# <span id="page-0-0"></span>**Bounded Collusion-Resistant Registered Functional Encryption for Circuits**

Yijian Zhang $^{1,3}$ , Jie Chen $^{1,2,\boxtimes}$ , Debiao He $^4$ , and Yuging Zhang $^{5,6}$ 

 $1$  Shanghai Key Laboratory of Trustworthy Computing, School of Software Engineering, East China Normal University, Shanghai, China.

s080001@e.ntu.edu.sg

 $^2$  Institute for Math & AI. Wuhan University, Wuhan, China.

 $^3$  Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong NSW, Australia.

 $4$  School of Cyber Science and Engineering, Wuhan University, Wuhan, China.

<sup>5</sup> National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, China.  $6$  School of Cyberspace Security, Hainan University, China.

**Abstract.** As an emerging primitive, *Registered Functional Encryption* (RFE) eliminates the key-escrow issue that threatens numerous works for functional encryption, by replacing the trusted authority with a transparent key curator and allowing each user to sample their decryption keys locally. In this work, we present a new black-box approach to construct RFE for all polynomial-sized circuits. It considers adaptive simulation-based security in the bounded collusion model (Gorbunov et al. - CRYPTO'12), where the security can be ensured only if there are no more than  $Q \ge 1$  corrupted users and Q is fixed at the setup phase. Unlike earlier works, we do not employ unpractical *Indistinguishability Obfuscation* (*i*O). Conversely, it can be extended to support unbounded users, which is previously only known from *i*O.

Technically, our general compiler exploits garbled circuits and a novel variant of slotted *Registered Broadcast Encryption* (RBE), namely global slotted RBE. This primitive is similar to slotted RBE, but needs *optimally compact* public parameters and ciphertext, so as to satisfy the efficiency requirement of the resulting RFE. Then we present two concrete global slotted RBE from pairings and lattices, respectively. With proposed compiler, we hence obtain two bounded collusion-resistant RFE schemes. Here, the first scheme relies on k-Lin assumption, while the second one supports unbounded users under LWE and evasive LWE assumptions.

# **1 Introduction**

*Registered Functional Encryption* (RFE) [\[FFM](#page-28-0)+23[,DP23\]](#page-27-0) has emerged as a rising public-key cryptographic primitive recently. Unlike standard *Functional Encryption* (FE) [\[BSW11\]](#page-27-1), RFE is particularly initiated to eliminate key-escrow problem that a lot of FE schemes have suffered for many years. In RFE, a common random string crs is initialized by the key curator who broadcasts crs to all users later. Then this curator is just responsible for providing registration service for each user, without holding any secret. With crs, a newly joined user can produce a pair of public key pk and secret key sk locally, then he submits a specified function  $f$  along with pk to the curator for registration. After receiving  $(f, pk)$ , the curator updates current master public key mpk and helper secret key hsk for the new user. For encryption, the data provider uses mpk to generate a ciphertext ct associated with private data x, and the user can perform decryption algorithm over ct with (sk, hsk) to learn  $f(x)$  and nothing else. During this process, it is required that (i) all registration procedures are deterministic and auditable, and (ii) mpk and hsk must be compact (i.e., polylogarithmic in the total number of user) and updates for mpk and hsk should be efficient.

Previously, lots of significant progress have been made on constructing RFE for various kinds of limited functionality. Focusing on identity-based policy, Garg et al. [\[GHMR18\]](#page-28-1) put forth the first registration-based encryption <span id="page-1-1"></span>construction, which inspired a line of subsequent works enhancing the security [\[GV20](#page-28-2)[,DKL](#page-27-2)+23] and efficiency [\[GKMR23\]](#page-28-3) of this primitive. In addition, a more generic subclass of RFE, i.e., registered attribute-based encryption (RABE), was built relying on general assumptions over bilinear groups [\[HLWW23](#page-28-4)[,ZZGQ23\]](#page-29-0), and then it was also achieved from lattice-based assumptions [\[FWW23\]](#page-28-5).

In light of these notable achievements, it is natural to ask how to construct RFE for more powerful and generic functionality, i.e., polynomial-sized circuits. Unfortunately, all existing constructions [\[FFM](#page-28-0)+23[,DP23\]](#page-27-0) are built upon the existence of*Indistinguishability Obfuscation* (*i*O). An exception is Branco et al.'s work [\[BLM](#page-27-3)+24], which proposed a generic framework based on RFE for linear function (also called linear RFE in short), but we observe the security of their result is pretty weak, only achieving selective indistinguishablility-based (IND) security against adversary with single corrupt key. In their definition, the adversary is forced to submit the challenge message and specify the corrupted user set in advance, and IND security is actually inadequate for some cases as noted in [\[BSW11\]](#page-27-1). For FE, stronger simulation-based (SIM) security is more desirable and up to now has facilitated a series of beneficial applications  $[AW17,KW17,AFNV19,JLLW23]$  $[AW17,KW17,AFNV19,JLLW23]$  $[AW17,KW17,AFNV19,JLLW23]$  $[AW17,KW17,AFNV19,JLLW23]$ . Thus, Zhu et al.  $[ZLZ^+24]$  $[ZLZ^+24]$  formalized the definition of SIM secure RFE and presented concrete realizations, whereas they also considered the selective setting and only focused on linear/quadratic function. Given all these, an open question that arises is

# *Can we construct a registered functional encryption for all polynomial-sized circuits that achieves SIM security without assistance of i*O*?*

*Bounded Collusion Model.* In this work, we will focus on bounded collusion-resistant RFE for circuits from weak assumptions, and consider stronger adaptive SIM security. Compared to plain RFE, bounded collusion-resistant RFE additionally requires that a prior-bound  $O^1$  $O^1$  of the number of corrupted users should be declared at the setup phase. The adversary cannot extract any useful information about encrypted data x (except for  $C_1(x), \ldots, C_O(x)$ ) even when he is able to adaptively query secret keys with respect to circuits  $C_1, \ldots, C_O$ .

Bounded collusion-resistant FE has been studied extensively, and a number of works gained satisfactory results from general cryptographic tools, including public-key encryption (PKE) [\[SS10\]](#page-28-8), multi-party computation (MPC) [\[GVW12,](#page-28-9)[AV19\]](#page-27-6) and FE for linear/quadratic function [\[ALS16](#page-27-7)[,AR17\]](#page-27-8). This model is firstly proposed to construct FE for circuits without *i*O, since several works [\[AJ15,](#page-27-9)[BV15\]](#page-27-10) have shown that fully collusion-resistant FE for circuits exactly implies *i*O, making itself difficult to be deduced from weak assumptions. This rule may also work on RFE, imagine that RFE can trivially simulate FE if the key curator acts as central authority by preparing sufficient secret keys for all possible functions and then distributing each to matched users.

#### **1.1 Results**

As a matter of fact, the notion of RFE should be naturally "bounded" since the user number  $L$  is fixed during initialization. A crucial point is the size of master public key and helper secret key, which should be  $poly(C, log L)$ (here, denotes the size of circuit), namely *compactness*. However, if we think of that in bounded collusion model, the overhead of all parameters could be  $poly(L)$ . Such fact contradicts to compactness, so we decide to slightly relax it by considering a lower collusion bound  $Q \ll L$  and allowing master public key and helper key of size  $poly(C, Q, log L)$ .

In this work, we manage to answer above question and conclude our contributions as follows:

<span id="page-1-0"></span> $^1$  Generally speaking, the collusion bound Q is implied by the security parameter 1 $^\lambda$ . Since [\[GVW12\]](#page-28-9) defined bounded collusionresistant FE, it has been widely accepted that  $Q$  is an integer much less than the total number of system users, which means not many users collude with adversary. In this work, our bounded collusion-resistant RFE also inherits this assumption as default.

- <span id="page-2-2"></span>**–** We propose a new black-box approach to construct bounded collusion-resilient RFE for all polynomial-sized circuits. It mainly contains two ingredients, i.e., garbled circuits and global slotted registered broadcast encryption, where the latter can be regarded as a compact variant of slotted registered broadcast encryption. Furthermore, our approach would also be useful when constructing RFE with unbounded users property, i.e., all parameters of size not scaling with L. Prior to this work, only *iO*-based works [\[FFM](#page-28-0)<sup>+</sup>23[,DP23\]](#page-27-0) are known to realize unbounded users.
- **–** With above general compiler, we obtain two bounded collusion-resistant RFE constructions, both of which are adaptively SIM secure (without malicious case). The first one is provably secure under  $k$ -Lin assumption in the standard model. Another one is secure in the random oracle model, relying on LWE and evasive LWE assumptions [\[Wee22\]](#page-29-2). The second RFE could be extended to support unbounded users. Compared to selective security, adaptive security does not require the adversary commit any challenge information, as well as the queried input to oracles.

