Registration-Based Encryption in the Plain Model

Jesko Dujmovic[1], Giulio Malavolta[2], and Wei Qi[2]

[1]CISPA Helmholtz Center for Information Security and Saarland University
jesko.dujmovic@cispa.de
[2]Bocconi University
giulio.malavolta@hotmail.it, wei.qi@unibocconi.it

March 21, 2025

Abstract

Registration-based encryption (RBE) is a recently developed alternative to identity-based encryption, that mitigates the well-known key-escrow problem by letting each user sample its own key pair. In RBE, the key authority is substituted by a key curator, a completely transparent entity whose only job is to reliably aggregate users' keys. However, one limitation of all known RBE scheme is that they all rely on one-time trusted setup, that must be computed honestly.

In this work, we ask whether this limitation is indeed inherent and we initiate the systematic study of RBE in the *plain model*, without any common reference string. We present the following main results:

- (Definitions) We show that the standard security definition of RBE is unachievable without a trusted setup and we propose a slight weakening, where one honest user is required to be registered in the system.
- (Constructions) We present constructions of RBE in the plain model, based on standard cryptographic assumptions. Along the way, we introduce the notions of non-interactive witness indistinguishable (NIWI) proofs secure against chosen statements attack and re-randomizable RBE, which may be of independent interest. A major limitation of our constructions, is that users must be updated upon every new registration.
- (Lower Bounds) We show that this limitation is in some sense inherent. We prove that any RBE in the plain model that satisfies a certain structural requirement, which holds for all known RBE constructions, must update all but a vanishing fraction of the users, upon each new registration. This is in contrast with the standard RBE settings, where users receive a logarithmic amount of updates throughout the lifetime of the system.

Contents

1 Introduction

Not long after the introduction of public-key encryption [DH76], Shamir envisioned a cryptographic primitive that would allow one to encrypt a message by just knowing the identity of the receiver. In an identity-based encryption (IBE), Alice can encrypt her messages to Bob knowing just Bob's identity, along with some additional public parameters. Bob can then decrypt Alice's ciphertexts using an identity-specific secret key that he obtains from the key authority. The work of Boneh and Franklin [BF01], which proposed the first cryptographic construction of IBE, started a fruitful line of investigation in cryptography that lead to new schemes [Wat05, Coc01, Gen06, GPV08, DG17] and new cryptographic primitives generalizing IBE [SW05, GPSW06, BSW11]. However, IBE also comes with important limitations. Most prominently, it suffers from the well-known *key escrow problem*: In an IBE system, the key authority that generates the public parameters and the decryption keys can also decrypt any message ever encrypted. In many cases, this is not an acceptable compromise, and IBE has received strong criticism [Rog15] because of this problem.

A series of recent works [GHMR18, GHM+19, GKMR23, DKL+23] proposes a new notion of encryption aimed at solving this problem. In registration-based encryption (RBE) the trusted authority of IBE is substituted by a completely transparent party, called the key curator (KC). The KC is responsible to aggregate the public keys of the users in a *small digest*, that can be later on used to encrypt with respect to an identity, thus mimicking the functionality of IBE. Differently from IBE however, the public keys are sampled locally by the users themselves, and the role of the KC is limited to reliably store and aggregate the users' keys, similarly to what a public key infrastructure would do. Thus, RBE can be seen as combining the best-of-both-worlds between public-key encryption and IBE. RBE has recently seen a surge of interest, with constructions getting closer and closer to practicality [GKMR23, DKL+23] and works achieving more and more general functionalities [HLWW23, FFM+23].

Despite these promising properties, one shortcoming shared by *all constructions* of RBE, is that they rely on a one-time trusted setup, needed to sample a common reference string (CRS). Granted, this is a one-time operation that can be realized using various techniques (such as running an MPC between mutually distrustful parties), and thus it does not detract substantial value from this notion. Nevertheless, a trusted setup goes against the transparent spirit of RBE, and it would be desirable to avoid it, if possible. Motivated by this concern, in this work we put forward the following question:

Is a trusted setup necessary for RBE?

Before stating the results of this work, we shall make an important distinction between existing RBE schemes. The trusted setup of an RBE can be either *structured* [GKMR23] or *transparent* [GHMR18, GHM+19, DKL+23], where the latter means that it can be sampled using public random coins. In practice, the latter setup is much more desirable, since it is easy and efficient to *heuristically* sample it by, e.g., hashing the current date and time. While these are important considerations, they are not the focus of this work: We are interested in understanding RBE schemes that we can prove secure in the standard model, with no setup at all.

1.1 Our Results

In this work we initiate the systematic study of RBE in the plain model, without any trusted setup. The aim of this work is to develop a formal study of RBE in these settings, and to characterize the efficiency/security tradeoffs of this notion. Overall, our main contributions can be summarized as follows.

- **Definitions:** We extend the definition of RBE to handle the absence of a trusted setup. In contrast with previous definitions [GHMR18], our security experiment requires the existence of *at least one honest user* registered in the system, which results in slightly weaker security. We show that this limitation is necessary in the trustless setting, by presenting a generic attack against any RBE without setup.

- **Constructions:** We present two constructions of RBE without setup. Our first construction is in a relaxed model where the key registration is interactive (in fact, a two-message protocol) and it is based on the combination of re-randomizable RBE and non-interactive witness indistinguisahble (NIWI) proofs secure against *chosen statement* attacks, two primitives that we introduce in this work and that may be of independent interest. We show how to construct the former from the computational Diffie-Hellman assumption, and the latter from a combination of regular NIWIs and CCA-secure commitments.

Our second construction is in the standard communication model of RBE, where the key registration consists of a single message from the user to the KC. This construction additionally assumes the existence of indistinguishability obfuscation [BGI+01].

- **Lower Bounds:** Both of our constructions have an important limitation, namely that upon every registration *all* users' keys must be updated. Thus the constructions are qualitatively closer to the weaker notion of laconic encryption, as proposed in [DKL+23]. We show that this is not a coincidence, by proving any RBE in the plain model that satisfies a certain structural requirement (which holds for all known RBE construction) must update all but a vanishing fraction of the users, upon each new registration. It can be shown that any scheme in the plain model must have $\Omega(n)$ number of updates where n is the number of registered users. This should be contrasted with the standard RBE in the CRS model, where each user receives at most $O(\log n)$ updates.

1.2 Technical Overview

In the following we present a high-level overview of our techniques, focusing more on clarity than precision. For more rigorous statements, we refer the reader to the technical sections.

1.2.1 A Primer on RBE.

To establish some notation, let us first recall in somewhat more details the notion of RBE. In an RBE system, user sample their own key locally, via the KeyGen algorithm, and they then send their public key to the KC, who is responsible for maintaining the public parameters pp of the scheme. This is done via the Reg algorithm, that the KC uses to determine the new public parameters. Since the public parameters must change upon each user registration, it might be the case that a set of other users have to update their decryption key, which the KC computes using the Upd algorithm. The relevant set of users is then notified by the KC of the update. Recall that the update information u is necessary for decryption, but it is not a substitute of the secret key, and in fact u can be published without affecting the security of the scheme. For efficiency reasons, it is important to keep the number of per-user updates at a minimum, and a lot of ingenuity in RBE constructions goes into designing schemes that require as few updates as possible (more on this later). Given the current public parameters pp and an identity id, anyone can encrypt (Enc) a message for the user identified with id. Provided that such users have registered some keys, they can indeed decrypt (Dec) the message, possibly after receiving some update from the KC.

Security is defined via an experiment, where the attacker is allowed to ask honest users to register keys and to register (possibly corrupted) keys itself. At some point of the execution, the attacker submits two messages m_0 and m_1 and an identity id, then the challenger flips a coin $b \in \{0,1\}$ and encrypts m_b with respect to id using the current public parameter. The attacker wins the experiment if it can guess the encrypted message with probability non-negligibly better than $1/2$. Furthermore, the attacker is considered to be admissible if id does not correspond to a corrupted key, at the time of encryption.

1.2.2 Plain Model Impossibility.

At a very high-level, the reason why known constructions of RBE require a common reference string crs, is the fact that the (deterministic) registration algorithm can be thought of as a *hash* function of the public keys and identities of the n registered users. In fact, we can think of the public parameter pp as the output of this hash function. By the compactness and efficiency requirement of RBE, the size of pp is $o(n)$, which means there must be two different sets of n users with the same keys that will result in the exact same hash pp. Note that even thought such a collision always exists, it could depend on crs. But, if there is no crs, a non-uniform adversary can simply hard-wire such a collision and register one set of the users and decrypt successfully with overwhelming probability any message encrypted to any identity in the other set using the common public parameter pp, due to the completeness of RBE, and break the security of RBE.

In more detail, the non-uniform adversary will get as advice a collision for an $i \in [n]$: A list of public and secret keys $(\mathsf{pk}_1, \mathsf{sk}_1), \cdots, (\mathsf{pk}_n, \mathsf{sk}_n)$, and two lists of identities $\mathsf{id}_1, \cdots, \mathsf{id}_n$ and $\mathsf{id}'_1, \cdots, \mathsf{id}'_n$ such that: (i) Registering $(\mathsf{id}_1, \mathsf{pk}_1), \cdots (\mathsf{id}_n, \mathsf{pk}_n)$ and registering $(\mathsf{id}'_1, \mathsf{pk}_1), \cdots, (\mathsf{id}'_n, \mathsf{pk}_n)$ will generate the same public parameter pp and (ii)

the identity id'_i is not in the first list of identities. One can show that such an advice must exist using the compactness requirement $|\mathsf{pp}| < o(n)$. The adversary can ask the challenger to register $(\mathsf{id}_1, \mathsf{pk}_1), \cdots (\mathsf{id}_n, \mathsf{pk}_n)$, which should produce pp. Now, the adversary can simply ask the challenger to encrypt to id'_i using pp and decrypt successfully with overwhelming probability, due to the completeness of the RBE scheme. Note that if an update is needed for decryption, the adversary has all the information it needed to locally compute the correct update u'_i. Since id'_i is not even registered by the challenger, this constitutes a legitimate attack against the security of RBE.

1.2.3 Weakening the Security Model.

As explained above, in normal RBE the attacker can ask the challenger to encrypt the challenge ciphertext with respect to any non-corrupted identity id. Because the attack above particularly uses the fact that the challenge can be with respect to non-registered party, an obvious weakening of the security requirement is to not allow encryptions to non-registered users.

One might argue that this is a realistic security model because a user could check whether an identity is registered before they encrypt with respect to that identity. We believe this goes against the purpose of RBE because this check would require storing information that scales linearly with the number of registered users, which is exactly what RBE is trying to avoid.

Therefore, our positive results are in a security model that sits in between the two. We require at least one honest party being registered. This subsumes the notion that does not allow challenges with respect to non-registered users, because the only non-corrupted but registered users are honest ones, so clearly an honest party has to be registered.

1.2.4 RBE in the Plain Model.

Once we have established the definition that we are aiming for, it is natural to ask whether we can actually build an RBE without a trusted setup. Since we have seen that an RBE inherently is a collision resistant hash function one would hope to just take a construction of an RBE and replace the implied hash function by a keyless hash function. This, however, first require to build RBE from unstructured collision resistant hash functions, which is an interesting open problem in itself. In this work, we propose a construction using the following cryptographic ingredients:

1. A one-way permutation f.
2. A non-interactive witness indistinguishable (NIWI) proof.
3. An RBE with re-randomizable CRS. Loosely speaking, this is an RBE where anyone can take the current CRS and "re-randomize" it, to produce a new CRS that is indistinguishable from a freshly sampled one.

We defer the discussion on how to instantiate these cryptographic building blocks to a later point in this overview, and for now we proceed to explain our general recipe to construct RBE in the plain model. Our starting point is the RBE with re-randomizable CRS, which we refer to as our *base RBE*. Our first modification to the base scheme is that, before registering their key, each honest user re-randomizes the current CRS and submits the new crs' to the KC. Additionally, it samples a random x_i and submits $y_i = f(x_i)$ along with the updated CRS. To ensure that each user re-randomizes the CRS appropriately, we also ask them to include a NIWI proof for the following statement:

- (Regular branch) I updated correctly the CRS, OR
- (Trapdoor branch) I know a valid pre-image x_i such that $f(x_i) = y_i$, AND for *all previous* CRSs sent by the j-th party ($1 \leq j < i$):
  - The j-th update was performed correctly, OR
  - I know the corresponding pre-image x_j.

Clearly, the honest user will always compute the NIWI using the witness for the regular branch, whereas the trapdoor branch will become useful later, during the security proof. Once the new CRS is received, the KC re-registers all existing users against the new CRS (more discussion on this later) and sends out the corresponding updates. At this

point, we have a well-formed RBE system, that can be used to encrypt/decrypt messages using the algorithms provided by the base RBE scheme.

