* $\mathcal{C}$. $\mathcal{C}$ *then samples* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\kappa)$, *updates* $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$ *and* $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, *and sends* pk^* *to* $\mathcal{A}$.
   (c) ***Encrypting for the target identity.*** *This step is allowed only if* $\mathsf{id}^* \neq \bot$. *In that case,* $\mathcal{C}$ *sets* $t = t + 1$. $\mathcal{A}$ *sends* $\mathsf{m}_t \in \{0,1\}^*$ *to* $\mathcal{C}$ *who then sets* $\mathsf{m}'_t \coloneqq \mathsf{m}_t$ *and sends back a corresponding ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ *to* $\mathcal{A}$.
   (d) ***Decryption for the target identity.*** $\mathcal{A}$ *sends a* $j \in [t]$ *to* $\mathcal{C}$. $\mathcal{C}$ *then lets* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$. *If* $\mathsf{m}'_j = \mathsf{GetUpd}$, $\mathcal{C}$ *gets* $\mathsf{u} = \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ *and then* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$.

Let $n = |\mathcal{D}|$ *be the number of identities registered when the adversary ends the game. We require the following properties to hold for such* $\mathcal{A}$ *(as specified above) in the game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$.

- ***Completeness.*** *The adversary* $\mathcal{A}$ *wins, if there is some* $j \in [t]$ *for which* $\mathsf{m}'_j \neq \mathsf{m}_j$. *We require that* $\mathsf{P}[\mathcal{A}\,\textit{wins}\,\mathsf{Comp}_{\mathcal{A}}(\kappa)] = \mathrm{negl}(\kappa)$.[15]
- ***Compactness and efficiency.*** *For the following three properties, here we state the default requirements for standard RBE; however, in this work, we also consider the relaxed version of RBE in which these quantities could be other parameters that are still sublinear in* n *(e.g.,* $\mathrm{poly}(\kappa) \cdot \sqrt{n}$*) for compactness and runtime efficiency. For number of updates, we also allow any sublinear function of* n *to be a feasible number for RBE.*
  - ***Compactness.*** $|\mathsf{pp}|, |\mathsf{u}| \leq \mathrm{poly}(\kappa, \log(n))$.
  - ***Efficiency of runtime of registration and update.*** *The running time of each invocation of* Reg *and* Upd *is at most* $\mathrm{poly}(\kappa, \log(n))$.
  - ***Efficiency of the number of updates.*** *The total number of invocations of* Upd *for identity* id^* *in Step 2(d) of the game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$ *is at most* $O(\log(n))$.

[15] For perfectly complete schemes we require this probability to be zero.

Definition A.2 (Security of RBE). *For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathsf{Sec}_{\mathcal{A}}(\kappa)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. ***Initialization.*** *$\mathcal{C}$ sets $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \bot$, $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\kappa)}$ and sends the sampled crs to $\mathcal{A}$.*
2. *Till $\mathcal{A}$ continues (which is at most $\mathrm{poly}(\kappa)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to perform.*
   (a) ***Registering non-target identity.*** *$\mathcal{A}$ sends some $\mathsf{id} \notin \mathcal{D}$ and pk to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk})$ by $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}\}$.*
   (b) ***Registering the target identity.*** *This step can be run only if $\mathsf{id}^* = \bot$. $\mathcal{A}$ sends some $\mathsf{id}^* \notin \mathcal{D}$ to $\mathcal{C}$. $\mathcal{C}$ then samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\kappa)$, updates $\mathsf{pp} \coloneqq \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} \coloneqq \mathcal{D} \cup \{\mathsf{id}^*\}$, and sends pk^* to $\mathcal{A}$.*
3. ***Encrypting for the target identity.*** *If $\mathsf{id}^* = \bot$, then $\mathcal{A}$ first sends some $\mathsf{id}^* \notin \mathcal{D}$ to $\mathcal{C}$ (this is for modeling encryptions for non-registered target identities.) Next $\mathcal{A}$ sends two messages m_0, m_1 of the same length to C. Next, $\mathcal{C}$ generates $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_b)$, where $b \leftarrow \{0,1\}$ is a random bit, and sends ct to $\mathcal{A}$.*[16]
4. *The adversary $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.*

An RBE scheme is secure if for all PPT $\mathcal{A}$, $\mathsf{P}[\mathcal{A}\, wins\, \mathsf{Sec}_{\mathcal{A}}(\kappa)] < \frac{1}{2} + \mathrm{negl}(\kappa)$.