<span id="page-2-0"></span>

Reference	Security	<b>Assumption</b>	<b>Unbounded Full</b>	
$[FFM+23,DP23]$	AD-IND	$SSB + iO$		
$[BLM^+24]$	<b>SEL-IND</b>	$q$ -type DDH	х	x
Ours	AD-SIM	$k$ -Lin	x	x
		$AD-SIM$ LWE + evasive LWE + RO		

Table 1: Comparison among existing RFE for circuits. In the column of "Security", "AD" and "SEL" denote adaptive and selective security, respectively. In the column of "Assumption", "SSB" represents somewhere statistically binding hash functions, and "RO" represents random oracle. The column "Unbounded" and "Full" denote unbounded users and full collusion-resistance.

As shown in Table [1,](#page-2-0) it is clear that our technique greatly differs from current works. Instead of unpractical *i*O, our results are based on more general assumptions, and achieve adaptive SIM security.

Prior to this work, adaptively secure RFE for circuits can also be gained from a generic framework introduced by Branco et al. [\[BLM](#page-27-3)<sup>+</sup>24]. Nevertheless, following this line, it would at least require a linear RFE with same security level. As we mentioned before, Zhu et al.  $[ZLZ^+24]$  $[ZLZ^+24]$  provided several schemes with SIM security, whereas they considered weaker selective settings. On the other hand, if post-quantum security or unbounded users are additionally required, linear RFE with comparable features ought to be ready. However, all existing linear RFE rely on pairing-based assumptions and only support a finite number of users.

#### <span id="page-2-1"></span>**1.2 Technique Overview**

As introduced in [\[HLWW23](#page-28-4)[,FFM](#page-28-0)+23], RFE can be generically derived from slotted RFE via "power-of-two" transformation. In slotted RFE, the key curator is replaced by a stateless aggregator who aggregates all public keys and functions to generates mpk and hsk's at once. In a similar sense, bounded collusion RFE can be gained from bounded collusion slotted RFE using the same method.

We adopt the notion of Q-bound L-slot RFE, i.e., slotted RFE supporting L users and against collusion attack from  $Q$  users. In  $Q$ -bound  $L$ -slot RFE for circuits, after collecting all  $\{(\mathsf{pk}_i,C_i)\}_{i\in [L]}$ , the aggregator would publish master public key mpk and helper secret keys  $\{\sf hsk_j\}_{j\in[L]}.$  Assume the adversary holds the set of secret keys sk $_{c_1},\ldots,$  sk $_{c_0}$ (where  $c_1, ..., c_0 \in [L]$ ), SIM security requires that it cannot distinguish the challenge ciphertext ct\* that is either

<span id="page-3-0"></span>normally generated from message  $x^*$ , or simulated using (mpk, {hsk $_j$ } $_{j\in[L]}$ , {sk $_{c_j}$ } $_{j\in[Q]}$ , {C $_{c_j}(x^*)$ } $_{j\in[Q]}$ ). If we additionally consider malicious case, ct\* should be simulated without  $\{sk_{c_i}\}_{i\in [Q]}$ . Here, we ignore this stringent case. For efficiency, we require mpk and hsk of size poly( $C$ ,  $Q$ ,  $log L$ ), where  $C$  denotes the circuit size.

**Roadmap.** Our technical line somewhat deviates from current RFE for limited functionality where they always start from 1-slot case and then generalize to  $L$ -slot. We will follow the roadmap:

> $1\text{-bound 1-slot RFE} \xrightarrow[\text{FWW23,ZZGQ23]}$  $1\text{-bound 1-slot RFE} \xrightarrow[\text{FWW23,ZZGQ23]}$  $1\text{-bound 1-slot RFE} \xrightarrow[\text{FWW23,ZZGQ23]}$  $1\text{-bound 1-slot RFE} \xrightarrow[\text{FWW23,ZZGQ23]}$  $1\text{-bound 1-slot RFE} \xrightarrow[\text{FWW23,ZZGQ23]}$ 1-bound L-slot RFE  $\frac{\text{Step 2}}{[\text{GVM12}]}$  $O$ -bound  $L$ -slot RFE

**Start Point: 1-Bound 1-Slot RFE.** First, we propose a new and straightforward construction for 1-bound 1-slot RFE supporting all polynomial circuits. Initially, Sahai et al. [\[SS10\]](#page-28-8) built the first 1-bound FE for circuits from standard assumptions, which was later evolved into Q-bound FE by Gorbunov et al. [\[GVW12\]](#page-28-9). Here, we also start from [\[SS10\]](#page-28-8), but stand by a new perspective. Our first observation is: the worry-free encryption in [\[SS10\]](#page-28-8) will yield a 1-bound 1-slot RFE after slight adaptions. An overview is depicted as below.

In 1-bound 1-slot RFE, only single user is going to register his circuit C. Suppose C can be translated into a bit string of length  $n$ , given public key encryption scheme PKE = (Setup, Enc, Dec) and garbled circuit algorithms (Garble, Eval, Garble) [\[Yao86,](#page-29-3)[BHR12\]](#page-27-11), the aggregator initially samples a sequence of public keys { $\widehat{pk}_w\}_{w\in [n]}$  by running algorithm PKE.Setup *n* times. Then it sets crs =  $(\{\widehat{pk}_w\}_{w\in[n]})$ . To register circuit C, the user samples public key pairs  $\{(\mathsf{pk}_w,\mathsf{sk}_w)\}_{w\in[n]}$ . He keeps sk =  $(\{\mathsf{sk}_w\}_{w\in[n]})$  as decryption key and sends  $(\mathsf{C},\{\mathsf{pk}_w\}_{w\in[n]})$  to aggregator. Thereafter, the aggregator would produce (mpk, hsk) in the following form:

$$
mpk = \begin{pmatrix} \overline{pk}_{1,0} & \cdots & \overline{pk}_{n,0} \\ \overline{pk}_{1,1} & \cdots & \overline{pk}_{n,1} \end{pmatrix} \text{ and } hsk = \perp,
$$

where for each  $w \in [n]$  and  $b \in \{0, 1\}$ , set  $\overline{pk}_{w, b} = pk_w$  when  $C[w] = b$ ; otherwise, set  $\overline{pk}_{w, b} = \widehat{pk}_w$ .

Next, to encrypt data x, let  $U(\cdot, \cdot)$  be the universal circuit such that  $U(C, x) = C(x)$  for any circuit C and data x. Then run  $(\tilde{U},\{|ab_{w,b}\}_{w\in[n],b\in\{0,1\}}) \leftarrow$  Garble $(1^\lambda,U[x])$  where  $U[x]$  is a universal circuit with  $x$  hard-wired. With mpk, the ciphertext is defined as:

$$
ct = \left(\tilde{U}, \ \left(\begin{matrix} PKE.Enc(\overline{pk}_{1,0},\text{lab}_{1,0}) \ \cdots \ PKE.Enc(\overline{pk}_{n,0},\text{lab}_{n,0}) \\ PKE.Enc(\overline{pk}_{1,1},\text{lab}_{1,1}) \ \cdots \ PKE.Enc(\overline{pk}_{n,1},\text{lab}_{n,1})\end{matrix}\right)\right).
$$

For decryption, since  $\overline{pk}_{w,b} = pk_w$  when  $C[w] = b$ , the user can recover labels  $\{lab_{w,C[w]}\}_{w \in [n]}$  by performing algorithm PKE.Dec *n* times. Finally, he obtains  $C(x) \leftarrow$  Eval( $\tilde{U}$ , {lab<sub>w.C[w]</sub>}<sub>w∈[n]</sub>). As for security, our analysis is listed as follows:

- <mark>– In corrupt case, the registered user has colluded with adversary. Then adversary obtains labels {lab<sub>w,C[w]</sub>}<sub>w∈[n]</sub>,</mark> whereas he is unable to acquire other labels which are encrypted by public keys issued from aggregator. Thus, following the security of garbled circuits, the adversary cannot learn any information about x except for  $C(x)$ ;
- **–** In honest case, the adversary has no idea about sk, so he cannot obtain any label according to the semantic security of PKE. Thus, the privacy of  $x$  is preserved.