We prove security of this construction with a reduction against the base RBE. Since we know that there exists at least one honest user, our objective will be to *plant* the challenge common reference string crs^* as the CRS for such user. However, in order to do so, the reduction cannot compute the honest branch of the NIWI, since it does not know the witness for the correct update. Instead, the reduction will *extract* the witness of *all previous NIWIs* (this procedure is potentially inefficient, but we will deal with this aspect later), which allows us to compute the new NIWI using the trapdoor branch. This allows us to plant crs^* as the CRS after the honest user registration. However, we still have to deal with updates of the CRS that may happen *after* the honest user is registered. Once again, the reduction extracts the update information from the NIWI supplied by the adversary: This allows us to *translate* the challenge ciphertext computed against crs^* into a ciphertext computed against the current (possibly updated) CRS. At this point, we can reduce the security to that of the base RBE.

Finally, to deal with the fact that the reduction is not efficient (since it requires extracting witnesses from adversarial NIWI proofs), we use a technique commonly referred to as *complexity leveraging*: In brief, we set the security parameter of the base RBE to be large enough so that even an adversary that has enough power to break NIWIs, still cannot break the security of the base RBE scheme.

We conclude this discussion by highlighting the fact that, in the construction as described above, the key generation algorithm depends on the current CRS, which is not standard for RBE, where the key generation only takes as input the security parameter. Concretely, this means that the key generation is a two-message protocol, instead of being completely non-interactive. We can overcome this limitation by additionally assuming the existence of indistinguishability obfuscation: Instead of sending directly the output of the KeyGen algorithm, each user sends an obfuscated circuit that, on input the current CRS, outputs a freshly sampled key (along with the appropriate NIWI proof). The security of this variant can be reduced to the security of the interactive scheme, via a standard puncturing argument.

1.2.5 NIWIs Secure Against Chosen-Statement Attacks.

Next, we discuss how to instantiate the necessary cryptographic building blocks. A subtle aspect of the above security argument is that, at some point of the proof, we have to simultaneously extract the witness from adversarial NIWI proofs, while at the same time appeal to the witness indistinguishability of an honestly computed NIWI. We formalize this property as *security against chosen statement attacks (CSA)*, where we require that witness indistinguishability holds, even in the presence of an oracle that (inefficiently) extracts a witness from any input NIWI. Importantly, the oracle only extracts NIWIs for statements that are not equal to the challenge statement (this is akin to the standard notion of CCA-security for encryption/commitments). To complete the picture, we show that CSA-secure NIWI can be constructed from regular NIWIs, combined with non-interactive CCA-secure commitments, which in turn can be built from a variety of computational assumptions [PPV08, LPS17, BL18, KK19, Khu21].

1.2.6 RBE with Re-Randomizable CRS.

Our starting point is the RBE construction of [GHMR18], where users' keys are hashed in a Merkle tree, and the witness needed to decrypt is the root-to-leaf path for the corresponding identity. The crucial building block is the notion of *hash encryption* [DGHM18], which allows one to encrypt a message with respect to an index i and a bit b: If the i-th bit of the hash pre-image equals b, then decryption is possible. The entire RBE construction reduces to building hash encryption, and therefore for us it will suffice to construct hash encryption with re-randomizable keys. We observe that the CDH chameleon encryption of [DG17] when interpreted as a hash encryption, indeed, has a re-randomizable key. In more details, the hash key corresponds to a matrix

$$\left(g, \begin{pmatrix} g_{1,0} & g_{2,0} & \cdots & g_{n,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{n,1} \end{pmatrix}\right)$$

and to encrypt with respect to a hash h, an index i, and a bit b, one computes a ciphertext

$$\left(h^\rho, \begin{pmatrix} g_{1,0}^\rho & g_{2,0}^\rho & \cdots & g_{i-1,0}^\rho & g_{i+1,0}^\rho & \cdots & g_{n,0}^\rho \\ g_{1,1}^\rho & g_{2,1}^\rho & \cdots & g_{i-1,1}^\rho & g_{i+1,1}^\rho & \cdots & g_{n,1}^\rho \end{pmatrix}\right)$$

and the message is masked with $m \oplus \mathrm{GL}(g_{i,b}^{\rho})$, where GL is the Goldreich-Levin hardcore predicate. Our re-randomization process will select uniform $\{\beta_{i,b}\}$ and re-randomize the CRS by computing

$$\left(g, \begin{pmatrix} g_{1,0}^{\beta_{1,0}} & g_{2,0}^{\beta_{2,0}} & \cdots & g_{n,0}^{\beta_{n,0}} \\ g_{1,1}^{\beta_{1,1}} & g_{2,1}^{\beta_{2,1}} & \cdots & g_{n,1}^{\beta_{n,1}} \end{pmatrix}\right)$$

which is easy to see that it is identically distributed as a freshly sampled CRS. For the security proof, it will be useful also to "update" a ciphertext according to a particular re-randomization factor, which can be easily done by computing

$$\left(\left(\frac{h^{\rho}}{\prod_{j\neq i} g_{j,x_j}^{\rho}}\right)^{\beta_{i,x_i}} \cdot \prod_{j\neq i} g_{j,x_j}^{\rho\cdot\beta_{j,x_j}}, \begin{pmatrix} g_{1,0}^{\rho\cdot\beta_{1,0}} & g_{2,0}^{\rho\cdot\beta_{2,0}} & \cdots & g_{i-1,0}^{\rho\cdot\beta_{i-1,0}} & g_{i+1,0}^{\rho\cdot\beta_{i+1,0}} & \cdots & g_{n,0}^{\rho\cdot\beta_{n,0}} \\ g_{1,1}^{\rho\cdot\beta_{1,1}} & g_{2,1}^{\rho\cdot\beta_{2,1}} & \cdots & g_{i-1,1}^{\rho\cdot\beta_{i-1,1}} & g_{i+1,1}^{\rho\cdot\beta_{i+1,1}} & \cdots & g_{n,1}^{\rho\cdot\beta_{n,1}} \end{pmatrix}\right)$$

where $x = (x_1, \ldots, x_n)$ is the pre-image of h. Once again, it is easy to see that such ciphertext is indeed a well-formed ciphertext under the updated key. In the technical sections, we will actually use a slightly different strategy to prove that security is preserved in the presence of a re-randomized keys, with a direct reduction against the CDH problem. However, the property outlined here provides a good intuition on what enables re-randomization.

1.2.7 Lower Bounds on the Number of Updates.

The savvy reader may have noticed that in our construction each user needs to receive an update every time that a new public key is registered. This is in contrast with the standard notion of RBE, where each user receives at most a logarithmic number of updates throughout the lifetime of the system. We argue this disadvantage is necessary unless we find a completely new way to build registration based encryption by showing a lower bound on the number of updates.

All known constructions follow the same structure: the public parameters consist of one (or more) succinct commitments of the public keys of the users. Then, to decrypt a ciphertext the user knows the opening for his public key, along with the corresponding secret key. We then assume this structure and prove that any plain-model RBE that follows this structure[1] has to update a constant fraction of users.

Intuitively, the argument says if a dishonest user does not get an update when the first honest party joins, then this means the succinct commitment where its identity and public key are did not change. Therefore, these commitments only contain adversarially chosen identities and public keys. Just like in Section 1.2.2 these commitments define a collision resistant hash function on the identities. Similarly, we can use non-uniform advice to make sure that the part of the collision we don't register contains the challenge identity but under a malicious public key.

Notice, this attack chooses the challenge ciphertext to be with respect to an honest party. Therefore, this lower bound also applies in the weaker security notion, where the challenge can not be with respect to a non-registered user.

1.3 Open Problems

We view our work as a starting point for a promising avenue of research, and there is a number of problems that remain open. On the theoretical side, it is interesting to understand what are the minimal assumptions needed to build RBE in the plain model. Regular RBE can be constructed just assuming the hardness of the CDH problem, whereas our construction in the plain model requires much stronger cryptographic tools, such as NIWI proofs and, in some of the settings, indistinguishability obfuscation. Another interesting question is to design an RBE in the plain model with *non-interactive* key generation, where there is no upper bound on the total number of registered users (our *interactive* construction achieves that, but we lose this property once we apply our compiler). Finally, from a more practical standpoint, it would be interesting to design RBE schemes with better concrete efficiency.

[1] This assumption is simplified for exposition purposes. In the actual prove we assume that there exist two deterministic (not necessarily efficient) algorithms Ext_1 and Ext_2 such that $\mathsf{Ext}_1(\mathsf{id}_i, i, \mathsf{pk}_i, \mathsf{u}_i) = \mathsf{Ext}_2(\mathsf{pp}, i)$ if and only if a user with the corresponding sk_i and u_i can successfully decrypt message encrypted with respect to id_i using pp. Think of Ext_1 as extracting the commitments from the update and Ext_2 them from the public parameters.

2 Cryptographic Preliminaries

Throughout this work, we write λ to denote the security parameter. We say a function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every $c > 0$ there exists $n_c \in \mathbb{N}$ such that $f(n) < \frac{1}{n^c}$ for $n > n_c$. We denote by $\mathrm{negl}(\lambda)$ a negligible function of λ. We say an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input. For a randomized algorithm A, the notation $y \leftarrow \mathsf{A}(x)$ means y is sampled by running A on input x and the notation $y \leftarrow \mathsf{A}(x; r)$ means y is computed by running A on input x with explicit randomness r. Furthermore, the notation $y \in \mathsf{A}(x)$ means there is a randomness r such that $y \leftarrow \mathsf{A}(x; r)$.

We recall the computational Diffie-Hellman (CDH) assumption [DH76].

Definition 1 (CDH Assumption)**.** *Let $(p, g, \mathbb{G})$ be a group $\mathbb{G}$ of prime order $p \in O(\lambda)$ with generator g. We say that such group is CDH-hard if there exists a negligible function* negl *such that for all PPT adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[g^{x \cdot y} = \mathsf{Adv}\left(g, g^x, g^y\right)\right] = \mathrm{negl}(\lambda).$$

2.1 CCA-Secure Commitments

In this work we consider the notion of tag-based commitment scheme Com, where we say that the commitment is binding if for all tags t, all messages m_0 and m_1, and all randomnesses r_0 and r_1 it holds that

$$\mathsf{Com}(t, m_0; r_0) \neq \mathsf{Com}(t, m_1; r_1).$$

We additionally require that the commitment satisfies the strong notion of CCA-security [CLP10], which we recall in the following.

Definition 2 (CCA-Security)**.** *A tag-based commitment Com is CCA-secure if there exists a negligible function* negl *such that for all PPT adversaries $\mathcal{A}$, all tags t, and all pairs of messages (m_0, m_1) it holds that*

$$\left|\Pr\left[1 = \mathsf{Adv}^{\mathcal{O}}(t, \mathsf{Com}(t, m_0))\right] - \Pr\left[1 = \mathsf{Adv}^{\mathcal{O}}(t, \mathsf{Com}(t, m_1))\right]\right| \leq \mathrm{negl}(\lambda)$$

where the oracle $\mathcal{O}$ takes as input a tag t^* and a commitment c^*. If $t^* \neq t$, it (inefficiently) computes the message committed in c^* and returns it as an output. If no such message exists, the oracle returns $\perp$.

CCA commitments can be constructed from adaptively secure one-way functions [PPV08], time-lock puzzles [LPS17], obfuscation [Khu21], sub-exponentially hard injective one-way functions [BL18], or quantum-easy commitments [KK19].

2.2 Indistinguishability Obfuscation

We recall the definition of indistinguishability obfuscation (iO) for circuits from [BGI+01, GGH+13].

Definition 3 (iO for Circuits)**.** *A PPT machine Obf is an indistinguishable obfuscator for circuit class $\{\mathcal{C}_\lambda\}$, if the following are satisfied:*

- *(Functional Equivalence) For all $\lambda \in \mathbb{N}$, all $C \in \mathcal{C}_\lambda$, all inputs x, we have*

$$\Pr\left[C'(x) = C(x) : C' \leftarrow \mathsf{Obf}(C)\right] = 1$$

- *(Indistinguishability) For all $\lambda \in \mathbb{N}$, all pairs of circuit $(C_0, C_1) \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and $C_0(x) = C_1(x)$ on all inputs x, it holds that the following distributions are computationally indistinguishable*

$$\mathsf{Obf}(C_0) \approx \mathsf{Obf}(C_1).$$

2.3 Puncturable Pseudorandom Functions

A puncturable pseudorandom function (PRF) is an augmented PRF that has an additional puncturing algorithm. Such an algorithm produces a punctured version of the key that can evaluate the PRF at all points except for the punctured one. It is required that the PRF value at that specific point is pseudorandom even given the punctured key. A puncturable PRF can be constructed from any one-way function [GGM84].