B Information-theoretic notions and lemmas

Definition B.1 (Mutual information). *The mutual information of two discrete random variables X, Y is defined as*

$$I(X;Y) = H(X) + H(Y) - H(XY) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

The conditional mutual information $I(X;Y|Z)$ is defined as $\mathbb{E}_{z \leftarrow Z}[I(X|_z; Y|_z)]$. The chain rule for mutual information states that $I(X;YZ) = I(X;Y) + I(X;Z|Y)$.

Definition B.2 (Kullback–Leibler divergence). *For any two random variables X and Y where $X \subseteq Y$, the he Kullback-Leibler (KL) divergence (in base 2) is defined as*

$$D_{\mathrm{KL}}(X \parallel Y) = \sum_{x \in X} \mathsf{P}_X[x] \cdot \log \frac{\mathsf{P}_X[x]}{\mathsf{P}_Y[x]}.$$

Lemma B.3 (Conditional mutual information vs. KL div). *For any three jointly distributed random variables X, Y, Z the following holds:*

$$I(X;Y|Z) = D_{\mathrm{KL}}(XYZ \parallel (X|_Z \otimes Y|_Z)Z).$$

In particular, when Z does not exist, we have $I(X;Y) = D_{\mathrm{KL}}(XY \parallel X \otimes Y)$.

[16] In the original paper of [GHMR18], the scheme's security was defined for bit encryption. Even though secure bit-encryption schemes can be extended for full-fledged schemes by independently encrypting every bit, here we write the definition directly for the resulting scheme.

We give a proof for completeness.

Proof. By definition, we know

$$I(X;Y|Z) = \mathop{\mathbb{E}}_{z \leftarrow Z}[I(X|_z; Y|_z)] = \sum_{z \in Z} \mathsf{P}_Z[z] I(X|_z; Y|_z).$$

Now, if we call $P = X|_z$ and $Q = Y|_z$, then $I(X|_z; Y|_z) = I(P;Q)$ is equal to:

$$\begin{aligned}
&= H(P) + H(Q) - H(PQ) \\
&= -\sum_{x \in P} \mathsf{P}_P[x] \log \mathsf{P}_P[x] - \sum_{y \in Q} \mathsf{P}_Q[y] \log \mathsf{P}_Q[y] + \sum_{x \in P} \sum_{y \in Q} \mathsf{P}_{PQ}[x,y] \log \mathsf{P}_{PQ}[x,y] \\
&= -\sum_{x \in P} \sum_{y \in Q} \mathsf{P}_{PQ}[x,y] \log \mathsf{P}_P[x] - \sum_{y \in Q} \sum_{x \in P} \mathsf{P}_{PQ}[x,y] \log \mathsf{P}_Q[y] \\
&\qquad + \sum_{x \in P} \sum_{y \in Q} \mathsf{P}_{PQ}[x,y] \log \mathsf{P}_{PQ}[x,y] \\
&= \sum_{x \in P} \sum_{y \in Q} \mathsf{P}_{PQ}[x,y] \log \frac{\mathsf{P}_{PQ}[x,y]}{\mathsf{P}_P[x] \cdot \mathsf{P}_Q[y]}.
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
I(X;Y|Z) &= \sum_{z \in Z} \left(\mathsf{P}_Z[z] \sum_{x \in X} \sum_{y \in Y} \mathsf{P}_{XY|z}[x,y] \log \frac{\mathsf{P}_{XY|z}[x,y]}{\mathsf{P}_{X|z}[x] \cdot \mathsf{P}_{Y|z}[y]} \right) \\
&= \sum_{z \in Z} \sum_{x \in X} \sum_{y \in Y} \mathsf{P}_{XYZ}[x,y,z] \log \frac{\mathsf{P}_{XY|z}[x,y]}{\mathsf{P}_{X|z}[x] \cdot \mathsf{P}_{Y|z}[y]} \\
&= \sum_{z \in Z} \sum_{x \in X} \sum_{y \in Y} \mathsf{P}_{XYZ}[x,y,z] \log \frac{\mathsf{P}_{XY|z}[x,y] \cdot \mathsf{P}_Z[z]}{\mathsf{P}_{X|z}[x] \cdot \mathsf{P}_{Y|z}[y] \cdot \mathsf{P}_Z[z]} \\
&= \sum_{z \in Z} \sum_{x \in X} \sum_{y \in Y} \mathsf{P}_{XYZ}[x,y,z] \log \frac{\mathsf{P}_{XYZ}[x,y,z]}{\mathsf{P}_{X|z}[x] \cdot \mathsf{P}_{Y|z}[y] \cdot \mathsf{P}_Z[z]} \\
&= D_{\mathrm{KL}}(XYZ \parallel (X|_Z \otimes Y|_Z)Z).
\end{aligned}$$