Actually, above construction would immediately lead to 1-bound FE enduring multiple users, by rendering a trusted authority to generate all public keys  $\{\overline{pk}_{w,b}\}$  and then distributing secret key corresponding to each user's circuit. However, in the context of registration, such idea is unrealistic since the aggregator must store no long-term secret. Most importantly,  $L$  users will generate  $L$  different public keys by themselves, so our problem is how to adapt above construction to accommodate more than one user.

**Step 1: 1-Bound L-Slot RFE.** Next, we proceed to convert 1-bound 1-slot RFE into 1-bound L-slot RFE that allows L users to register their circuits  $C_1, \ldots, C_L$ . Apparently, public key encryption is insufficient to accommodate all these circuits in mpk, so our idea is to replace it with a more powerful tool, i.e., slotted registered broadcast encryption (RBE). In slotted RBE, each user will register his slot index into mpk, and ciphertext is associated with a broadcast set (that is denoted by a bit string  $S\in\{0,1\}^L$ ) and a message m. For a user indexed by  $i$ , the decryption algorithm will recover m properly only when  $S[i] = 1$ . As for security, we just need "minimal" IND security, which states that the adversary cannot distinguish the ciphertext encrypted by either  $m_0$  or  $m_1$  given public parameters. The reason why we call minimal security is that the adversary is assumed to be unable to collude with any registered user. Let sRBE = (Setup, Gen, Ver, Agg, Enc, Dec) be a slotted RBE with minimal security, we depict 1-bound L-slot RFE as follows.

First, the aggregator initializes 2n instances of sRBE and obtains a sequence of  $\{crs_{w,b}\}_{w\in[n],b\in\{0,1\}}$ . For each instance, it runs sRBE.Gen to generate  $L$  public keys  $\{\widehat{\mathsf{pk}}_{i,w,b}\}_{i\in [L], w\in [n], b\in \{0,1\}}.$  Then set

$$
\text{crs} = (\{\text{crs}_{w,b}\}_{w \in [n], b \in \{0,1\}}, \{\widehat{\text{pk}}_{i,w,b}\}_{i \in [L], w \in [n], b \in \{0,1\}}).
$$

For a user with slot index *i*, he samples  $(\mathsf{pk}_{i,w,b}, \mathsf{sk}_{i,w,b}) \leftarrow \mathsf{sRBE}.\mathsf{Gen}(\mathsf{crs}_{w,b}, i)$  for each instance. Then set public key and secret key as

$$
\mathsf{pk}_{i} = (\{\mathsf{pk}_{i,w,b}\}_{w \in [n], b \in \{0,1\}}), \ \mathsf{sk}_{i} = (\{\mathsf{sk}_{i,w,b}\}_{w \in [n], b \in \{0,1\}}).
$$

Upon receiving  $\{ \mathsf{pk}_i,\mathsf{C}_i\}_{i\in [L]},$  the aggregator will initialize broadcast sets  $S_{w,b}\ \in\ \{0,1\}^L$  for each  $w\ \in\ [n]$  and  $b \in \{0, 1\}$ , then define master public key mpk and helper secret key hsk<sub>i</sub> (for slot  $j \in [L]$ ) as follows:

$$
mpk = \begin{pmatrix} mpk_{1,0} \cdots mpk_{n,0} \\ mpk_{1,1} \cdots mpk_{n,1} \end{pmatrix} \text{ and } hsk_j = \begin{pmatrix} hsk_{j,1,0} \cdots hsk_{j,n,0} \\ hsk_{j,1,1} \cdots hsk_{j,n,1} \end{pmatrix},
$$

where for each  $w \in [n]$  and  $b \in \{0, 1\}$ , it involves two steps:

(i) for all  $i \in [L]$ , set

$$
(\overline{\mathsf{pk}}_{i,w,b}, S_{w,b}[i]) := \begin{cases} (\mathsf{pk}_{i,w,b}, 1), & \text{when } \mathsf{C}_i[w] = b, \\ (\widehat{\mathsf{pk}}_{i,w,b}, 0), & \text{otherwise.} \end{cases}
$$

(ii) run (mpk'<sub>w,b</sub>, {hsk<sub>j,w,b</sub>}<sub>j∈[L]</sub>)  $\leftarrow$  sRBE.Agg(crs<sub>w,b</sub>, {*i*,  $\overline{pk}_{i,w,b}$ }<sub>i∈[L]</sub>) and set mpk<sub>w,b</sub> = (mpk'<sub>w,b</sub>, S<sub>w,b</sub>).

The encryption algorithm works in a similar way. Briefly, we run algorithm sRBE.Enc to generate the ciphertext:

$$
ct = \left(\tilde{U}, \left(\begin{matrix} sRBE.Enc(mpk_{1,0}, lab_{1,0}) \cdots sRBE.Enc(mpk_{n,0}, lab_{n,0}) \\ sRBE.Enc(mpk_{1,1}, lab_{1,1}) \cdots sRBE.Enc(mpk_{n,1}, lab_{n,1})\end{matrix}\right)\right),
$$

where  $(\tilde{U},\{|ab_{w,b}\}_{w\in[n],b\in\{0,1\}}) \leftarrow$  Garble $(1^\lambda,U[x])$  and note that broadcast set  $\mathcal{S}_{w,b}$  has been contained in mpk $_{w,b}.$ The decryption follows algorithms sRBE.Dec and Eval. At last, the security analysis is as follows:

– In corrupt case, suppose C\* is the unique corrupted circuit, let  $b_w$  = C\*[w] and  $\overline{b}_w$  = 1 − C\*[w], then we have

$$
\tilde{U}, \{sRBE.Enc(mpk_{w,b_w}lab_{w,b_w})\}_{w \in [n]}, \{sRBE.Enc(mpk_{w,\overline{b}_w}, lab_{w,\overline{b}_w})\}_{w \in [n]}
$$
\n
$$
\approx \tilde{U}, \{sRBE.Enc(mpk_{w,b_w}, lab_{w,b_w})\}_{w \in [n]}, \{sRBE.Enc(mpk_{w,\overline{b}_w}, Random)\}_{w \in [n]}
$$
\n
$$
\approx \tilde{U}, \{sRBE.Enc(mpk_{w,b_w}, |ab_{w,b_w})\}_{w \in [n]}, \{sRBE.Enc(mpk_{w,\overline{b}_w},Random)\}_{w \in [n]}
$$

where  $\left(\widetilde{\bm{\mathsf{U}}}, \{\widetilde{\bm{\mathsf{lab}}}_{w,b_w}\}_{w\in[n]}\right) \leftarrow \widetilde{\bm{\mathsf{Garble}}}(1^\lambda,{\mathsf{C}}^*(x)).$  The first  $\approx$  follows the IND security of slotted RBE, and the second  $\approx$  follows the simulation security of garbled circuits.

<span id="page-5-1"></span>– In honest case, since the adversary has no idea about the secret key for some honest circuit C<sub>i</sub> ≠ C\*, he cannot acquire all labels  $\{\text{lab}_{w,C_i[w]}\}_{w\in[n]}$ . Thus, it is impossible to learn other information about x, including C<sub>i</sub>(x).

*Efficiency*. It seems that 1-bound L-slot RFE is almost accomplished, because slotted RBE with minimal security can be directly obtained from recent works [\[HLWW23,](#page-28-4)[FWW23](#page-28-5)[,ZZGQ23\]](#page-29-0). However, we observe above transformation has a vital drawback. Recall that the compactness of slotted RBE requires mpk and hsk of size poly(P, log L) where P denotes the size of broadcast space. Considering broadcast space is exactly  $[L]$  and broadcast set for encryption contains  $L$  bits, it is completely reasonable that slotted RBE has the following properties:



Unfortunately, applying such slotted RBE will immediately lead to mpk and hsk of size poly( $L$ ) in our resulting construction since it does not reach the compactness goal of slotted RFE, i.e.,

$$
|\text{mpk}| = \text{poly}(C, \log L), |\text{hsk}| = \text{poly}(C, \log L).
$$
  
Ideal compactness

Besides, the encryption algorithm would be extremely inefficient when  $L$  is a large number.

To address this issue, we have to severely restrict the efficiency of underlying slotted RBE. Specifically, we expect that the underlying slotted RBE provides

**–** *Optimal compactness*. It means |mpk| = poly(log ), |hsk| = poly(log ) and ct = poly(log ). Thus, our 1-bound L-slot RFE naturally enjoys mpk and hsk of size poly(C,  $log L$ ), as well as compact ciphertext.