Definition 4 (Puncturable PRFs)**.** *A puncturable family of PRFs is a tuple of PPT algorithms* $(PRF, \mathsf{Puncture})$ *defined as follows.*

- $PRF(K, i) \to y$*: A deterministic algorithm that takes as input a key* $K \in \mathcal{K}$ *and a point* $i \in \mathcal{X}$ *and returns a value* $y \in \mathcal{Y}$*.*
- $\mathsf{Puncture}(K, i) \to K_i$*: A deterministic algorithm that takes as input a key* $K \in \mathcal{K}$ *and a point* $i \in \mathcal{X}$ *and returns a punctured key* K_i*.*

We require that a puncturable PRF satisfies the following properties.

- *(Correctness) For all* $\lambda \in \mathbb{N}$*, for all keys* $K \leftarrow \{0,1\}^\lambda$*, for all* $i \in \mathcal{X}$ *and* $x \in \mathcal{X} \setminus i$*, and for all* $K_i \leftarrow \mathsf{Puncture}(K, i)$*, we have that*
$$PRF(K_i, x) = PRF(K, x).$$
- *(Pseudorandomness) For all* $\lambda \in \mathbb{N}$ *and for all* $i \in \mathcal{X}$ *the following distributions are computationally indistinguishable*
$$\{i, K_i, PRF(K, i)\} \approx \{i, K_i, u\}$$
where $K_i \leftarrow \mathsf{Puncture}(K, i)$ *and* $u \leftarrow \mathcal{Y}$ *is uniformly sampled.*

3 NIWI Proofs Secure Against Chosen Statement Attacks

We recall the notion of non-interactive witness-indistinguishable (NIWI) proof for NP from [BOV03].

Definition 5 (NIWI Proof for NP)**.** *A NIWI proof* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *for an NP-language* $\mathcal{L}$ *with relation* $\mathcal{R}$ *consists of the following efficient algorithms.*

- $\mathsf{NIWI.Prove}(1^\lambda, w, x) \to \pi$*: On input the security parameter* 1^λ*, a witness* w*, and a statement* x*, the proving algorithm returns a proof* π*.*
- $\mathsf{NIWI.Verify}(\pi, x) \to b$*: On input a proof* π*, and a statement* x*, the verification algorithm returns a bit* $b \in \{0,1\}$*.*

We define the properties of interest below, including correctness, soundness and computational witness indistinguishability.

Definition 6 (Correctness)**.** *A NIWI proof* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *is correct if for all* $\lambda \in \mathbb{N}$*, all* $x \in \mathcal{L}$*, and all* $w \in \mathcal{R}(x)$ *it holds that*
$$\Pr\left[\mathsf{NIWI.Verify}(\mathsf{NIWI.Prove}(1^\lambda, w, x), x) = 1\right] = 1.$$

Definition 7 (Soundness)**.** *A NIWI proof* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *is sound if for all* $x^* \notin \mathcal{L}$ *and all proofs* π^* *it holds that*
$$\Pr\left[\mathsf{NIWI.Verify}(\pi^*, x^*) = 1\right] = 0.$$

Definition 8 (Computational Witness Indistinguishability)**.** *A NIWI proof* $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ *is witness indistinguishable if for all* $x \in \mathcal{L}$*, and all pairs of witnesses* $w_0, w_1 \in \mathcal{R}(x)$ *it holds that the following distributions are computationally indistinguishable*
$$\mathsf{NIWI.Prove}(1^\lambda, w_0, x) \approx \mathsf{NIWI.Prove}(1^\lambda, w_1, x).$$

NIWI proofs are known to exist under a variety of assumptions, such as trapdoor permutations [BOV03], bilinear pairings [GOS06], and obfuscation and one-way permutations [BP15].

CSA-Security. In the following we present our construction of *chosen statement attack* (CSA) secure NIWI. Informally, a NIWI is CSA-secure if the witness-indistinguishability property holds, even in the presence of an oracle that extracts a witness from NIWIs for different statements. Before defining this property, we define the notion of *witness extractable* NIWI.

Definition 9 (Witness Extractability)**.** *A NIWI proof* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *is witness extractable if there exists an inefficient algorithm* $\mathsf{NIWI.Extract}$ *such that for all* $x \in \mathcal{L}$ *and all proofs* π *it holds that*

$$\mathsf{NIWI.Verify}(\pi, x) = 1 \implies w \in \mathcal{R}(x)$$

where $w \leftarrow \mathsf{NIWI.Extract}(\pi, x)$.

Note that this definition does not contradict witness indistinguishability, since the extractor is not efficient. We are now ready to define the notion of CSA-securty for NIWIs.

Definition 10 (CSA-Security)**.** *A witness extractable NIWI proof* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify}, \mathsf{NIWI.Extract})$ *is CSA-secure if there exists a negligible function* negl *such that for all PPT adversaries* $\mathcal{A}$*, all statements* x*, and all pairs of witnesses* $(w_0, w_1) \in \mathcal{R}(x)$ *it holds that*

$$\Big|\Pr\Big[1 = \mathsf{Adv}^{\mathcal{O}}(x, \mathsf{NIWI.Prove}(1^\lambda, w_0, x))\Big] - \Pr\Big[1 = \mathsf{Adv}^{\mathcal{O}}(x, \mathsf{NIWI.Prove}(1^\lambda, w_1, x))\Big]\Big| \\ \leq \mathrm{negl}(\lambda)$$

where the oracle $\mathcal{O}$ *takes as input a statement* x^* *and a proof* π^**. If* $x^* \neq x$ *and* $\mathsf{NIWI.Verify}(\pi^*, x^*) = 1$*, it returns* $w^* \leftarrow \mathsf{NIWI.Extract}(\pi^*, x^*)$.

Construction of CSA-Secure NIWI. We show that a simple combination of CCA-secure commitments and (standard) NIWIs with sub-exponential security already suffices to construct CSA-secure NIWI. More in details, on input a statement x and a witness w, the prover algorithm computes

$$c_0 \leftarrow \mathsf{Com}(x, w) \quad \text{and} \quad c_1 \leftarrow \mathsf{Com}(x, 0)$$

and furthermore it computes a (regular) NIWI proof π for the statement:

$$(x, c_0, c_1) = \{\exists (w, b) : c_b \in \mathsf{Com}(x, w) \quad \& \quad w \in \mathcal{R}(x)\}\,.$$

(Recall that $c_b \in \mathsf{Com}(x, w)$ means there is a randomness r with $c_b \leftarrow \mathsf{Com}(x, w; r)$.)

We denote by λ_{Com} the security parameter of the commitment and we set λ (the security parameter of the NIWI) so that

$$\lambda = (\lambda_{\mathsf{Com}})^C$$

for some constant $C > 1$, and we require that witness indistinguishability holds against all attackers running in time polynomial in $2^{\lambda_{\mathsf{Com}}}$ (sub-exponential security).

By the soundness of the NIWI, it is clear that the above construction is witness extractable, since an inefficient extractor can simply brute-force both commitments and recover a valid witness from at least one of them. The following theorem shows that this NIWI is also CSA-secure.

Theorem 3.1. *Let* $(\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *be a sub-exponentially secure NIWI proof and let* Com *be a CCA-secure commitment scheme. Then the construction as described above is a CSA-secure NIWI.*

Proof. Let us fix the bit of the experiment to 0, in which case the view of the adversary includes the variables

$$\mathsf{Com}(x, w_0) \quad \text{and} \quad \mathsf{Com}(x, 0) \quad \text{and} \quad \mathsf{NIWI.Prove}(1^\lambda, (0, w_0), (x, c_0, c_1))$$

along with the queries to the oracle $\mathcal{O}$ as defined in Definition 10. We then argue that the following distributions ensembles are computationally indistinguishable, even in the presence of such an oracle:

$$\begin{aligned}
&\left\{\mathsf{Com}(x, w_0), \mathsf{Com}(x, 0), \mathsf{NIWI.Prove}(1^\lambda, (0, w_0), (x, c_0, c_1))\right\} \\
&\approx \left\{\mathsf{Com}(x, w_0), \mathsf{Com}(x, w_1), \mathsf{NIWI.Prove}(1^\lambda, (0, w_0), (x, c_0, c_1))\right\} \\
&\approx \left\{\mathsf{Com}(x, w_0), \mathsf{Com}(x, w_1), \mathsf{NIWI.Prove}(1^\lambda, (1, w_1), (x, c_0, c_1))\right\} \\
&\approx \left\{\mathsf{Com}(x, w_1), \mathsf{Com}(x, w_1), \mathsf{NIWI.Prove}(1^\lambda, (1, w_1), (x, c_0, c_1))\right\} \\
&\approx \left\{\mathsf{Com}(x, w_1), \mathsf{Com}(x, w_1), \mathsf{NIWI.Prove}(1^\lambda, (0, w_1), (x, c_0, c_1))\right\} \\
&\approx \left\{\mathsf{Com}(x, w_1), \mathsf{Com}(x, 0), \mathsf{NIWI.Prove}(1^\lambda, (0, w_1), (x, c_0, c_1))\right\}
\end{aligned}$$

which suffices to conclude the proof, since the last distribution is precisely identical to the experiment with the bit fixed to 1. We only discuss indistinguishability for the first two hybrid distributions, and the further ones follow along the same lines. For the first hybrid, note that the only difference is that c_1 is computed as a commitment to 0 or a commitment to w_1. Note that the NIWI is computed using the random coins of c_0, and therefore indistinguishability follows by the CCA-security of the commitment: The reduction can simulate the witness extraction oracle using the oracle provided by the security of the commitment scheme. Note that the witness extraction oracle always returns $\perp$ if the statement equals x, which in particular means that the tag of the commitment must be different, in order for the oracle to return an answer.

As for the second hybrid, the only change here is the witness that we use in the NIWI proof. We can prove indistinguishability by appealing to the sub-exponential witness indistinguishability of the NIWI: The reduction simply simulates the extraction oracle by brute-forcing the commitments, thus running in time polynomial in $2^{\lambda_{\mathsf{Com}}}$. By our choice of parameters, the reduction runtime is still low enough to derive a contradiction against the witness indistinguishability of the NIWI proof. The indistinguishability of the remaining hybrids is argued along the same lines as the ones above. □

4 Model and Definitions

We recall the standard definitions of RBE.

4.1 Standard RBE Definitions

In more detail, a registration-based encryption has the following algorithms as defined in [GHMR18]:

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: The setup algorithm takes as input the security parameter 1^λ and outputs a common reference string crs.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm takes as input the security parameter and outputs a public key pk and a secret key sk. This algorithm should be run by any party who wishes to register.

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$: The deterministic registration algorithm takes as input a common reference string crs, public parameters pp, identity id, and public key pk and outputs new public parameters pp′. In this process Reg has read-write access to aux. This algorithm should be run by the key curator.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$: The encryption algorithm takes as input the common reference string crs, the public parameters pp, an identity id, and a message m and outputs a ciphertext ct.

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$: The deterministic update algorithm Upd takes public parameters pp and identity id as input and outputs an update u. This algorithm has read access to aux. This algorithm should also be run by the key curator.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to \mathsf{m}$: The deterministic decryption algorithm takes as input a secret key sk, an update u, and ciphertext ct and outputs a message m.

Definition 11 (Completeness, compactness, and efficiency of RBE)**.** *Consider the following game for a computationally unbounded adversary* Adv *which still has only polynomially many rounds of interaction with a challenger* Chal*.*

1. *Initialization:* Chal *sets* $\mathsf{pp} \leftarrow \bot$, $\mathsf{aux} \leftarrow \bot$, $u \leftarrow \bot$, $\mathsf{ID} \leftarrow \emptyset$, $\mathsf{id}^* \leftarrow \bot$, $t \leftarrow 0$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and send* crs *to* Adv*.*
2. *Every round* Adv *does one of these actions:*
   (a) *Registering a new (non-target) identity:* Adv *sends a new identity* $\mathsf{id} \notin \mathsf{ID}$ *and a public key* pk *to* Chal*, which registers and updates* $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$*. The challenger* Chal *also updates* $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$*.*
   (b) *Registering the target identity: If the target identity* id^* *is already chosen then do nothing. Otherwise* Adv *sends a target identity* id^* *to* Chal*. If* $\mathsf{id}^* \notin \mathsf{ID}$ *then* Chal *samples an honest public-key-secret-key pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(\lambda)$ *and registers and updates* $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}^*\}$*, and sends* pk^* *to* Adv*.*
   (c) *Encrypting to the target identity: If the target identity* id^* *is not yet chosen then do nothing. Otherwise,* Chal *sets* $t \leftarrow t + 1$ *and receives a message* m_t *from* Adv*.* Chal *sends a ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_t)$ *to* Adv*.*
   (d) *Decrypting for target identity: The adversary* Adv *sends* $i \in [t]$ *to* Chal*, which computes the plaintext* $m_i' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}_i)$*. If* $m_i' = \mathsf{GetUpd}$*, then* Chal *computes the update* $\mathsf{u} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ *and computes the plaintext* $m_i' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}_i)$*.*

   The Adv *wins if* $m_i \neq m_i'$ *at some point in the game.*

We require the following requirements to be met, where $n = |\mathsf{ID}|$*:*

- *Completeness:* Adv *wins with probability* $\leq \mathrm{negl}(\lambda)$*.*
- *Compactness of public parameters and updates:* $|\mathsf{pp}|$ *and* $|u|$ *are both at most* $\mathrm{poly}(\lambda, \log n)$*.*
- *Efficiency of Registration and Updates:* Reg *and* Upd *run both in time at most* $\mathrm{poly}(\lambda, \log n)$*.*
- *Number of Updates: The total number* Upd *calls for identity* id^* *in step 2d is* $\leq O(\log n)$*.*

The security is defined similarly using an experiment.