□

Theorem B.4 (Pinsker's inequality). *For random variables X, Y we have*

$$\mathrm{SD}(X,Y) \le \sqrt{\frac{D_{\mathrm{KL}}(X \parallel Y) \cdot \ln 2}{2}}$$

The following lemma follows from Lemma B.3 and Pinsker's inequality.

Lemma B.5. *For random variables X, Y, Z, it holds that*

$$\mathrm{SD}(XYZ, (X|_Z \otimes Y|_Z)Z) \le \sqrt{\frac{I(X;Y|Z) \cdot \ln 2}{2}}.$$

In particular, when Z does not exist, we have $\mathrm{SD}(XY, X \otimes Y) \le \sqrt{\frac{I(X;Y) \cdot \ln 2}{2}}$.

We finally prove the twig lemma (i.e., Lemma 2.7).

Proof (of Lemma 2.7). Let $I(Y; X_0 \dots X_\ell) = \alpha$ and $\alpha_i = I(Y; X_i | X_{i-1} \dots X_0)$. Firstly, we have $\alpha = H(Y) - H(Y|X_0 \dots X_\ell) \le H(Y)$. By repeated applications of the chain rule of mutual information,

$$H(Y) \ge \alpha = I(Y; X_0) + \sum_{i \in [\ell]} I(Y; X_i | X_0 \dots X_{i-1}) \ge \ell \cdot \mathop{\mathbb{E}}_{i \in [\ell]} [\alpha_i].$$

For each $i \in [\ell]$, we get $\alpha_i = D_{\mathrm{KL}}(Y X_i \dots, X_0 \parallel Y X_i' X_{i-1} \dots X_0)$ by letting $X = X_i, Z = X_0 \dots X_{i-1}$ in Lemma B.3. By applying Pinsker's inequality through Lemma B.5 we now get

$$\mathrm{SD}(Y X_i \dots, X_0, Y X_i' X_{i-1} \dots X_0) \le \sqrt{\frac{\alpha_i \ln 2}{2}}.$$

To conclude, we get

$$\mathop{\mathbb{E}}_{i \leftarrow [\ell]} \left[\sqrt{\frac{\alpha_i \ln 2}{2}} \right] \le \sqrt{\frac{\mathbb{E}_{i \leftarrow [\ell]} [\alpha_i \ln 2]}{2}} \le \sqrt{\frac{(\alpha/\ell) \ln 2}{2}} \le \sqrt{\frac{H(Y) \ln 2}{2\ell}}.$$

The first inequality is due to the concavity of $\sqrt{\cdot}$, and Jensen's inequality. □

C Theorem 3.2 is Optimal

In this section, we show that the bound in Theorem 3.2 is tight. Namely, we prove the following theorem.

Theorem C.1 (Optimality of Theorem 3.2). *For all $n = \binom{k+d}{d+1} - 1$ where integers $k \ge 1, d \ge 0$, there exists a forward DAG $G_{k,d}$ of n vertices and $\deg^+(G_{k,d}) \le d$ that does* not *have any skipping sequence of size k.*

We will use induction on d to prove Theorem C.1.

Construction C.2 (Construction of optimal DAG of out-degree d) *Let $k \ge 1, d \ge 0$ be integer. We construct a graph $G_{k,d}$ recursively as follows.*

1. *If $d = 0$, $G_{k,0}$ has $k - 1$ vertices and no edges.*[17]
2. *If $d \ge 1$, do the following.*
   (a) *For $i \in [k-1]$ let $\mathcal{G}_i$ be a copy of $G_{k-i+1,d-1}$ followed by a new vertex u_i at the end. Moreover, in addition to the edges in $G_{k-i+1,d-1}$, for all $v \in \mathcal{G}_i$ (including $v = u_i$) add the edge (v, u_i) to $\mathcal{G}_i$.*
   (b) *Divide the vertices of $G_{k,d}$ into $k - 1$ groups, such that the i-th group is a copy of $\mathcal{G}_i$ that comes right after $\mathcal{G}_{i-1}$.*

To prove Theorem C.1, it suffices to prove the following lemma.