However, it seems rather tough to carry above thought into practice, because such efficiency requirement (especially compact ciphertext) is too restrictive. As an alternative solution, we pay attention to a weaker variant of RBE, called *global slotted RBE*. Concretely, this primitive is identical to slotted RBE except that its encryption always sets  $\mathcal{S}=1^L$  as default. This is inspired by the fact that the aggregator is generally assumed to be honest and transparent (implied by common reference string model [\[HLWW23\]](#page-28-4)). Therefore, it is unnecessary to assign broadcast sets  $\{S_{w,b}\}_{w\in[n],b\in\{0,1\}}$  for each component in ct, so we can directly fixed them at  $1^L$ , which relieves us of the difficult task of designing broadcast strategy in highly compact ciphertext.

In summary, a global slotted RBE with optimal compactness will yield a 1-bound L-slot RFE for circuits through our transformation (in Section [5\)](#page-19-0). Then the next question is how to obtain a concrete global slotted RBE. We claim that this thing is not technically harder than constructing an RABE scheme. In particular, global slotted RBE can be derived from RABE by applying the generic transformation in [\[FWW23\]](#page-28-5) which was used to build flexible BE. This reflects the universality of the primitive we proposed because several RABE schemes [\[ZZGQ23,](#page-29-0)[FWW23\]](#page-28-5) from suitable assumptions have been provided. Nevertheless, we remark that this transformation seems a bit cumbersome, requiring a pair of dummy attribute and policy for functionality and thus causing extra overhead on performance. In this work, we present two global slotted RBE schemes (in Section [3](#page-11-0) and Section [4\)](#page-15-0) that do not need any dummy attribute/policy while still achieving optimal compactness and adaptive security.

As a result, we obtain two concrete 1-bound  $L$ -slot RFE for circuits that are adaptively SIM secure under  $k$ -Lin assumption and (evasive) LWE assumption, respectively<sup>[2](#page-5-0)</sup>, Comparatively, although [\[BLM](#page-27-3)<sup>+</sup>24] has given a concrete 1-bound L-slot RFE for circuits, it just achieves weaker selective IND security, relying on  $q$ -type DDH assumption.

<span id="page-5-0"></span> $2$  Our pairing-based construction has the same structure as the k-Lin distributed BE in [\[KMW23\]](#page-28-10), because both of them are based on the BE scheme in [\[GW09\]](#page-28-11). Our lattice-based construction can also be seen as a new application of witness encryption.

<span id="page-6-0"></span>**Step 2:** *Q***-Bound** *L***-Slot RFE.** Here, we adopt Gorbunov et al.'s generic transformation [\[GVW12\]](#page-28-9). In essence, it exploits a reusable dynamic MPC protocol  $[AV19]$  to upgrade 1-bound FE to Q-bound FE. This methodology is as well suitable for the conversion from 1-bound  $L$ -slot RFE to  $Q$ -bound  $L$ -slot RFE (without malicious case).

Roughly speaking, we improves 1-bound security to O-bound security by implementing  $N$  instances of 1-bound slotted RFE in parallel, where N is a system parameter dependent on Q. To resist the adversary colluding with Q users, the encryption algorithm will divide data  $x$  into  $N$  secret shares, then use these instances to encrypt each share. In the meantime, we restrict each user only register into a part of  $N$  instances. For decryption, the user first computes multiple local parts of  $C(x)$  using secret keys, then recovers the whole  $C(x)$  by aggregating these local parts. In security reduction, Q-bound security are based on the security of underlying 1-bound L-slot RFE and MPC protocol. Finally, we manage to build a Q-bound  $L$ -slot RFE for circuits (in Section [6\)](#page-23-0) which can be later transformed into a full-fledged Q-bound RFE via "power-of-two" [\[HLWW23,](#page-28-4)[FFM](#page-28-0)+23].

**Towards Unbounded Users.** As we can see, above generic construction only supports a finite number of users, due to crs of size poly $(L)$ . Even so, we point out that it can also be utilized to construct RFE supporting an arbitrary number of users, as long as the underlying global slotted RBE supports unbounded users as well. This can be done in Step 1 by removing all public keys  $\{\widehat{\mathsf{pk}}_{i,w,b}\}$  in crs and directly aggregating all public keys from users to generate mpk and hsk. In this way, crs only consists of a limited number of global slotted RBE instances, so the size of crs will naturally not scale with  $L$  if the crs of global slotted RBE does not grow with  $L$ . Thereby, we obtain a concrete RFE for circuits enjoying unbounded users property. Considering all parameters of size unavoidably growing with , our unbounded notion is a bit weaker than earlier works [\[HLWW23](#page-28-4)[,FWW23\]](#page-28-5), but this will not be an issue due to the fact that  $Q \ll L$ .

#### **1.3 Disscussion**

**Malicious Case.** The technical barrier to tackle malicious case in O-bound slotted RFE lies on the fact that the challenger cannot ensure adversary generates secret key with right randomness. Although non-interactive zeroknowledge arguments (NIZK) [\[ZZGQ23,](#page-29-0)[BLM](#page-27-3)+24] would be helpful in simulating challenge ciphertext with only public keys, the adversary can still control the generation of randomness which is essential to the upgradation from 1-bound security to *Q*-bound security. Previously, only *iO*-based solution is known to resist malicious users, and it aims at weak IND security. Therefore, we leave seeking new technology to tackle malicious case in RFE for circuits without *i*O as a future work.

**Dynamic Bounded Collusion Model.** Recently, Agrawal et al. [\[AMVY21\]](#page-27-12) and Garg et al. [\[GGLW22\]](#page-28-12) initiated the notion of *dynamic bounded collusion model*, where *Q* is given in the encryption algorithm (instead of setup algorithm) and hence enable to more flexibly select collusion bound while balancing performance. Comparatively, our RFE is static bounded collusion-resistant. At a high level, it is feasible to spread the concept of dynamic bounded collusionresistance to the registering setting, then there is no need to require all parameters of size relevant to Q. However, to our best knowledge, it seems necessary to build a dynamic bounded RFE based on the existence of static bounded RFE [\[GGL24\]](#page-28-13). Therefore, we believe this work will motivate the study of dynamic bounded collusion-resistant RFE. **Succinctness.** One may want to ask whether it is possible to achieve succinct RFE, i.e., the encryption overhead sublinear in the size of the circuit. Intuitively, we can build a succinct 1-bound slotted RFE from our 1-bound slotted RFE and a *Laconic Function Evaluation* (LFE). It is analogous to Quach et al.'s transformation [\[QWW18\]](#page-28-14) applying on non-succinct 1-bound FE. Concretely, LFE can be used to deterministically compress the large-sized registered circuit into a short digest, then succinctness is guaranteed by performing RFE encryption with respect to LFE encryption, as the overhead of LFE encryption is small.

#### <span id="page-7-1"></span>**1.4 Related Work**

We mention other works to remove the trusted authority in FE. Chandran et al. [\[CGJS15\]](#page-27-13) introduced the notion of multi-authority functional encryption (MAFE), then proposed a MAFE for arbitrary polynomial-time function based on subexponentially secure *i*O and injective one-way functions. On the other hand, Chotard et al. [\[CDG](#page-27-14)+18] formalized the notion of decentralized muliti-client functional encryption (DMCFE) and gave the first instance supporting inner-product computation, afterwards an elegant line of work [\[ABKW19](#page-26-0)[,ABG19,](#page-26-1)[ACF](#page-27-15)+20] are devoted to this filed, while all of them only focus on linear function. Furthermore, Chotard et al. [\[CDSG](#page-27-16)+20] formalized a new extension called dynamic decentralized functional encryption (DDFE) that allows multiple users to join the system dynamically and generate secret keys in a decentralized fashion. Beyond linear function, a recent work [\[ATY23\]](#page-27-17) provided the first DDFE for attributed-weighted sums that includes arithmetic branch programs. In addition, Agrawal et al. [\[AGT21\]](#page-27-18) initiated the study of multi-party functional encryption (MPFE) that unifies a wide range of FE variants, including but not limited to MAFE, DMCFE and DDFE.

# <span id="page-7-0"></span>**2 Preliminaries**

For a finite set S, we write  $s \leftarrow S$  to denote that s is picked uniformly from finite set S. Then, we use |S| to denote the size of S. Let  $\approx$  stand for two distributions being statistically indistinguishable, and  $\approx_c$  denote two distributions being computationally indistinguishable. For any  $x \in \{0,1\}^n$ , we use  $x[w]$  to denote the  $w$ -th bit of  $x$ .