Definition 12 (RBE Security)**.** *Consider the following game between an interactive PPT adversary* Adv *and a challenger* Chal*:*

1. *Initialization:* Chal *sets* $\mathsf{pp} \leftarrow \bot$, $\mathsf{aux} \leftarrow \bot$, $u \leftarrow \bot$, $\mathsf{ID} \leftarrow \emptyset$, $\mathsf{id}^* \leftarrow \bot$, $t \leftarrow 0$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and send* crs *to* Adv*.*
2. *Every round* Adv *does one of these actions:*
   (a) *Registering a new (non-target) identity: The adversary* Adv *sends a new identity* $\mathsf{id} \notin \mathsf{ID}$ *and a public key* pk *to* Chal*, which registers and updates* $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$*. The challenger* Chal *also updates* $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$*.*
   (b) *Registering the target identity: If the target identity* id^* *is already chosen then do nothing. Otherwise* Adv *sends a target identity* $\mathsf{id}^* \notin \mathsf{ID}$ *to* Chal*, which samples an honest public-key-secret-key pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and registers and updates* $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}^*\}$*, and sends* pk^* *to* Adv*.*
3. *Encrypting for target identity: If no target identity* id^* *is chosen* Adv *sends* id^* *to* Chal*. Then, the challenger* Chal *samples a random bit* $b \leftarrow \{0, 1\}$ *and generates an encryption to the target identity* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, b)$ *and sends* ct *to* Adv*.*

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

We call an RBE secure if the adversary wins with probability $1/2 + \mathrm{negl}(\lambda)$.

Remark 1 (Interactive RBE)**.** *In this work, we consider also a variant of the standard RBE, as an intermediate abstraction, where the registration process requires one round of interaction between the key curator and the clients. This is modeled by allowing the key generation algorithm to have read-only access to the auxiliary information. Namely, an interactive RBE is obtained from standard RBE by replacing* $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *by* $\mathsf{KeyGen}^{\mathsf{aux}}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$.

4.2 Necessity of One Honest User

Now we show that without a CRS no RBE scheme can satisfy both Definition 11 and Definition 12 against non-uniform PPT adversary. The proof is very reminiscent of proof that there can not be collision resistant hash functions without setup.

Proposition 4.1. *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ *be an RBE scheme without CRS satisfying Definition 11. Then, there exists a non-uniform PPT adversary* $\mathcal{A}$ *that breaks the security of* Π *defined in Definition 12.*

Proof. Fix a list of n pairs of public and secret keys $(\mathsf{pk}_1, \mathsf{sk}_1), \cdots, (\mathsf{pk}_n, \mathsf{sk}_n)$ where $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Pi.\mathsf{KeyGen}(1^\lambda)$. Define $H_n(\mathsf{id}_1, \cdots, \mathsf{id}_n)$ recursively by

$$H_{i+1}(\mathsf{id}_1, \cdots, \mathsf{id}_{i+1}) := \mathsf{Reg}^{[\mathsf{aux}]}(H_i(\mathsf{id}_1, \cdots, \mathsf{id}_i), \mathsf{id}_{i+1}, \mathsf{pk}_{i+1})$$

for $i = 0, \cdots, n-1$ and $H_0 = \perp$.

By definition, we know $|\mathsf{pp}| \leq \mathrm{poly}(\lambda, \log n) \leq \lambda^{c_1} \cdot (\log n)^{c_2}$ for some constants $c_1, c_2 > 0$. Without loss of generality, assume the length of identities is $|\mathsf{id}| = \lambda$. There are $\binom{2^\lambda}{n}$ different sets of n identities. We identify each set as a list of n identities. By the well known lower bound for binomial coefficients, we have $\log \binom{2^\lambda}{n} \geq n \cdot (\lambda - \log n)$. Take $n = \lambda^{c_1}$. One can easily check that $n \cdot (\lambda - \log n) > \lambda^{c_1} \cdot (\log n)^{c_2}$ for sufficiently large λ.

This implies there exist $(\mathsf{id}_1, \cdots, \mathsf{id}_n)$ and $(\mathsf{id}'_1, \cdots, \mathsf{id}'_n)$ where there exists $i \in [n]$ such that $\mathsf{id}'_i \neq \mathsf{id}_j$ for every $j \in [n]$ and $H_n(\mathsf{id}_1, \cdots, \mathsf{id}_n) = H_n(\mathsf{id}'_1, \cdots, \mathsf{id}'_n)$. Consider the following adversary Adv given the two lists of identities, the list of keys and the index i as advice.

1. Send $(\mathsf{id}_j, \mathsf{pk}_j)$ to the challenger Chal to register for $j = 1, \cdots, n$.
2. Locally register $((\mathsf{id}'_j, \mathsf{pk}_j))$ for $j = 1, \cdots, n$.
3. Ask Chal to encrypt to id'_i.
4. Locally compute update u'_i for id'_i.
5. Use $(\mathsf{sk}_i, \mathsf{u}'_i)$ to decrypt and output the bit.

The adversary Adv is clearly efficient. The attack is legitimate because id'_i has not been registered by the challenger. By construction, registering $(\mathsf{id}_j, \mathsf{pk}_j)$ for $j = 1, \cdots, n$ and registering $(\mathsf{id}'_j, \mathsf{pk}_j)$ for $j = 1, \cdots, n$ produces the exact same public parameter pp. Then, decryption will succeed with overwhelming probability, due to completeness. □

5 RBE in the Plain Model

In this section, we show despite the impossibility results how to construct RBE in the plain model.

5.1 One Honest User Security

In the following, we define a variant of registration-based encryption that only has any security if one of the users is honest. We believe this is still meaningful. We mark the difference from the previous definition in grey.

Definition 13 (One Honest User Security)**.** *Consider the following game between an interactive PPT adversary* Adv *and a challenger* Chal*:*

1. *Initialization:* Chal *sets* $\mathsf{pp} \leftarrow \bot$, $\mathsf{aux} \leftarrow \bot$, $u \leftarrow \bot$, $\mathsf{ID} \leftarrow \emptyset$, $\mathsf{HID} \leftarrow \emptyset$, $\mathsf{id}^* \leftarrow \bot$, $t \leftarrow 0$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and send* crs *to* Adv.
2. *Every round* Adv *does one of these actions:*
   (a) *Registering a new (non-target) identity: The adversary* Adv *sends a new identity* $\mathsf{id} \notin \mathsf{ID}$ *and public key* pk *to* Chal*, which updates the public parameter* $\mathsf{pp} = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and updates* $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$.
   (b) *Registering an honest identity: The adversary* Adv *sends a new identity* $\mathsf{id} \notin \mathsf{ID}$ *then* Chal *samples an honest public-key-secret-key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. *Then, the challenger* Chal *updates* $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$, $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$ *and* $\mathsf{HID} \leftarrow \mathsf{HID} \cup \{\mathsf{id}\}$*, and then sends* pk *to* Adv.
3. *Encrypting for target identity:* Adv *sends a target identity* $\mathsf{id}^* \in \mathsf{HID}$ *or* $\mathsf{id}^* \notin \mathsf{ID}$ *to* Chal*, which samples a random bit* $b \leftarrow \{0,1\}$ *and computes the encryption* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, b)$ *with respect to* id^* *and send* ct *to* Adv
4. Adv *outputs a bit* b' *and wins the game if* $b = b'$ *and* $\mathsf{HID} \neq \emptyset$.

We call an RBE one honest user secure if the adversary wins with probability $1/2 + \mathrm{negl}(\lambda)$.

All that really changes is that we keep track of the honestly registered identities. We track those in HID and we require that $\mathsf{HID} \neq \emptyset$ for the adversary to win. The major differences are highlighted.

Interactive Registration Based Encryption. We start by constructing an interactive RBE and showing that the construction satisfies the above one honest user security. Please see Remark 1 for the syntax of an interactive RBE.

Construction 5.1 (Interactive Registration Based Encryption)**.** *Let* $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$ *be a CSA-secure NIWI,* $\mathsf{RRRBE} = (\mathsf{Setup}, \mathsf{SampleRand}, \mathsf{ReRand}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be a re-randomizable RBE scheme, and* f *be an one-way permutation defined on* $\mathcal{X} = \{0,1\}^\lambda$*. We construct the algorithms of an interactive RBE. (In the following, part of* aux *is identified as the auxiliary information* $\mathsf{aux}_{\mathsf{RRRBE}}$ *for the underlying re-randomizable RBE* RRRBE*.)*

- $\mathsf{KeyGen}^{\mathsf{aux}}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: First compute* $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{RRRBE.KeyGen}(1^\lambda)$*. If* $\mathsf{aux} = \bot$*, which means no user has been registered yet, compute* $\mathsf{crs}_0 = \mathsf{RRRBE.Setup}(1^\lambda, 0; r_0)$ *with fixed randomness* $r_0 = 0$*. Otherwise, read from* aux *a list* $(\mathsf{crs}_0, \cdots, \mathsf{crs}_i)$ *and a list* $(y_1, \cdots, y_i)$*. (Here the index* i *is the number of users already registered. In the previous case where* $\mathsf{aux} = \bot$*, we let* $i = 0$*.) Then do the following.*
  1. *Sample* $r_{i+1} \leftarrow \mathsf{RRRBE.SampleRand}(1^\lambda)$ *and compute* $\mathsf{crs}_{i+1} = \mathsf{RRRBE.ReRand}(\mathsf{crs}_i; r_{i+1})$.
  2. *Sample* $x_{i+1} \leftarrow \mathcal{X}$ *and compute* $y_{i+1} = f(x_{i+1})$.
  3. *Compute* $\pi = \mathsf{NIWI.Prove}(1^{\lambda_{\mathrm{NIWI}}}, r_{i+1}, \mathsf{stm})$ *where* $\lambda_{\mathrm{NIWI}} = \lambda^C$*, for some constant* $C < 1$*,* $\mathsf{stm} = ((\mathsf{crs}_0, \cdots, \mathsf{crs}_{i+1}), (y_1, \cdots, y_{i+1}))$ *and* $\mathcal{R}(\mathsf{stm})$ *is defined to be:*
     - r *such that* $\mathsf{crs}_{i+1} = \mathsf{RRRBE.ReRand}(\mathsf{crs}_i; r)$*, or*
     - x *such that* $y_{i+1} = f(x)$*, and for all* $j \in [i]$
       - r_j *such that* $\mathsf{crs}_j = \mathsf{RRRBE.ReRand}(\mathsf{crs}_{j-1}; r_j)$*, or*
       - x_j *such that* $y_j = f(x_j)$.
  4. *Output* $(\mathsf{pk} = (\mathsf{crs}_{i+1}, y_{i+1}, \pi, \mathsf{pk}'), \mathsf{sk} = \mathsf{sk}')$.

- $\mathsf{Reg}^{[\mathsf{aux}]}(1^\lambda, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$*: If* $\mathsf{aux} = \bot$*, which means no user has been registered yet, compute* $\mathsf{crs}_0 = \mathsf{RRRBE.Setup}(1^\lambda, 0; r_0)$ *with fixed randomness* $r_0 = 0$*. Add* crs_0 *to* aux*. Otherwise, read from* aux *a list* $(\mathsf{crs}_0, \cdots, \mathsf{crs}_i)$ *and a list* $(y_1, \cdots, y_i)$*, where* i *is the number of registered users. Parse* $\mathsf{pk} = (\mathsf{crs}_{i+1}, y_{i+1}, \pi, \mathsf{pk}')$*. Reconstruct* stm *and run* $\mathsf{NIWI.Verify}(\pi, \mathsf{stm})$*. If verification fails, halt and output nothing. Otherwise, add* $\mathsf{crs}_{i+1}, y_{i+1}$ *to* aux*. Then, perform re-registration of previous users and the registration of new user in the following way. Compute* $\mathsf{pp}_1 \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}_{i+1}, \bot, \mathsf{id}_1, \mathsf{pk}_1)$*. Then, for each* $j = 2, \cdots, i+1$*, compute* $\mathsf{pp}_j \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}_{i+1}, \mathsf{pp}_{j-1}, \mathsf{id}_j, \mathsf{pk}_j)$*. Finally, output* $\mathsf{pp}' = \mathsf{pp}_{i+1}$*.*
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$*:*
  - *Parse* pp *as* $(\mathsf{crs}', \mathsf{pp}')$*.*
  - *Compute* $\mathsf{ct}' \leftarrow \mathsf{RRRBE.Enc}(\mathsf{crs}', \mathsf{pp}', \mathsf{id}, \mathsf{m})$*.*
  - *Output* $\mathsf{ct} = (\mathsf{crs}', \mathsf{ct}')$*.*
- $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$*:*
  - *Parse* pp *as* $(\mathsf{crs}', \mathsf{pp}')$*.*
  - *Output the update* $\mathsf{u} = \mathsf{RRRBE.Upd}^{\mathsf{aux}_{\mathsf{RRRBE}}}(\mathsf{pp}', \mathsf{id})$*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to \mathsf{m}$*:*
  - *Parse* ct *as* $(\mathsf{crs}', \mathsf{ct}')$*.*
  - *Output* $\mathsf{m} = \mathsf{RRRBE.Dec}(\mathsf{crs}', \mathsf{sk}, \mathsf{u}, \mathsf{ct}')$*.*

Correctness, compactness, and efficiency directly follow from the underlying re-randomizable RBE, one-way permutation and NIWI. In the following, we show that the construction is secure.