[17] For $k = 1$, this graph is the empty graph that has no vertices.

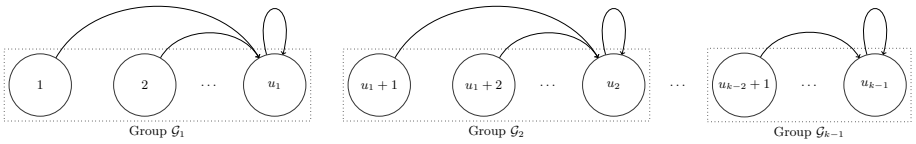

Fig. 3. Illustration of the construction of the optimal forward DAG $G_{k,d} = (\mathcal{V}_{G_{k,d}}, \mathcal{E}_{G_{k,d}})$ where $\mathcal{V}_{G_{k,d}} = [\binom{k+d}{d+1} - 1]$. Group $\mathcal{G}_1$ has $\binom{k+d-1}{d}$ vertices and each vertex of $\mathcal{G}_1$ has an out-going edge to vertex u_1 which is the last vertex of $\mathcal{G}_1$. Group $\mathcal{G}_{k-1}$ has $\binom{d+1}{d}$ vertices and each vertex of $\mathcal{G}_{k-1}$ has an out-going edge to vertex u_{k-1}.

Lemma C.3. *Graph $G_{k,d}$ of Construction C.2 has $n = \binom{k+d}{d+1} - 1$ vertices, degree d and all of its skipping sequences are of size at most $k-1$.*

Note that by Theorem C.1 we already know that if $k \geq 1$, then $G_{k,d}$ has a skipping sequence of size $\geq k-1$; so by proving Lemma C.3 we actually conclude that its maximum size of skipping sequences will be exactly $k-1$.

Proof (of Lemma C.3). The proof is by induction. For $d = 0$, the proof is trivial.

Now suppose $d \geq 1$. The number vertices of $G_{k,d}$ by induction and the hockey-stick identity will be

$$\sum_{i \in [k-1]} |\mathcal{V}_{G_{k-i+1,d-1}}| + 1 = \sum_{i \in [k-1]} \binom{k+d-i}{d} = \binom{k+d}{d+1} - \binom{d}{d}.$$

Let $\mathcal{S}$ be any skipping sequence in $G_{k,d}$. Let $j \in [k-1]$ be the largest integer such that there is a vertex from $\mathcal{G}_j$ in $\mathcal{S}$. We first show that there can be at most one vertex from each of the previous $j-1$ groups $\{\mathcal{G}_i\}_{i \in [j-1]}$ in $\mathcal{S}$. Assume that there are two vertices $u < v$ such that $u, v \in \mathcal{S} \cap \mathcal{G}_i$. Let $x \in \mathcal{G}_j \cap \mathcal{S}$. Then, $v < u < x$ will all be in $\mathcal{S}$, while v has an outgoing edge to u_i (the last vertex in $\mathcal{G}_i$) with $u \leq u_i < x$, but this contradicts the definition of skipping sequences.

Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, where $\mathcal{S}_1 = \mathcal{S} \cap (\cup_{i<j} \mathcal{G}_i)$ and $\mathcal{S}_2 = \mathcal{S} \cap \mathcal{G}_j$. We already know that $|\mathcal{S}_1| \leq j-1$. It is sufficient to show that $|\mathcal{S}_2| \leq k-j$. Firstly, note that if u_j (i.e., the last node in $\mathcal{G}_j$) belongs to $\mathcal{S}$, then no other vertex in $\mathcal{G}_j$ can belong to $\mathcal{S}$, as otherwise, it will contradict the definition of skipping sequences. Secondly, note that if $\mathcal{S}$ is a skipping sequence, then its restriction $\mathcal{S}_2 = \mathcal{S} \cap \mathcal{G}_j$ shall be skipping as well. Therefore, by induction $|\mathcal{S}_2| \leq \max\{1, k-j\} = k-j$, and so $|\mathcal{S}| = |\mathcal{S}_1| + |\mathcal{S}_2| \leq j - 1 + k - j = k - 1$. □