#### **2.1 Prime-Order Bilinear Groups**

A generator  $G$  takes as input a security parameter 1<sup> $\lambda$ </sup> and outputs a description G :=  $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e)$ , where  $p$  is a prime,  $\mathbb{G}_1$ ,  $\mathbb{G}_2$  and  $\mathbb{G}_T$  are cyclic groups of order p, and  $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$  is a non-degenerate bilinear map. Group operations in  $\mathbb{G}_1$ ,  $\mathbb{G}_2$ ,  $\mathbb{G}_T$  and bilinear map *e* are computable in deterministic polynomial time in  $\lambda$ . Let  $g_1 \in \mathbb{G}_1$ ,  $g_2 \in \mathbb{G}_2$  and  $g_T = e(g_1, g_2) \in \mathbb{G}_T$  be the respective generators, we employ *implicit representation* of group elements: for a matrix **M** over  $\mathbb{Z}_p$ , we define  $[\mathbf{M}]_s = g_s^{\mathbf{M}}, \forall s \in \{1, 2, T\}$ , where exponentiation is carried out componentwise. Given  $[A]_1$ ,  $[B]_2$  where **A** and **B** have proper sizes, we let  $e([A]_1, [B]_2) = [AB]_T$ . We review *matrix decisional Diffie-Hellman (MDDH) assumption, which is implied by k-Lin [\[EHK](#page-28-15)<sup>+</sup>13].* 

**Assumption 1 ((k, ℓ, d)-MDDH** over  $\mathbb{G}_s$ ,  $s \in \{1, 2\}$ ) Let  $k, \ell, d \in \mathbb{N}$  with  $k < \ell$ . We say that the  $(k, \ell, d)$ -MDDH as*sumption holds in*  $\mathbb{G}_s$  *if for all efficient adversaries*  $\mathcal{A}$ *, the following advantage function is negligible in*  $\lambda$ *.* 

$$
Adv_{\mathcal{A},s,k,\ell,d}^{MDDH}(\lambda) = \left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_s, [\mathbf{SM}]_s) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_s, [\mathbf{U}]_s) = 1] \right|
$$

 $where \mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^{\lambda}), \mathbf{M} \leftarrow \mathbb{Z}_p^{k \times \ell}, \mathbf{S} \leftarrow \mathbb{Z}_p^{d \times k} \text{ and } \mathbf{U} \leftarrow \mathbb{Z}_p^{d \times \ell}.$ 

#### **2.2 Garbled Circuits**

**Algorithms.** A garbled circuit scheme [\[Yao86](#page-29-3)[,BHR12\]](#page-27-11) (with input  $x \in \{0,1\}^n$  and circuit family C) consists of two efficient algorithms as follows:

- $-$  Garble $(1^\lambda,C)\to(\tilde{\mathsf{C}},\{\mathsf{lab}_{w,b}\}_{w\in\{n],b\in\{0,1\}}).$  It takes as input security parameter  $1^\lambda$  and a circuit  $\mathsf{C}\in\mathcal{C},$  and then outputs a garbled circuit  $\tilde{\mathsf{C}}$  and labels  $\{\mathsf{lab}_{w,b}\}_{w\in [n],b\in \{0,1\}}.$
- Eval(Č, {lab<sub>w,x[w]</sub>}<sub>w∈[n]</sub>) → z. It takes as input a garbled circuit Č and a sequence of input labels {lab<sub>w,x[w]</sub>}<sub>w∈[n]</sub>, and then deterministically outputs a value z.

Without loss of generality, we assume that the size of each label lab $_{w,b}$  is  $O(\lambda)$ .

**Correctness.** For all  $\lambda$ , for any circuit C and input  $x \in \{0, 1\}^n$ , we have

 $Pr[\text{Eval}(\tilde{C}, \{\text{lab}_{w, x[w]}\}_{w \in [n]}) = C(x) \mid (\tilde{C}, \{\text{lab}_{w,b}\}_{w \in [n], b \in \{0,1\}}) \leftarrow \text{Garble}(1^{\lambda}, C)] = 1.$ 

**Security.** There exists a simulator Garble such that for any circuit C and input  $x \in \{0, 1\}^n$ , we have

$$
(\tilde{C}, \{\mathsf{lab}_{w,x[w]}\}_{w \in [n]}) \approx_c \widetilde{\mathsf{Garble}}(1^{\lambda}, C(x))
$$

where  $(\tilde{\mathsf{C}}, \{\mathsf{lab}_{w,b}\}_{w \in [n], b \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{C}).$ 

#### <span id="page-8-0"></span>**2.3 Global Slotted Registered Broadcast Encryption**

**Algorithms.** A global slotted registered broadcast encryption (global slotted RBE for short) consists of six efficient algorithms as follows:

- $-$  Setup $(1^{\lambda},1^L)\to$  crs. It takes as input the security parameter  $1^{\lambda}$ , the upper bound  $1^L$  of the number of slots, outputs a common reference string crs.
- $-$  Gen $(crs, i) \rightarrow (pk<sub>i</sub>, sk<sub>i</sub>)$ . It takes as input crs and  $i \in [L]$ , outputs a key pair  $(\mathsf{pk}_{i}, \mathsf{sk}_{i})$ .
- $-$  Ver $(crs, i, pk<sub>i</sub>) \rightarrow 0/1$ . It takes as input crs,  $i, pk<sub>i</sub>$ , outputs a bit indicating whether pk<sub>i</sub> is valid.
- $\sim$  Agg(crs, { $i, \mathrm{pk}_i\}_{i \in [L]}$ )  $\to$  (mpk, {hsk $_j\}_{j \in [L]}$ ). It takes as input crs and a series of pk $_i$  with slot index  $i$  for all  $i \in [L]$ , outputs master public key mpk and a series of helper keys hsk<sub>i</sub> for all  $j \in [L]$ .
- $-$  Enc(mpk, m)  $\rightarrow$  ct. It takes as input mpk and a message m, outputs a ciphertext ct.
- Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) → m/⊥ . It takes as input hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct, outputs m or an empty symbol ⊥.

**Completeness.** For all  $\lambda, L \in \mathbb{N}$ , and all  $i \in [L]$ , we have

$$
\Pr\left[\text{Ver(crs}, i, pk_i) = 1 | \text{crs} \leftarrow \text{Setup}(1^{\lambda}, 1^L); (pk_i, sk_i) \leftarrow \text{Gen(crs}, i) \right] = 1.
$$

**Correctness.** For all  $\lambda, L \in \mathbb{N}$ , and all  $i^* \in [L]$ , all crs  $\leftarrow$  Setup $(1^{\lambda}, 1^L)$ , all  $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Gen}(\mathsf{crs}, i^*)$ , all  $\{\mathsf{pk}_i\}_{i \in [L] \setminus \{i^*\}}$ such that Ver $(crs, i, pk) = 1$ , and all m, we have

$$
\Pr\left[\text{Dec}(\text{hsk}_{i^*},\text{sk}_{i^*},\text{ct}) = m \middle| \begin{cases} \text{(mpk, }\{\text{hsk}_j\}_{j \in [L]}) \leftarrow \text{Agg}(\text{crs},\{i,\text{pk}_i\}_{i \in [L]}) \\ \text{ct} \leftarrow \text{Enc}(\text{mpk},m) \end{cases} \right] = 1.
$$

**Optimal Compactness.** For all  $\lambda, L \in \mathbb{N}$ , and all  $i \in [L]$ , it holds that

 $|mpk| = poly(\lambda, log L)$  and  $|hsk_i| = poly(\lambda, log L)$ .

In addition, it requires  $|ct| = poly(\lambda, \log L)$ .