Proposition 5.2. *The RBE of Construction 5.1 is secure as defined in Definition 13.*

Proof. The proof proceeds by defining a series of hybrid experiments, where we modify the challenger's behaviour. We define the following series of hybrids.

- Hyb_0: This is the original experiment.
- Hyb_1: Let i be the index of the first registered honest identity. Extract the witnesses from the NIWIs from all identities previous to i. Call these witnesses $w_1, \ldots, w_{i-1}$. This can be done because of witness extractability of the NIWI.
- Hyb_2: Introduce a condition that if the extracted witnesses $w_1, \ldots, w_{i-1}$ do not for each $j \in [i-1]$ contain
  - an r_j such that $\mathsf{crs}_j = \mathsf{RRRBE.ReRand}(\mathsf{crs}_{j-1}; r_j)$, or
  - an x_j such that $y_j = f(x_j)$

  the adversary wins automatically. This condition does not happen by the soundness of the NIWI.
- Hyb_3: In this hybrid, we change how the challenger proves the NIWI for the first honestly registered use π_i. It uses the extracted witnesses $w_1, \ldots, w_{i-1}$ and input to the one-way permutation x_i to switch to the second branch. By CSA-Security this change stays unnoticed with all but negligible probability.
- Hyb_4: In this hybrid, we change how the crs_i is computed. Instead of re-randomizing crs_{i-1} we generate a fresh $\mathsf{crs}_i \leftarrow \mathsf{RRRBE.Setup}(1^\lambda, i)$. An adversary can not detect this change because a freshly generated crs is identically distributed to a re-randomized one as it fulfills Definition 17.
- Hyb_5: Extract the witnesses from the NIWIs from all identities after i. Call these witnesses $w_{i+1}, \ldots, w_m$, where m is the number of registered parties at the end.

- Hyb_6: In this hybrid, we introduce a condition that if the extracted witnesses $w_{i+1}, \ldots, w_m$ contain x_i, the input to the one-way permutation of the first honest user, the adversary wins automatically. This condition does not happen with all but negligible probability by the one-wayness of f.
- Hyb_7: In this hybrid, we introduce a condition that if the extracted witnesses $w_{i+1}, \ldots, w_m$ contain r_i, randomness with the property that if crs_i matches the re-randomized crs $\mathsf{RRRBE.ReRand}(\mathsf{crs}_{i-1}; r_i)$, the adversary wins automatically. This condition does not happen with all but negligible probability by the one-wayness of the re-randomization of the RBE.
- Hyb_8: In this hybrid, we introduce a condition, that checks, whether $w_{i+1}, \ldots, w_m$ contain randomnesses $r_{i+1}, \ldots, r_m$ such that $\mathsf{crs}_j = \mathsf{RRRBE.ReRand}(\mathsf{crs}_{j-1}; r_j)$ for all $j \in [i+1, m]$. This condition does happen by the soundness of the NIWI.

Hyb_8 reduces to re-randomizable RBE because every registered user after i can at most modify the CRS by re-randomizing it. □

5.2 From Interactive To Non-Interactive Registration

In the following we present a generic method to turn an RBE scheme with interactive registration (i.e., where the key generation takes as input the public parameters pp instead of the common reference string crs) into a standard RBE, with non-interactive registration. Given an RBE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ with interactive registration, our new RBE will only modify the key generation, registration, and decryption algorithms, which we describe below. Let PKEnc be the encryption algorithm for a semantically secure public-key encryption scheme.

$\mathsf{KeyGen}^*(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: Sample a regular public-key encryption pair $(\mathsf{pk}^*, \mathsf{sk}^*)$. Sample $K \leftarrow \{0,1\}^\lambda$ to be the key of a puncturable PRF. Then compute $\tilde{C} \leftarrow \mathsf{Obf}(C_{K,\mathsf{pk}^*})$ and return $\tilde{C}$ as the public key and sk^* as the secret key. The circuit C_{K,pk^*} is defined to take as input some public parameters pp and it behaves as follows:

- Evaluate $(r_0, r_1) \leftarrow \mathrm{PRF}(K, \mathsf{pp})$.
- Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; r_0)$.
- Return pk and $c \leftarrow \mathsf{PKEnc}(\mathsf{pk}^*, \mathsf{sk}; r_1)$.

$\mathsf{Reg}^{*[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$: The new registration algorithm evaluates $(\mathsf{pk}, c) \leftarrow \tilde{C}(\mathsf{pp})$ and proceeds as the old Reg, recording c as part of the auxiliary information.

$\mathsf{Dec}^*(\mathsf{sk}, u, \mathsf{ct}) \to \mathsf{m}$: We assume without loss of generality that the ciphertext c is part of the update u. Then the decryption algorithm uses sk^* to decrypt c and recover sk, which is then used as input to run the old Dec algorithm.

The correctness of the new scheme follows easily from the correctness of the old RBE. The following theorem summarizes the result of this section.

Theorem 5.3. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ *be an RBE with interactive registration. Then the construction as described above is a secure RBE.*

Proof. The proof proceeds by defining a series of hybrid experiments, where we modify the way we compute the output of the key generation algorithm KeyGen^* for all honest parties in the experiment. For the i-th party, we define the following series of hybrids.

- $\mathsf{Hyb}_{i,0}$: This is the original experiment.
- $\mathsf{Hyb}_{i,1}$: In this hybrid, we modify the way the obfuscated circuit is computed. Let pp^* be the current version of the public parameters of the scheme, we define $K\{\mathsf{pp}^*\} \leftarrow \mathsf{Puncture}(K, \mathsf{pp}^*)$ and $(r_0^*, r_1^*) \leftarrow \mathrm{PRF}(K, \mathsf{pp}^*)$. We then obfuscate the circuit $C_{K\{\mathsf{pp}^*\},\mathsf{pk}^*,r_0^*,r_1^*}$ defined as follows:
  - If $\mathsf{pp} = \mathsf{pp}^*$ set $(r_0, r_1) = (r_0^*, r_1^*)$. Else evaluate $(r_0, r_1) \leftarrow \mathrm{PRF}(K\{\mathsf{pp}^*\}, \mathsf{pp})$.

- Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; r_0)$.
- Return pk and $c \leftarrow \mathsf{PKEnc}(\mathsf{pk}^*, \mathsf{sk}; r_1)$.

It is easy to see that the two circuits are functionally equivalent, and therefore computational indistinguishability follows from the security of Obf.

- $\mathsf{Hyb}_{i,2}$: In this hybrid, we proceed as before except that we substitute (r_0^*, r_1^*) with two uniformly random values. Indistinguishability follows by the security of the puncturable pseudorandom function.
- $\mathsf{Hyb}_{i,3}$: In this hybrid, we hardwire the outputs of key generation $\mathsf{KeyGen}(\mathsf{pp}^*; r_0^*)$ and encryption $\mathsf{PKEnc}(\mathsf{pk}^*, \mathsf{sk}; r_1^*)$ in the obfuscated circuit, instead of computing them on-the-fly. Indistinguishability follows once again by appealing to the security of the obfuscation scheme.
- $\mathsf{Hyb}_{i,4}$: We substitute the hardwired c with an encryption of 0. Indistinguishability follows by the semantic security of the public-key encryption scheme.

After we have switched all distributions for all calls to the honest key generation algorithm, we can see that the security of the modified scheme follows by a simple reduction to the security of the underlying RBE. For calls to the honest key generation algorithm, the reduction hardwires the public keys sampled by the old KeyGen algorithm in the obfuscated circuit, as shown above. Since the key curator is acting honestly, the public parameters are always kept consistent across the experiment. Thus, the adversarial advantage in breaking the new scheme is bounded by that against the old scheme, up to some negligible factors lost in the above hybrid transitions. □

We remark that, because of the programming argument in the proof, we have to scale up the size of the obfuscated circuit to be able to contain the largest key. Since the size of the key potentially depends on the current number of users, this means that we need to set an upper-bound on the total number of users, that one needs to know when running the key generation algorithm.

6 CRS Re-Randomizable RBE

We construct a CRS re-randomizable RBE. We start by constructing a re-randomizable hash encryption and then compile it into a re-randomizable RBE.

6.1 Key Re-Randomizable Hash Encryption

We construct a hash encryption according to the definition of [DGHM18]. The construction is heavily inspired by the chameleon encryption from CDH by [DG17] for a p (prime)-order group $\mathbb{G}$ and detailed in proposition 6.4.

Construction 6.1. *We construct the algorithms of the hash encryption scheme.*

$\mathsf{Gen}(1^\lambda, n) \to k$ *: For each* $j \in [n]$*, sample* $\alpha_{j,0}, \alpha_{j,1} \xleftarrow{\$} \mathbb{Z}_p^*$ *uniformly at random and compute* $g_{j,0} \leftarrow g^{\alpha_{j,0}}$ *and* $g_{j,1} \leftarrow g^{\alpha_{j,1}}$.
Output $k := \left(g, \begin{pmatrix} g_{1,0} & g_{2,0} & \cdots & g_{n,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{n,1} \end{pmatrix}\right)$

$\mathsf{H}(k, x) \to h$ *:*

- *Parse* k *as above and* $x \in \{0,1\}^n$.
- *Output* $h := \prod_{j=1}^n g_{j,x_j}$.

$\mathsf{Enc}(k, (h, i, b), \mathsf{m}) \to \mathsf{ct}$ *:*

- *Parse* k *as above,* $(h, i, b) \in \mathbb{G} \times [n] \times \{0,1\}$*, and* $m \in \{0,1\}$.
- *Sample* $\rho \xleftarrow{\$} \mathbb{Z}_p$ *uniformly at random.*

- *Let* $c \leftarrow h^\rho$.
- *For every* $j \in [n] \setminus \{i\}$, *set* $c_{j,0} \leftarrow g_{j,0}^\rho$ *and* $c_{j,1} \leftarrow g_{j,1}^\rho$.
- *Let* $c_{i,0} \leftarrow \perp$ *and* $c_{i,1} \leftarrow \perp$.
- *Let* $e := \mathsf{m} \oplus GL(g_{i,b}^\rho)$.
- *Output* $\mathsf{ct} := \left(e, c, \begin{pmatrix} c_{1,0} & c_{2,0} & \cdots & c_{n,0} \\ c_{1,1} & c_{2,1} & \cdots & c_{n,1} \end{pmatrix}\right)$.

$\mathsf{Dec}(k, x, \mathsf{ct}) \to \mathsf{m}$ *:*

- *Parse* k *as above,* $x \in \{0,1\}^n$*, and* ct *as above.*
- *Output* $\mathsf{m} = e \oplus GL\left(\frac{c}{\prod_{j \in [n] \setminus \{i\}} c_{j,x_j}}\right)$.

We show that it is a hash encryption according to the definition of [DGHM18] even with a semi-honestly re-randomized key. Formally, we prove the following two properties:

Perfect Correctness For all $i \in [n]$, $x \in \{0,1\}^n$, $m \in \{0,1\}$, and $k \in \mathbb{G} \times \mathbb{G}^{2 \times n}$, we have

$$\mathsf{Dec}(k, x, \mathsf{Enc}(k, (\mathsf{H}(k, x), i, x_i), m)) = m.$$

Proof. We have $\mathsf{H}(k, x) = \prod_{j=1}^n g_{j,x_j} = h$. Then, we have

$$\begin{aligned} &\mathsf{Enc}(k, (h, i, x_i), m) \\ =&\left(e = \mathrm{GL}(g_{i,x_i}^\rho) \oplus m, c = \prod_{j=1}^n g_{j,x_j}^\rho, (c_{j,0}, c_{j,1}) = (g_{j,0}^\rho, g_{j,1}^\rho)_{j \in [n] \setminus \{i\}}\right) \\ =&ct \end{aligned}$$

and

$$\mathsf{Dec}(k, x, ct) = e \oplus \mathrm{GL}\left(\frac{\prod_{j=1}^n g_{j,x_j}^\rho}{\prod_{j \in [n] \setminus \{i\}} g_{j,x_j}^\rho}\right) = \mathrm{GL}(g_{i,x_i}^\rho) \oplus \mathrm{GL}(g_{i,x_i}^\rho) \oplus m = m.$$

□

Re-Randomizable Key We add two new algorithms to the construction: SampleRand and ReRand. SampleRand samples randomness for the re-randomization and ReRand re-randomizes the key. More specifically, the syntax of these algorithms is as follows:

$\mathsf{SampleRand}(1^\lambda, n) \to \text{rand}$: Takes as input the security parameter λ and integer n and outputs randomness rand.

$\mathsf{ReRand}(k, \text{rand}) \to k'$: Takes as input a key k and randomness rand and outputs a new key k'.

We then want the following properties to hold:

Definition 14 (Re-Randomizable Key)**.** *For all keys* k*, we have that a re-randomized key* $k' \leftarrow \mathsf{ReRand}(k, \mathsf{SampleRand}(1^\lambda, n))$ *is identically distributed to* $\mathsf{Gen}(1^\lambda, n)$.

Definition 15 (Re-Randomizing One-Wayness)**.** *For all keys* k *and* $k' \leftarrow \mathsf{Gen}(1^\lambda, n)$ *we have that any PPT adversary* $\mathsf{Adv}(k, k')$ *only has a negligible probability of computing an* r *such that* $k' = \mathsf{ReRand}(k, r)$.

Definition 16 (Security with Re-Randomized Key)**.** *A hash encryption is secure with re-randomizable key if for all PPT adversaries* Adv_1 *and* Adv_2*,* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$ *there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$ *the adversary's probability of winning the following experiment is* $1/2 + \mathrm{negl}(\lambda)$*:*

1. *Let* $k_0 \leftarrow \mathsf{Gen}(1^\lambda, n)$.
2. *Let* $(i, m_0, m_1, l, (\mathsf{rand}_j)_{j\in[l]}, st) \leftarrow \mathsf{Adv}_1(k_0)$.
3. *For* $j \in [l]$ *let* $k_j \leftarrow \mathsf{ReRand}(k_{j-1}, \mathsf{rand}_j)$.
4. *Sample* $b \xleftarrow{\$} \{0,1\}$ *uniformly at random.*
5. *Let* $ct \leftarrow \mathsf{Enc}(k_l, (\mathsf{H}(k_l, x), i, 1 - x_i), m_b)$.
6. *Let* $b' \leftarrow \mathsf{Adv}_2(st, k_l, ct)$.
7. *The adversary wins if* $b' = b$.

Extending DGHM For Construction 6.1 we define the algorithms SampleRand and ReRand as follows:

$\mathsf{SampleRand}(1^\lambda, n)$: For each $j \in [n]$, sample $\beta_{j,0}, \beta_{j,1} \xleftarrow{\$} \mathbb{Z}_p^*$ uniformly at random.

Output rand $:= \begin{pmatrix} \beta_{1,0} & \beta_{2,0} & \cdots & \beta_{n,0} \\ \beta_{1,1} & \beta_{2,1} & \cdots & \beta_{n,1} \end{pmatrix}$.

$\mathsf{ReRand}(k, \mathrm{rand})$:

- Parse k as above and rand $= (\beta_{i,j})_{i\in[n],j\in\{0,1\}}$.
- Output $k' := \left(g, \begin{pmatrix} g_{1,0}^{\beta_{1,0}} & g_{2,0}^{\beta_{2,0}} & \cdots & g_{n,0}^{\beta_{n,0}} \\ g_{1,1}^{\beta_{1,1}} & g_{2,1}^{\beta_{2,1}} & \cdots & g_{n,1}^{\beta_{n,1}} \end{pmatrix} \right)$.

Proposition 6.2 (Re-Randomizable Key)**.** *Construction 6.1 has a re-randomizable key according to Definition 14.*

Proof. For all $k \in (\mathbb{G} \setminus \{1\})^{2\times n}$, we have that

$$k' \leftarrow \mathsf{ReRand}(k, \mathsf{SampleRand}(1^\lambda, n))$$

and $\mathsf{Gen}(1^\lambda, n)$ are uniform distributions over $(\mathbb{G} \setminus \{1\})^{2\times n}$. □

Proposition 6.3. *In Construction 6.1 re-randomizion is one-way accoding to Definition 15.*

Proof. The key of the hash function has $2n$ elements but for this argument we only need to look at a single fixed one, say the position $(1, 1)$. Because the adversary Adv can depend on the key k there also exists an adversary Adv' that has non-uniform advice of the discrete logarithm of the $(1, 1)$-th element of k and can break the discrete logarithm.

$\mathsf{Adv}'(h)[\alpha]$:

- Has non-uniform advise α such that $g^\alpha = k_{1,1}$.
- Sample $k' \xleftarrow{\$} (\mathbb{G} \setminus \{1\})^{2\times n}$ uniformly at random.
- Replace $k'_{1,1}$ by h^α.
- Get $(d_{i,j})_{i\in\{0,1\},j\in[n]} \leftarrow \mathsf{Adv}_k(k')$.
- Output $d_{1,1}$.

The modified k' that Adv receives has the correct distribution as all elements are uniform over $\mathbb{G} \setminus \{1\}$, so Adv behave like it does on a real challenge with respect to k. If Adv has an output that breaks one-wayness of the re-randomization then $h^\alpha = k'_{1,1} = k_{1,1}^{d_{1,1}} = g^{\alpha d_{1,1}}$. Therefore, $g^{d_{1,1}} = h$ and Adv' wins as well. □

Proposition 6.4 (Security with Re-Randomized Key)**.** *The hash encryption of Construction 6.1 is secure with re-randomizable key according to Definition 16.*

Proof. This proof follows the security proof of the chameleon encryption of [DG17] up to the details of the re-randomized key. We prove via reduction to CDH. Given an adversary Adv that can win the security with re-randomizable key game with non-negligible advantage we construct an adversary Adv' that can win the CDH game over group $(\mathbb{G}, p, g)$ with non-negligible advantage.

$\mathsf{Adv}'(g, U, V)$:

- For $j \in [n]$:
  - Sample $\alpha_{j,0}, \alpha_{j,1} \xleftarrow{\$} \mathbb{Z}_p^*$ uniformly at random.
  - Let $g_{j,0} = g^{\alpha_{j,0}}$ and $g_{j,1} = g^{\alpha_{j,1}}$.
- Sample $x^* \xleftarrow{\$} \{0,1\}$ and $i^* \xleftarrow{\$} [n]$ uniformly at random.
- Replace $g_{i^*,1-x_i}$ by U.
- Let $k := \left(g, \begin{pmatrix} g_{1,0} & g_{2,0} & \cdots & g_{n,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{n,1} \end{pmatrix}\right)$.
- Let $(i, m_0, m_1, l, (\mathsf{rand}_j)_{j\in[l]}, st) \leftarrow \mathsf{Adv}_1(k)$ where $\mathsf{rand}_j = \begin{pmatrix} \beta_{1,0}^{(j)} & \beta_{2,0}^{(j)} & \cdots & \beta_{n,0}^{(j)} \\ \beta_{1,1}^{(j)} & \beta_{2,1}^{(j)} & \cdots & \beta_{n,1}^{(j)} \end{pmatrix}$
- Let $\beta := \begin{pmatrix} \beta_{1,0} := \prod_{i\in[l]} \beta_{1,0}^{(i)} & \beta_{2,0} := \prod_{i\in[l]} \beta_{2,0}^{(i)} & \cdots & \beta_{n,0} := \prod_{i\in[l]} \beta_{n,0}^{(i)} \\ \beta_{1,1} := \prod_{i\in[l]} \beta_{1,1}^{(i)} & \beta_{2,1} := \prod_{i\in[l]} \beta_{2,1}^{(i)} & \cdots & \beta_{n,1} := \prod_{i\in[l]} \beta_{n,1}^{(i)} \end{pmatrix}$
- Let $k' := \left(g, \begin{pmatrix} g_{1,0}^{\beta_{1,0}} & g_{2,0}^{\beta_{2,0}} & \cdots & g_{n,0}^{\beta_{n,0}} \\ g_{1,1}^{\beta_{1,1}} & g_{2,1}^{\beta_{2,1}} & \cdots & g_{n,1}^{\beta_{n,1}} \end{pmatrix}\right)$.
- If $i \neq i^*$ or $x^* \neq x_i$ output a uniformly random bit $b \xleftarrow{\$} \{0,1\}$.
- Otherwise:
  - Let $V' := V^{\beta_{i^*,1-x^*}^{-1}}$.
  - Let $ct := \left(e, c, \begin{pmatrix} c_{1,0}, c_{2,0}, \ldots, c_{n,0} \\ c_{1,1}, c_{2,1}, \ldots, c_{n,1} \end{pmatrix}\right)$ where $c := V'$, $e \xleftarrow{\$} \{0,1\}$, for all $j \in [n] \setminus \{i\}$ we have $c_{j,0} := V'^{\alpha_{j,0}\cdot\beta_{j,0}}, c_{j,1} := V'^{\alpha_{j,1}\cdot\beta_{j,1}}$
  - $b \leftarrow \mathsf{Adv}_2(st, k', ct)$.
  - Output $b \oplus e$.

Observe that the distribution of k is identical to that of Gen. This implies that the view of Adv_1 is identical to the one in the experiment. We call the event that $i = i^*$ and $x = x^*$ event E. The event E happens with probability $\frac{1}{2n}$. Conditioned on E we have that the view of Adv_2 is identically distributed to the view in the experiment where ct is an encryption of $e \oplus \mathrm{GL}(g^{u\cdot v})$, where $U = g^u$ and $V = g^v$. Now if Adv_2 correctly predicts $e \oplus \mathrm{GL}(g^{u\cdot v})$ with non-negligible probability then Adv' predicts $\mathrm{GL}(g^{u\cdot v})$ with non-negligible probability. □

6.2 Compiling to CRS Re-Randomizable RBE

We compile a key re-randomizable hash encryption into a CRS re-randomizable RBE. A CRS re-randomizable RBE is an RBE with two extra algorithms $\mathsf{SampleRand}$ that samples randomness for the re-randomization and ReRand that re-randomizes the CRS. These algorithms have the following syntax:

$\mathsf{SampleRand}(1^\lambda) \to \mathsf{rand}$: Takes as input the security parameter λ and outputs randomness rand.

$\mathsf{ReRand}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{rand}) \to \mathsf{crs}'$: Takes as input a crs and rand and outputs a new crs'.

Further, we amend the syntax for Setup slightly.

$\mathsf{Setup}(1^\lambda, i) \to \mathsf{crs}$: Takes as input the security parameter λ an index i and outputs a common reference string crs.

In addition to previous RBE properties, we need the following three properties to hold.

Definition 17 (Re-Randomizable CRS)**.** *For all* $i \in [2^\lambda]$ *and* crs *from the range of* $\mathsf{Setup}(1^\lambda, i-1)$*, we have* $\mathsf{ReRand}(\mathsf{crs}, \mathsf{SampleRand}(1^\lambda))$ *is identically distributed as* $\mathsf{Setup}(1^\lambda, i)$.

Definition 18 (Re-Randomizing One-Wayness)**.** *For all* $i \in [2^\lambda]$ *and CRS* crs *from the range of* $\mathsf{Setup}(1^\lambda, i-1)$ *and* $\mathsf{crs}' \leftarrow \mathsf{Setup}(1^\lambda, i)$ *we have that any randomized sub-exponential time adversary* $\mathsf{Adv}(\mathsf{crs}, \mathsf{crs}')$ *only has a negligible probability of computing an* r *such that* $\mathsf{crs}' = \mathsf{ReRand}(\mathsf{crs}, r)$.