**Indistinguishability-Based (IND) Security.** For all  $\lambda \in \mathbb{N}$  and all efficient adversaries  $\mathcal{A}$ , the indistinguishabilitybased security requires the advantage

> I I I I I I I I I I l

$$
\begin{bmatrix}\nL \leftarrow \mathcal{A}(1^{\lambda}); \text{crs} \leftarrow \text{Setup}(1^{\lambda}, 1^{L}) \\
\begin{aligned}\n b' &= b \\
 \text{(i, pki} & h_{i\in[L]}, m_{0}^{*}, m_{1}^{*}) \leftarrow \mathcal{A}^{\text{OGen}(\cdot). \text{OCor}(\cdot)}(\text{crs}) \\
 (mpk, \{hsk_{j}\}_{j\in[L]}) \leftarrow \text{Agg}(\text{crs}, \{i, pk_{i}^{*}\}_{i\in L}) \\
 b \leftarrow \{0, 1\}; \text{ct}^{*} \leftarrow \text{Enc}(mpk, m_{b}^{*}); b' \leftarrow \mathcal{A}(\text{ct}^{*})\n\end{aligned}\n\right]
$$

is negligible in  $\lambda$ , where oracles OGen, OCor work with initial setting  $\{\mathcal{D}_i = \emptyset\}_{i \in [L]}, C = \emptyset$  as follows:

**–** OGen(*i*): run (pk, sk) ← Gen(crs, *i*), set  $\mathcal{D}_i$ [pk] = sk and return pk.

**–** OCor(*i*, pk): return  $\mathcal{D}_i$ [pk] and update  $C = C \cup \{(i, pk)\}.$ 

and for all  $i \in [L]$ , we require  $\mathcal{D}_i[\mathsf{pk}_i^*] \neq \perp$  and  $(i, \mathsf{pk}_i^*) \notin C$ .

Indeed, global slotted RBE can be seen as a plain slotted RBE which always set broadcast set as  $1^L$  and achieves the minimal security, i.e., IND security only under honest case.

#### <span id="page-9-1"></span>**2.4 Q-Bound Slotted Registered Functional Encryption**

Algorithms. A  $Q$ -bound slotted registered functional encryption ( $Q$ -bound slotted RFE for short) for circuit family  $C: X \rightarrow Z$  consists of six efficient algorithms as follows:

- $-$  Setup $(1^{\lambda},1^L,1^Q,\mathcal{C})\to$  crs. It takes as input the security parameter  $1^{\lambda}$ , upper bound  $1^L$  of the number of slots, collusion bound  $1^Q$  and circuit family  $\mathcal C$ , outputs a common reference string crs.
- Gen(crs, *i*) → (pk<sub>i</sub>, sk<sub>i</sub>). It takes as input crs and slot index  $i \in [L]$ , outputs a key pair (pk<sub>i</sub>, sk<sub>i</sub>).
- $-$  Ver $(crs, i, pk<sub>i</sub>) \rightarrow 0/1$ . It takes as input crs,  $i, pk<sub>i</sub>$ , outputs a bit indicating whether pk<sub>i</sub> is valid.
- $-$  Agg(crs, {pk<sub>i</sub>, C<sub>i</sub>}<sub>i∈[L]</sub>)  $\to$  (mpk, {hsk<sub>j</sub>}<sub>j∈[L]</sub>). It takes as input crs and a series of pk<sub>i</sub> with C<sub>i</sub> ∈ C for all  $i$  ∈ [L], outputs master public key mpk and a series of helper keys hsk<sub>i</sub> for all  $j \in [L]$ . This algorithm is deterministic.
- **–** Enc(mpk,  $x$ ) → ct. It takes as input mpk,  $x \in X$ , outputs a ciphertext ct.
- Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct)  $\to$  z/ $\bot$  . It takes as input hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct, outputs z ∈ Z or an empty symbol ⊥.

**Completeness.** For all  $\lambda, L \in \mathbb{N}$ , all  $Q \ll L$  and all  $C$ , and all  $i \in [L]$ , we have

$$
\Pr\left[\text{Ver(crs}, i, pk_i) = 1 | \text{crs} \leftarrow \text{Setup}(1^{\lambda}, 1^L, 1^Q, C); (pk_i, sk_i) \leftarrow \text{Gen(crs}, i) \right] = 1.
$$

**Correctness.** For all  $\lambda, L \in \mathbb{N}$ , all  $Q \ll L$  and all C, and all  $i^* \in [L]$ , all crs  $\leftarrow$  Setup $(1^{\lambda}, 1^L, 1^Q, C)$ , all  $(\text{pk}_{i^*}, \text{sk}_{i^*}) \leftarrow$ Gen(crs, *i*<sup>\*</sup>), all  $\{pk_i\}_{i\in[L]\setminus\{i^*\}}$  such that Ver(crs, *i*,  $pk_i$ ) = 1, all  $x \in X$  and  $C_1, \ldots, C_L \in C$ , we have

$$
\Pr\left[\text{Dec}(\text{hsk}_{i^*},\text{sk}_{i^*},\text{ct}) = C_{i^*}(x) \middle| \begin{aligned} &(\text{mpk}, \{\text{hsk}_j\}_{j \in [L]}) \leftarrow \text{Agg}(\text{crs}, \{\text{pk}_i, C_i\}_{i \in [L]}) \\ &\text{ct} \leftarrow \text{Enc}(\text{mpk},x) \end{aligned} \right] = 1.
$$

**Compactness.** For all  $\lambda, L \in \mathbb{N}$ , all  $0 \ll L$  and all  $C$ , and all  $i \in [L]$ , it holds that

$$
|mpk| = poly(\lambda, C, Q, log L)
$$
 and  $|hsk_i| = poly(\lambda, C, Q, log L)$ .

**Simulation-Based (SIM) Security.** For all  $\lambda \in \mathbb{N}$  and all efficient adversaries  $\mathcal{A}$ , the adaptive simulation-based security requires that there exists simulator  $\widetilde{\mathsf{Enc}}$  such that:

$$
\begin{vmatrix} (L,Q) \leftarrow \mathcal{A}(1^{\lambda}); \text{crs} \leftarrow \text{Setup}(1^{\lambda},1^L,1^Q,C) \\ (\{pk_i^*,C_i^*\}_{i\in[L]},x^*) \leftarrow \mathcal{A}^{OGen(\cdot),OCor(\cdot)}(\text{crs}) \\ (\text{mpk},\{hsk_j\}_{j\in[L]}) \leftarrow \text{Agg}(\text{crs},\{pk_i^*,C_i^*\}_{i\in L}) \\ ct^* \leftarrow \text{Enc}(\text{mpk},x^*); \alpha \leftarrow \mathcal{A}(\text{ct}^*) \end{vmatrix} \approx_c \begin{vmatrix} (L,Q) \leftarrow \mathcal{A}(1^{\lambda}); \text{crs} \leftarrow \text{Setup}(1^{\lambda},1^L,1^Q,C) \\ (\{pk_i^*,C_i^*\}_{i\in[L]},x^*) \leftarrow \mathcal{A}^{OGen(\cdot),OCor(\cdot)}(\text{crs}) \\ (\text{mpk},\{hsk_j\}_{j\in[L]}) \leftarrow \text{Agg}(\text{crs},\{pk_i^*,C_i^*\}_{i\in L}) \\ \text{ct}^* \leftarrow \text{Enc}(\text{mpk},\mathcal{K}); \alpha \leftarrow \mathcal{A}(\widetilde{\text{ct}}^*) \end{vmatrix}
$$

where oracles OGen, OCor work with initial setting  $\{\mathcal{D}_i = \emptyset\}_{i \in [L]}, C = \emptyset$  and  $\mathcal{K} = \emptyset$  as follows:

- **–** OGen(*i*): run (pk, sk) ← Gen(crs, *i*), set  $\mathcal{D}_i$ [pk] = sk and return pk.
- **–** OCor(*i*, pk): return  $\mathcal{D}_i$ [pk] and update  $C = C \cup \{(i, pk)\}.$

Here, We consider the notion of  $Q$ -bound SIM security without malicious case. More concretely, it requires (i)  $\mathcal{D}_i[\mathsf{pk}_i^*] \neq \perp \text{ for all } i \in [L];$  (ii) for each  $(i, \mathsf{pk}_i^*) \in C$  where  $|C| \leq Q^3$  $|C| \leq Q^3$ , set  $\mathcal{K} = \mathcal{K} \cup \{(i, \mathsf{C}_i^*, \mathsf{C}_i^*(x^*), \mathcal{D}_i[\mathsf{pk}_i^*])\}.$ 

<span id="page-9-0"></span> $3$  Here, we consider the bounded collusion model in a weak sense, i.e., the number of corruption queries is restricted. Nevertheless, our 1-bound RFE construction is still secure, even allowing arbitrary polynomial number of corruption queries and the existence of malicious user.