Definition 19 (CRS Re-Randomizable RBE Security)**.** *Consider the following game between an interactive PPT adversary* Adv *and a challenger* Chal*:*

1. *Initialization:* Chal *sets* $\mathsf{pp} \leftarrow \bot$, $\mathsf{aux} \leftarrow \bot$, $u \leftarrow \bot$, $\mathsf{ID} \leftarrow \emptyset$, $\mathsf{id}^* \leftarrow \bot$, $t \leftarrow 0$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 0)$ *and sends* crs *to* Adv.
2. *Every round* Adv *does one of these actions:*
   (a) *Registering a new (non-target) identity:* Adv *sends a new identity* $\mathsf{id} \notin \mathsf{ID}$ *and its corresponding public key* pk *to* Chal. Chal *registers the public key* pk *on identity* id *and obtains an update* $u = \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and updated (non-target) list* $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$.
   (b) *Registering the target identity: If the target identity* id^* *is already chosen then do nothing. Otherwise,* Adv *sends a target identity* id^* *to* Chal. *If* $\mathsf{id}^* \notin \mathsf{ID}$ *then* Chal *samples an honest public key and secret key pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and update the public parameter* $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}^*\}$*, and sends* pk^* *to* Adv.
   (c) *Re-randomizing the CRS: The challenger* Chal *receives* rand *from* Adv *and re-randomizes the CRS* $\mathsf{crs} \leftarrow \mathsf{ReRand}(\mathsf{crs}, \mathsf{rand})$. *Then, the challenger* Chal *re-registers all users using the new* crs *in the following way. Let* n *be the number of users that have already been registered and* $(\mathsf{id}_i, \mathsf{pk}_i)$ *be the identity and public key pair of the* $i-$*th registered user. (This information can be read from the auxiliary information* aux*.) First, clear the auxiliary information* aux*. Then, compute* $\mathsf{pp}_1 \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \bot, \mathsf{id}_1, \mathsf{pk}_1)$ *and then for each* $i = 2, \cdots, n$*, compute* $\mathsf{pp}_i \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}_{i-1}, \mathsf{id}_i, \mathsf{pk}_i)$*. Finally, set* $\mathsf{pp} \leftarrow \mathsf{pp}_n$ *and send* crs *to the adversary* Adv.
3. *Encrypting for target identity: If no target identity* id^* *is chosen* Adv *sends* id^* *to the challenger* Chal *first. Then the challenger* Chal *samples a random bit* $b \leftarrow \{0, 1\}$*, compute its encryption to the target identity* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, b)$ *and sends* ct^* *to* Adv.
4. Adv *outputs a bit* b' *and wins the game if* $b = b'$.

We call a CRS Re-Randomizable RBE secure if the probability of any sub-exponential adversary Adv*'s winning in the above game is* $< 1/2 + \mathrm{negl}(\lambda)$.

The completeness definition of registration-based encryption can be similarly adapted to CRS re-randomizable RBE by adding a re-randomization query. Since the modification is mainly syntactic and not fundamental, we omit it here.

We now show that with minor modifications the registration-based encryption of [GHM+19] based on hash encryption is a CRS re-randomizable RBE. The only changes we need to make are specific to re-randomizing and keeping track of the number of re-randomizations.

Construction 6.5. *Let* RBE' = (Setup, KeyGen, Reg, Upd, Enc, Dec) *be the RBE of [GHM+19] and* HashEnc = (Gen, H, Enc, Dec, SampleRand, ReRand) *be the key re-randomizable hash encryption used to instantiate the hash encryption of* RBE'*. We define the CRS re-randomizable RBE* RBE *as follows:*

$\mathsf{Setup}(1^\lambda, i)$ *:*

- *Let* $k \leftarrow \mathsf{HashEnc.Gen}(1^\lambda, n)$ *for some* n *as determined by the RBE construction in [GHM+19].*
- *Output* $\mathsf{crs} \leftarrow (\mathsf{crs}' := k, i)$*, where* crs' *is exactly the CRS of* RBE'.

$\mathsf{KeyGen}(\lambda)$ *: Output* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RBE}'.\mathsf{KeyGen}(\lambda)$.

$\mathsf{Reg}^{[\mathsf{aux}]}((\mathsf{crs}', t), \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *:*

- *Split* aux *into* aux' *and* aux^**, where* aux' *is the auxiliary information for* RBE'.
- *Store* $(\mathsf{id}, \mathsf{pk})$ *in* aux^*.
- *Output* $\mathsf{pp}' \leftarrow \mathsf{RBE}'.\mathsf{Reg}^{[\mathsf{aux}']}(\mathsf{crs}', \mathsf{pp}, \mathsf{id}, \mathsf{pk})$.

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id})$ *:*

- *Split* aux *into* aux' *and* aux^**, where* aux' *is the auxiliary information for* RBE'.
- *Output* $u \leftarrow \mathsf{RBE}'.\mathsf{Upd}^{\mathsf{aux}'}(\mathsf{pp}, \mathsf{id})$.

$\mathsf{Enc}((\mathsf{crs}', t), \mathsf{pp}, \mathsf{id}, b)$ *:*

- *Let* $\mathsf{ct}' \leftarrow \mathsf{RBE}'.\mathsf{Enc}(\mathsf{crs}', \mathsf{pp}, \mathsf{id}, b)$.
- *Output* (ct', t).

$\mathsf{Dec}((\mathsf{crs}', t), \mathsf{sk}, u, \mathsf{ct} = (\mathsf{ct}', t^*))$ *:*

- *If* $t^* = t$ *then output* $\mathsf{RBE}'.\mathsf{Dec}(\mathsf{crs}', \mathsf{sk}, u, \mathsf{ct}')$.
- *Else output* GetUpd.

We add the following two algorithms to the construction, in which we crucially use the fact that the CRS is exactly the key of the hash encryption and that the hash encryption is re-randomizable.

$\mathsf{SampleRand}(1^\lambda)$ *: Output* $\mathsf{rand} \leftarrow \mathsf{HashEnc.SampleRand}(1^\lambda, n)$.

$\mathsf{ReRand}((\mathsf{crs}', t), \mathrm{rand})$ *:*

- *Parse* crs' *as* k.
- *Let* $k' \leftarrow \mathsf{HashEnc.ReRand}(k, \mathrm{rand})$.
- *Output* $\mathsf{crs} = (\mathsf{crs}' = k', t + 1)$.

Correctness, compactness, and efficiency directly follows from the correctness, compactness, and efficiency of the RBE of [GHM+19].

CRS re-randomizability and one-wayness of the re-randomization follow directly from the fact that the CRS is only a key of a re-randomizeable hash encryption with these properties and a counter.

CRS re-randomizable security follows from the property of the RBE of [GHM+19] that the CRS is only used for hash encryptions and Proposition 6.4, which tells us that the hash encryption stays secure, even if the CRS is re-randomized. They only make black-box use of the security property in the reduction, therefore, the re-randomizable RBE security follows from their proof straightforwardly.

7 Lower Bound on Number of Updates

In this section, we show that $\Omega(n)$ number of updates is necessary for RBE without CRS under two mild assumptions that are satisfied by known constructions.

Assumptions.

The first assumption is about fixed update times, as in Definition 2.4 in [MQR22]. Conceptually, this assumption states that the time when a registered user requires an update is fixed and known. We capture this using a DAG, where the $i-$th user is represented by vertex i and an edge from i to j means that the $i-$th registered user needs an update immediately after the registration of the $j-$th user. Since an user can receive updates only after it has already been registered, it is clear that in such graphs edges can only go from smaller vertices to larger vertices, which we refer to as *Forward DAGs*.

Definition 20 (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 $i \leq j$.*

With the use of Forward DAGs, we state our first assumption. Conceptually, we say that an RBE scheme has fixed update times according to a forward DAG G if completeness of the scheme holds when user i receives an update at time j for every edge $(i, j) \in G$. In particular, we demand each decryption query to return a message. This is in contrast with the original completeness of RBE where a decryption query can return the special symbol GetUpd, upon seeing which an user can request decryption update. Since we are assuming a user only needs updates at times specified in the forward DAG and every user is indeed given the required update at the right time, decryption query should never output GetUpd.

Definition 21 (Completeness of RBE with fixed update times)**.** *Let G be an infinite forward DAG. For an RBE scheme and any interactive* computationally unbounded *adversary* Adv *that still has a limited* $\mathrm{poly}(\lambda)$ *round complexity, consider the game* $\mathsf{UpdTimes}^{G}_{\mathsf{Adv}}(\lambda)$ *between* Adv *and a challenger* Chal *as follows.*

1. ***Initialization.*** Chal *sets* $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{u} = \bot$, $\mathcal{D} = \emptyset$, $\mathcal{S} = \emptyset$, $t = 0$, *and* $\mathsf{crs} \leftarrow U_{\mathrm{poly}(\lambda)}$, *and sends the sampled* crs *to* Adv.

2. *Till* Adv *continues (which is at most* $\mathrm{poly}(\lambda)$ *steps), proceed as follows. At every iteration,* Adv *chooses exactly one of the actions below to perform.*

   (a) ***Registering identities.*** Adv *performs exactly one out of Step 2(a)i and Step 2(a)ii below, but regardless of this choice,* Chal *will continue to send the updates as described next.*

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

      ii. ***Registering the target uncorrupted identity.*** *This step is allowed only if* $\mathsf{id}^* = \bot$. *In that case,* Adv *sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to* Chal. Chal *then samples* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$, *runs* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} := \mathcal{D} \cup \{\mathsf{id}^*\}$, *and sends* pk^* *to* Adv.

      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* i*th registered identity, and let the identity registered in either of Step 2(a)i Step or 2(a)ii be the* j*th identity. 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.*

   (b) ***Encrypting for the target identity.*** *This step is allowed only if* $\mathsf{id}^* \neq \bot$. *In that case,* Chal *sets* $t = t + 1$. Adv *sends* $\mathsf{m}_t \in \{0, 1\}^*$ *to* Chal *who then sets* $\mathsf{m}'_t := \mathsf{m}_t$ *and sends back a corresponding ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ *to* Adv.

   (c) ***Decryption for the target identity.*** Adv *sends* $j \in [t]$ *to* Chal *who lets* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$.

The adversary Adv *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* $\Pr[\mathsf{Adv}\ \textit{wins}] = \mathrm{negl}(\lambda)$. *In this case, we also say that the completeness holds with fixed update graph G.*

The second assumption states that encryption and decryption succeed if and only if one can recover part of the public parameters from the update witness. This is motivated by known constructions, where the update witness is the

opening of a succinct commitment and the decryption recomputes some public information depending on the index. As an example, it is useful to keep in mind the example of Merkle-tree based schemes, where the witness is just a root-to-leaf path, which in particular allows one to recompute the root of the tree.

Definition 22 (PP-recoverability)**.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be an RBE scheme with completeness probability* ρ*. We say* Π *is* pp-recoverable *if there exist two deterministic algorithms* Ext_1 *and* Ext_2 *such that for all* pp*,* crs*,* id*,* u*, and any* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)$*, there exists an index* i *such that*

$$\mathsf{Ext}_1(\mathsf{crs}, (\mathsf{id}, \mathsf{pk}), \mathsf{u}) = \mathsf{Ext}_2(\mathsf{crs}, \mathsf{pp}, i)$$

if and only if

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m})) = \mathsf{m}] \geq \rho$$

for any message m*.*

Justification for the two assumptions. We use the construction in the first RBE paper [GHMR18] as a concrete example to illustrate and justify the assumptions since follow-up works all follow the same paradigm of accumulating identity and key pairs into succinct commitments and encrypting with respect to the commitments. The construction works in the following way: KC maintains a list of Merkle trees where the leaves are the identity and key pairs and publish roots as public parameters. An update for a user is an opening for the corresponding identity and key leaf in the tree. Encryption is an obfuscated program which first checks that the identity and key provided by whoever wishes to decrypt are consistent with one of the roots in the public parameter and then outputs an encryption of the message using the provided key. To justify fixed update times assumption, note that an update is needed if and only if two trees are merged and the times when two trees are merged only depend on the number of registered users. To justify PP-recoverability, note that to verify an identity and key pair, one uses the pair and an opening to re-compute a root and check it against the published roots. If there is a valid opening for some identity and key pair that results in one of the roots, then the obfuscated program will accept the public key and use it to generate an encryption, which can be decrypted using the corresponding secret key. Some papers use other scheme than Merkle trees, such as vector commitment scheme [GKMR23]. However, since the underlying ideas are the same, the assumptions still hold.

Proof of Main Result.

We state our main theorem below.

Theorem 7.1 (Main Result)**.** *Let* Π *be an RBE scheme without CRS satisfying (1) completeness with fixed update times in Definition 21, (2) one honest user security in Definition 13, and (3) pp-recoverability in Definition 22. Then, the number of decryption updates cannot be* $o(n)$*.*

The proof takes the following main steps.

1. Define the concept of a good tuple (i, n) for an RBE scheme Π.
2. Show that if (i, n) is a good tuple then the ith identity requires a decryption update at time $n + 1$.
3. Show that there are $\Omega(n)$ many good tuples of the form (i, n) for every n. Then, by a simple counting argument, the result follows.

Definition 23 (Good Tuple)**.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be an RBE scheme without CRS. We say* (i, n)*, where* $1 \leq i \leq n$*, is a* good tuple *for* Π *if there exist*

1. $((\mathsf{pk}_1, \mathsf{sk}_1), \cdots, (\mathsf{pk}_n, \mathsf{sk}_n))$ *where* $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, and*
2. $(\mathsf{id}_1, \cdots, \mathsf{id}_n)$ *and* $(\mathsf{id}'_1, \cdots, \mathsf{id}'_n)$ *where* $\mathsf{id}'_i \notin (\mathsf{id}_1, \cdots, \mathsf{id}_n)$

satisfying $H^n_{n,\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}_1,\cdots,\mathsf{id}_n) = H^n_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}'_1,\cdots,\mathsf{id}'_n)$*, where*

$$H^{i+1}_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}_1,\cdots,\mathsf{id}_{i+1}) := \mathsf{Reg}^{[\mathsf{aux}]}(H^i_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}_1,\cdots,\mathsf{id}_i),\mathsf{id}_{i+1},\mathsf{pk}_{i+1})$$

for $i = 0,\cdots,n-1$ *and* $H^0_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n} = \bot$.

Lemma 7.2. *Let* $\Pi = (\mathsf{Setup},\mathsf{KeyGen},\mathsf{Reg},\mathsf{Upd},\mathsf{Enc},\mathsf{Dec})$ *be an RBE scheme without CRS satisfying (1) completeness with fixed update times, (2) one honest user security, and (3) pp-recoverability. Let* G *be the update graph for* Π*. If* (i,n) *is a good tuple for* Π*, then* $(i,n+1)$ *is an edge in* G*.*

Proof. For contradiction, assume $(i,n+1)$ is not an edge in G. We will show that there exists a non-uniform adversary breaking the security of Π.

First of all, since (i,n) is a good tuple for Π, the following advice must exist.

Advice:

- i,n.
- $(\mathsf{pk}_1,\mathsf{sk}_1),\cdots,(\mathsf{pk}_n,\mathsf{sk}_n))$ where each $(\mathsf{pk}_j,\mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- $(\mathsf{id}_1,\cdots,\mathsf{id}_n)$ and $(\mathsf{id}'_1,\cdots,\mathsf{id}'_n)$ satisfying $id'_i \notin (\mathsf{id}_1,\cdots,\mathsf{id}_n)$ and
$$H^n_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}_1,\cdots,\mathsf{id}_n) = H^n_{\mathsf{pk}_1,\cdots,\mathsf{pk}_n}(\mathsf{id}'_1,\cdots,\mathsf{id}'_n).$$

Given the above advice, we construct the following non-uniform adversary.

Adversary:

1. Send each $(\mathsf{id}_j,\mathsf{pk}_j)$ for $j=1,\ldots,n$ to Chal to register. (After this step, Chal will update the public parameter to pp_n.)
2. Send id'_i to Chal to register as the honest user. Note that this is legitimate since $id'_i \notin (\mathsf{id}_1,\cdots,\mathsf{id}_n)$. (In the view of Chal, id'_i is the identity of the $(n+1)$th user id_{n+1}. After this step, Chal will update the public parameter to pp_{n+1}.)
3. Locally register $(\mathsf{id}'_j,\mathsf{pk}_j)$ for $j=1,\ldots,n$ and generate decryption update u'_i for id'_i.
4. Ask Chal to send $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}_{n+1},\mathsf{id}_{n+1}=\mathsf{id}'_i,b)$.
5. Output $\mathsf{Dec}(\mathsf{sk}_i,\mathsf{u}'_i,\mathsf{ct})$.

Clearly, the adversary is efficient. Also, the attack is legitimate since there is one honest user $\mathsf{id}_{n+1} = \mathsf{id}'_i \notin (\mathsf{id}_1,\cdots,\mathsf{id}_n)$. Next, we bound its advantage. Let u_i be the decryption update generated for id_i after registering $(\mathsf{id}_j,\mathsf{pk}_j)$ for $j=1,\ldots,n$ and recall that in the attack u'_i is the decryption update generated for id'_i after registering $(\mathsf{id}'_j,\mathsf{pk}_j)$ for $j=1,\ldots,n$.

We first show that $\mathsf{Ext}_1((\mathsf{id}_i,\mathsf{pk}_i),\mathsf{u}_i) = \mathsf{Ext}_1((\mathsf{id}'_i,\mathsf{pk}_i),\mathsf{u}'_i)$. Note that by construction, we must have $\mathsf{Ext}_1((\mathsf{id}_i,\mathsf{pk}_i),\mathsf{u}_i) = \mathsf{Ext}_2(\mathsf{pp}_n,i)$ due to completeness and pp-recoverability. Similarly, we must have $\mathsf{Ext}_1((\mathsf{id}'_i,\mathsf{pk}_i),\mathsf{u}'_i) = \mathsf{Ext}_2(\mathsf{pp}_n,i)$.

We then show that $\mathsf{Ext}_1((\mathsf{id}_i,\mathsf{pk}_i),\mathsf{u}_i) = \mathsf{Ext}_2(\mathsf{pp}_{n+1},i)$. Since $(i,n+1) \notin G$, we have

$$\Pr\big[\mathsf{Dec}(\mathsf{sk}_i,\mathsf{u}_i,\mathsf{Enc}(\mathsf{pp}_{n+1},\mathsf{id}_i,\mathsf{m})) = \mathsf{m}\big] \geq \rho$$

for any message m, by completeness. By pp-recoverability, we must have that

$$\mathsf{Ext}_1((\mathsf{id}_i,\mathsf{pk}_i),\mathsf{u}_i) = \mathsf{Ext}_2(\mathsf{pp}_{n+1},i)$$

We thus have $\mathsf{Ext}_1((\mathsf{id}'_i,\mathsf{pk}_i),\mathsf{u}'_i) = \mathsf{Ext}_2(\mathsf{pp}_{n+1},i)$. Then, by pp-recoverability, we must have

$$\Pr\big[\mathsf{Dec}(\mathsf{sk}_i,\mathsf{u}'_i,\mathsf{Enc}(\mathsf{pp}_{n+1},\mathsf{id}'_i,\mathsf{m})) = \mathsf{m}\big] \geq \rho$$

for any message m. □

In the following, let α be the length of the public parameter, β be the length of the identity and γ be the upper bound for number of decryption updates.

Proposition 7.3. *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be an RBE scheme. Let* n *be a positive integer. Then there are at least* $(1 - \frac{\log n}{\beta}) \cdot n - \frac{\alpha}{\beta}$ *good tuples of the form* (i, n).

Proof. Fix n pairs of keys $(\mathsf{pk}_1, \mathsf{sk}_1), \cdots, (\mathsf{pk}_n, \mathsf{sk}_n)$. There are $\binom{2^\beta}{n}$ distinct sets of n identities. We identify each of them as an ordered list of identities. Note that there are 2^α different public parameters.

- Suppose $\binom{2^\beta}{n} \leq 2^\alpha$. Since $0 \geq \log\left(\frac{\binom{2^\beta}{n}}{2^\alpha}\right) > (\beta - \log n) \cdot n - \alpha$, we know $(1 - \frac{\log n}{\beta}) \cdot n - \frac{\alpha}{\beta}$ is a lower bound for number of good tuples.
- Suppose $\binom{2^\beta}{n} > 2^\alpha$. Consider a public parameter pp such that $|(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})| = \max_{\mathsf{pp}'} |(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp}')|$. Since $\binom{2^\beta}{n} > 2^\alpha$, we know $|(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})| \geq 2$. Take two different lists of identities from $(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})$. There must exist $i_1 \in [n]$ where the i_1th identity of one list does not belong to the other list. This means (i_1, n) is a good tuple.
  - Suppose $\frac{\binom{2^\beta}{n}}{2^\alpha} \leq 2^\beta$. Since $\beta \geq \log\left(\frac{\binom{2^\beta}{n}}{2^\alpha}\right) \geq (\beta - \log n) \cdot n - \alpha$, which implies $1 \geq (1 - \frac{\log n}{\beta}) \cdot n - \frac{\alpha}{\beta}$, we again have $(1 - \frac{\log n}{\beta}) \cdot n - \frac{\alpha}{\beta}$ as a lower bound for number of good tuples.
  - Suppose $\frac{\binom{2^\beta}{n}}{2^\alpha} > 2^\beta$. Let m be the largest positive integer such that $\frac{\binom{2^\beta}{n}}{2^\alpha} > (2^\beta)^m$. Consider the i_1th identity of every list in $(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})$. Since there are only 2^β different values for identity and $\frac{\binom{2^\beta}{n}}{2^\alpha} > 2^\beta$, there must be two different lists of identities from $(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})$ whose i_1th identities are the same. This means there must exist $i_2 \neq i_1$ such that (i_2, n) is a good tuple. Moreover, we know there must exist a value id_{i_1} such that there are at least $\frac{\binom{2^\beta}{n}}{2^\alpha \cdot 2^\beta}$ lists of identities in $(H^n_{\mathsf{pk}_1, \cdots, \mathsf{pk}_n})^{-1}(\mathsf{pp})$ whose i_1th identities have value id_{i_1}. One can proceed similarly again and show that there are at least $m+1$ good tuples. By definition, we know $2^{(m+1)\cdot\beta} \geq \frac{\binom{2^\beta}{n}}{2^\alpha} \geq \frac{(\frac{2^\beta}{n})^n}{2^\alpha}$. Equivalently, we have $(m+1) \cdot \beta \geq (\beta - \log n) \cdot n - \alpha$. This means $(1 - \frac{\log n}{\beta}) \cdot n - \frac{\alpha}{\beta}$ is a lower bound for number of good tuples.

□

Proposition 7.4. *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Reg}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be an RBE scheme without CRS satisfying (1) completeness with fixed update times, (2) one honest user security, and (3) pp-recoverability. Let* G *be the update graph for* Π. *Let* n *be a positive integer. There exists an* $i \in [n]$ *with at least* $(1 - \frac{\log n}{\beta}) \cdot \frac{n}{2} - \frac{\alpha}{\beta}$ *out degrees in* G_n*, the graph* G *restricted to the first* n *vertices.*

Proof. For every $i \in [n-1]$, there are $(1 - \frac{\log i}{\beta}) \cdot i - \frac{\alpha}{\beta}$ good tuples of the form (j, i). This means $i + 1$ has at least $(1 - \frac{\log i}{\beta}) \cdot i - \frac{\alpha}{\beta}$ in degree. We thus know there are at least

$$\sum_{i=1}^{n-1} (1 - \frac{\log i}{\beta}) \cdot i - \frac{\alpha}{\beta} \geq \sum_{i=1}^{n-1} (1 - \frac{\log n}{\beta}) \cdot i - \frac{\alpha}{\beta} = (1 - \frac{\log n}{\beta}) \cdot \frac{n \cdot (n-1)}{2} - (n-1) \cdot \frac{\alpha}{\beta}$$

edges in G_n. Therefore, there exists an $i \in [n-1]$ with at least $(1 - \frac{\log n}{\beta}) \cdot \frac{n}{2} - \frac{\alpha}{\beta}$ out degrees in G_n. □

Proof of Theorem 7.1. It remains to find a polynomial $n = \mathrm{poly}(\lambda)$, which can depend on α, β, and show that $(1 - \frac{\log n}{\beta}) \cdot \frac{n}{2} - \frac{\alpha}{\beta} > c \cdot n$ for some $c > 0$ and sufficiently large λ.

Note that $\alpha \leq \mathrm{poly}(\lambda, \log n)$ and $\beta = \mathrm{poly}(\lambda)$. Thus, we have $\frac{\alpha}{\beta} \leq \mathrm{poly}(\lambda, \log n) \leq \lambda^{c_1} \cdot (\log n)^{c_2}$ for sufficiently large λ. Take $n = \lambda^{2c_1}$. For sufficiently large λ, we have $1 - \frac{\log n}{\beta} > \frac{1}{2}$. We thus have $(1 - \frac{\log n}{\beta}) \cdot \frac{n}{2} - \frac{\alpha}{\beta} > \frac{n}{4} - \frac{\alpha}{\beta}$.

It is also obvious that

$$\frac{1}{8} \cdot n = \frac{1}{8} \cdot \lambda^{2c_1} > \lambda^{c_1} \cdot (2c_1 \cdot \log \lambda)^{c_2} = \lambda^{c_1} \cdot (\log n)^{c_2} \geq \frac{\alpha}{\beta}$$

for sufficiently large λ. We then have $\frac{1}{4} \cdot n - \frac{\alpha}{\beta} > \frac{1}{8} \cdot n$. □

Acknowledgement

G.M. is supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). G.M. is also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972. J.D. is supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101041207, LACONIC). W.Q. is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019547).

References

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

[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, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 209–234. Springer, Heidelberg, November 2018.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 299–315. Springer, Heidelberg, August 2003.

[BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, October 2010.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001.

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

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Heidelberg, March 2018.

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 1976.

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell WF Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 417–446. Springer, 2023.

[FFM+23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. *Cryptology ePrint Archive*, 2023.

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

[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 FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[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, Part II*, volume 11443 of *LNCS*, pages 63–93. Springer, Heidelberg, April 2019.

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

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1065–1079, 2023.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J Wu. Registered attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 511–542. Springer, 2023.

[Khu21] Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 186–215. Springer, Heidelberg, October 2021.

[KK19] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 552–582. Springer, Heidelberg, August 2019.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *58th FOCS*, pages 576–587. IEEE Computer Society Press, October 2017.

[MQR22] Mohammad Mahmoody, Wei Qi, and Ahmadreza Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography*, pages 559–587, Cham, 2022. Springer Nature Switzerland.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. *Cryptology ePrint Archive*, 2015.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

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