#### <span id="page-10-0"></span>**2.5 Q-Bound Registered Functional Encryption**

**Algorithms.** A Q-bound registered functional encryption (Q-bound RFE for short) for circuit family  $C: X \rightarrow Z$ consists of six efficient algorithms as follows:

- $-$  Setup $(1^{\lambda},1^L,1^Q,C) \to \textsf{crs}.$  It takes as input the security parameter  $1^{\lambda},$  the maximum number of users  $1^L,$ collusion bound  $1^Q$  and circuit family  $\mathcal C$ , outputs a common reference string crs.
- **–** Gen(crs, aux) → (pk, sk). It takes as input crs and state aux, outputs key pair (pk, sk).
- $-$  Reg(crs, aux, pk, C)  $\to$  (mpk, aux'). It takes as input crs, aux, pk along with C ∈ C, outputs master public key mpk and updated state aux′ .
- **–** Upd(crs, aux, pk) → hsk. It takes as input crs, aux, pk, outputs a helper key hsk.
- **–** Enc(mpk,  $x$ ) → ct. It takes as input mpk,  $x \in X$ , outputs a ciphertext ct.
- **–** Dec(hsk, sk, ct)  $\rightarrow$   $z/\perp$  /getupd. It takes as input hsk, sk, ct, outputs  $z \in Z$  or an empty symbol  $\perp$  to indicate a decryption failure, or a symbol getupd to indicate the need of an updated helper key.

**Correctness.** For all stateful adversary  $\mathcal{A}$ , the following advantage function is negligible in  $\lambda$ :

$$
\Pr[b=1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda,1^L,1^Q,\mathcal{C}); b=0; \mathcal{A}^{\mathsf{ORegNT}(\cdot,\cdot),\mathsf{ORegT}(\cdot),\mathsf{OEnc}(\cdot,\cdot),\mathsf{ODec}(\cdot)}(\mathsf{crs})]
$$

where the oracles work as follows with initial setting aux =  $\bot$ ,  $\mathcal{E} = \emptyset$ ,  $\mathcal{R} = \emptyset$  and  $t = \bot$ :

- **–** ORegNT(pk, C): run (mpk, aux′ ) ← Reg(crs, aux, pk, C), update aux = aux′ , append (mpk, aux) to R and return  $(|\mathcal{R}|,$  mpk, aux);
- **–** ORegT(C ∗ ): run (pk<sup>∗</sup> , sk<sup>∗</sup> ) ← Gen(crs, aux) , (mpk, aux′ ) ← Reg(crs, aux, pk<sup>∗</sup> , C ∗ ), update aux = aux′ , compute hsk $^* \leftarrow$  Upd(crs, aux, pk $^*$ ), append (mpk, aux) to  $\mathcal{R}$ , return ( $t = |\mathcal{R}|$ , mpk, aux, pk $^*$ , sk $^*$ , hsk $^*$ );
- $\mathsf{I}$  OEnc(*i*, *x*): let  $\mathcal{R}[i]$  = (mpk, ·) and run ct ← Enc(mpk, *x*), append (*x*, ct) to  $\mathcal{E}$  and return ( $|\mathcal{E}|$ , ct);
- **–** ODec(j): let  $E[j] = (x_j, ct_j)$ , compute  $z_j$  ← Dec(hsk<sup>\*</sup>, sk<sup>\*</sup>, ct<sub>j</sub>). If  $z_j$  = getupd, run hsk<sup>\*</sup> ← Upd(crs, aux, pk<sup>\*</sup>) and recompute  $z_j \leftarrow \text{Dec}(\text{hsk}^*, \text{sk}^*, \text{ct}_j)$ . Set  $b = 1$  when  $z_j \neq C^*(x_j)$ .

with the following restrictions:

- **–** there exists one query to ORegT;
- **–** for query  $(i, x)$  to OEnc, it holds that  $t \geq i$ ,  $\mathcal{R}[i] \neq \bot$ ;
- **–** for query (*j*) to ODec, it holds that  $\mathcal{E}[j] \neq \bot$ .

**Compactness and Update Efficiency.** For all  $\lambda, L \in \mathbb{N}$ , all  $Q \ll L$  and all C, it holds that

$$
|mpk| = poly(\lambda, C, Q, log L)
$$
 and  $|hsk| = poly(\lambda, C, Q, log L)$ .

Furthermore, the number of invocations of Upd in ODec is at most  $O(\log |R|)$  and each invocation costs poly(log  $|R|$ ) time.

**Simulation-Based (SIM) Security.** For all  $\lambda \in \mathbb{N}$  and all efficient adversaries  $\mathcal{A}$ , the adaptive simulation-based security requires that there exists simulator  $\widetilde{\mathsf{Enc}}$  such that:

$$
\begin{vmatrix} crs &\leftarrow Setup(1^{\lambda}, 1^L, 1^Q, C) \\ x^* &\leftarrow \mathcal{A}^{ORegHK(\cdot),OCorHK(\cdot)}(crs) \\ ct^* &\leftarrow Enc(mpk, x^*) \\ \mathcal{A}^{OCorHK(\cdot)}(ct^*); \alpha &\leftarrow \mathcal{A}(ct^*) \end{vmatrix} \approx_c \begin{vmatrix} crs &\leftarrow Setup(1^{\lambda}, 1^L, 1^Q, C) \\ x^* &\leftarrow \mathcal{A}^{ORegHK(\cdot),OCorHK(\cdot)}(crs) \\ \widetilde{ct}^* &\leftarrow \widetilde{Enc}(mpk, \mathcal{K}); \\ \mathcal{A}^{OCorHK(\cdot)}(ct^*); \alpha &\leftarrow \mathcal{A}(ct^*) \end{vmatrix}
$$

where oracles ORegHK, OCorHK work with initial setting mpk = $\bot$ , aux = $\bot$ ,  $\mathcal{R} = \emptyset$ ,  $C = \emptyset$ ,  $\mathcal{K} = \emptyset$  and  $\mathcal D$  being a dictionary with  $\mathcal{D}[pk] = \emptyset$  for all possible pk:

- **–** ORegHK(C): run (pk, sk) ← Gen(crs, aux) and (mpk′ , aux′ ) ← Reg(crs, aux, pk, C), update mpk = mpk′ , aux = aux',  $\mathcal{D}[pk] = \mathcal{D}[pk] \cup \{C\}$ , append (pk, sk) to  $\mathcal R$  and return ( $|\mathcal R|$ , mpk, aux, pk);
- $-$  OCorHK(*i*): let  $\mathcal{R}[i] = (pk, sk)$  and  $C = \mathcal{D}[pk]$ , append pk to C and return sk.

Similarly, we require the Q-bound SIM security without malicious case. More concretely, it requires (i)  $\mathcal{R}[i] \neq \perp$ for each query  $i$  to OCorHK; (ii) for each  $(i, \text{pk}_i^*) \in C$  where  $|C| \leq Q$ , let  $\mathcal{R}[i] = (\text{pk}_i^*, \text{sk}_i^*)$  and  $C_i = \mathcal{D}[\text{pk}_i^*]$ , set  $K = K \cup \{(i, C_i^*, C_i^*(x^*), sk_i^*)\}.$ 

### <span id="page-11-0"></span>**3 Pairing-Based Global Slotted RBE**

In this section, we present a global slotted RBE relying on MDDH assumption.

#### **3.1 Construction**

Our construction works as follows:

 $-$  Setup $(1^{\lambda}, 1^L)$  : Generate  $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^{\lambda})$  and sample

$$
\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1) \times k}, \mathbf{k} \leftarrow \mathbb{Z}_p^{1 \times (k+1)}.
$$

For all  $i \in [L]$ , sample  $\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(k+1)\times (k+1)}$ ,  $\mathbf{r}_i \leftarrow \mathbb{Z}_p^{1\times k}$ . Output

$$
\text{crs} = \left( [\mathbf{A}]_1, \{ [\mathbf{A}\mathbf{V}_i]_1, [\mathbf{B}\mathbf{r}_i^{\mathsf{T}}, \mathbf{V}_i \mathbf{B}\mathbf{r}_i^{\mathsf{T}} + \mathbf{k}^{\mathsf{T}}]_2 \}_{i \in [L]}, \right)
$$

$$
\left\{ [\mathbf{V}_i \mathbf{B}\mathbf{r}_j^{\mathsf{T}}]_2 \}_{j \in [L], i \in [L] \setminus \{j\}}, [\mathbf{A}\mathbf{k}^{\mathsf{T}}]_T \right)
$$

.

.

 $-$  Gen(crs,  $i$ ) : Sample  $\mathbf{U}_i \leftarrow \mathbb{Z}_p^{(k+1)\times (k+1)}$ , output p $\mathsf{k}_i = (\llbracket \mathbf{A}\mathbf{U}_i \rrbracket_1, \{ \llbracket \mathbf{U}_i\mathbf{B}\mathbf{r}_j^{\scriptscriptstyle \sf T} \rrbracket_2 \}_{j\in [L]\setminus \{i\}})$  and s $\mathsf{k}_i = \mathbf{U}_i.$ 

 $\mathsf{P}\text{-}\mathsf{Ver}(\mathsf{crs},i,\mathsf{pk}_i)$  : Parse the public key  $\mathsf{pk}_i = (\llbracket \mathbf{A}\mathbf{U}_i \rrbracket_1,\{\llbracket \mathbf{U}_i\mathbf{Br}_j\rrbracket_2\}_{j\in[L]\setminus\{i\}}).$  For each  $j\in[L]\setminus\{i\},$  check

$$
e([{\bf A}]_1, [{\bf U}_i{\bf B}{\bf r}_j^{\scriptscriptstyle \sf T}]_2)\stackrel{?}{=}e([{\bf A}{\bf U}_i]_1, [{\bf B}{\bf r}_j^{\scriptscriptstyle \sf T}]_2).
$$

If above checks pass, output 1; otherwise, output 0.

 $-$  Agg(crs,  $\{i, \mathrm{pk}_i\}_{i \in [L]})$  : For all  $i \in [L]$ , parse  $\mathrm{pk}_i = (\mathbf{[AU}_i]_1, \{[\mathbf{U}_i\mathbf{Br}_i^{\mathsf{T}}]_2\}_{j \in [L] \setminus \{i\}})$ . Output

$$
mpk = \left( [\mathbf{A}]_1, [\mathbf{A}\mathbf{k}^{\mathsf{T}}]_T, \left[ \sum_{j \in [L]} (\mathbf{A}\mathbf{V}_j + \mathbf{A}\mathbf{U}_j) \right]_1 \right),
$$

and for all  $i \in [L]$ , output

$$
\mathsf{hsk}_i = \left(\underbrace{\mathbf{Br}_i^{\mathsf{T}}\left[2,\left[\underbrace{\mathbf{V}_i\mathbf{Br}_i^{\mathsf{T}}+\mathbf{k}^{\mathsf{T}}}_{\mathbf{k}_1^{\mathsf{T}}}\right]2,\left[\underbrace{\sum_{j\in[L]\backslash\{i\}}\left(\mathbf{V}_j\mathbf{Br}_i^{\mathsf{T}}+\mathbf{U}_j\mathbf{Br}_i^{\mathsf{T}}\right)}_{\mathbf{k}_2^{\mathsf{T}}}\right]_2\right)
$$

 $-$  Enc(mpk, m) : Parse mpk  $= \left([{\bf A}]_1,[{\bf A}{\bf k}^\tau]_T,\left[\sum_{j\in[L]}({\bf A}{\bf V}_j+{\bf A}{\bf U}_j)\right]_1\right)\! .$  Sample  ${\bf s}\leftarrow\mathbb{Z}_p^{1\times k},$  output

$$
ct = \left( \underbrace{SA \atop c_0} \underbrace{1}_{1}, \left( \underbrace{\sum_{j \in [L]} (sAV_j + sAU_j)}_{c_1} \right), \underbrace{[sAK^{\mathsf{T}}]_T \cdot m}_{C} \right).
$$

**−** Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) : Parse sk<sub>i\*</sub> = **U**<sub>i\*</sub>, hsk<sub>i\*</sub> = ([**k**<sub>0</sub><sup>†</sup>]<sub>2</sub>, [**k**<sub>1</sub><sup>†</sup>]<sub>2</sub>, [**k**<sub>2</sub><sup>†</sup>]<sub>2</sub>) and ct = ([**c**<sub>0</sub>]<sub>1</sub>, [**c**<sub>1</sub>]<sub>1</sub>, *C*). Compute

$$
[z_1]_T = e([\mathbf{c}_1]_1, [\mathbf{k}_0^{\mathsf{T}}]_2), \quad [z_2]_T = e([\mathbf{c}_0]_1, [\mathbf{k}_2^{\mathsf{T}}]_2),
$$
  
\n
$$
[z_3]_T = e([\mathbf{c}_0 \mathbf{U}_{i^*}]_1, [\mathbf{k}_0^{\mathsf{T}}]_2), \quad [z_4]_T = e([\mathbf{c}_0]_1, [\mathbf{k}_1^{\mathsf{T}}]_2),
$$
  
\n
$$
[z_5]_T = [z_1 - z_2 - z_3 - z_4]_T,
$$

and output  $z = C \cdot [z_5]_T$ .

**Completeness and Optimal Compactness.** For completeness, it just follows the definition of bilinear map *e* and the fact  $\bf A\cdot U_iBr_j^r=AU_i\cdot Br_j^r$ . As for optimal compactness, it is easy to see that the above construction satisfies our requirements, i.e.,  $|mpk| = poly(\lambda, \log L)$ ,  $|hsk_i| = poly(\lambda, \log L)$  and  $|ct| = poly(\lambda, \log L)$ .

**Correctness.** For all  $\lambda, L \in \mathbb{N}$ , all  $P$ , all  $i^* \in [L]$ , all crs  $\leftarrow$  Setup $(1^{\lambda}, 1^L)$ , all  $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Gen}(\mathsf{crs}, i^*)$ , all  ${pk_i}_{i \in [L] \setminus \{i^*\}}$  such that Ver(crs, *i*, pk<sub>*i*</sub>) = 1, for all m, we have

$$
\mathsf{hsk}_{i^*} = ([\mathbf{k}_0^{\!\top}]_2, [\mathbf{k}_1^{\!\top}]_2, [\mathbf{k}_2^{\!\top}]_2), \ \mathsf{ct} = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1, C).
$$

We obtain

$$
z_1 = \sum_{i \in [L]} (\mathbf{sAV}_i \mathbf{Br}_{i^*}^{\mathsf{T}} + \mathbf{sAU}_i \mathbf{Br}_{i^*}^{\mathsf{T}}),
$$
  
\n
$$
z_2 = \sum_{i \in [L] \setminus \{i^*\}} (\mathbf{sAV}_i \mathbf{Br}_{i^*}^{\mathsf{T}} + \mathbf{sAU}_i \mathbf{Br}_{i^*}^{\mathsf{T}}),
$$
  
\n
$$
z_3 = \mathbf{sAU}_{i^*} \mathbf{Br}_{i^*}^{\mathsf{T}},
$$
  
\n
$$
z_4 = \mathbf{sAV}_{i^*} \mathbf{Br}_{i^*}^{\mathsf{T}} + \mathbf{sAK}^{\mathsf{T}},
$$

and then

$$
z_5 = z_1 - z_2 - z_3 - z_4 = -s A k^{T}.
$$

Finally, we have  $z = C \cdot [z_5]_T = m$ . This proves the correctness.

#### **3.2 Security**

<span id="page-12-0"></span>**Theorem 1.** *Assume MDDH assumption holds, our pairing-based global slotted RBE achieves the IND security in the standard model as defined in Section [2.3.](#page-8-0)*

**Game Sequence.** We prove Theorem [1](#page-12-0) via the following game sequences. Let  $L$  be the number of slots,  $i^*$  be the challenge slot, and m\* be the challenge message; Suppose  $\{\mathsf{pk}_i^*\}_{i\in [L]}$  are challenge public keys to be registered. For all  $i \in [L]$ ,  $\mathcal{D}_i = \{pk_i : \mathcal{D}_i[pk_i] = sk_i \neq \perp\}$  stores the response to OGen(i);  $C_i = \{pk_i : (i, pk_i) \in C\}$  stores the response to  $OCor(i, \cdot)$ .

#### **–** Game<sub>0</sub> : Real Game. Recall that:

• the common reference string is that

$$
crs = \begin{pmatrix} [\mathbf{A}]_1, \{ [\mathbf{A}\mathbf{V}_i]_1, [\mathbf{B}\mathbf{r}_i^{\mathsf{T}}, \mathbf{V}_i \mathbf{B}\mathbf{r}_i^{\mathsf{T}} + \mathbf{k}^{\mathsf{T}}]_2 \}_{i \in [L]}, \\ \{ [\mathbf{V}_i \mathbf{B}\mathbf{r}_j^{\mathsf{T}}]_2 \}_{j \in [L], i \in [L] \setminus \{j\}}, [\mathbf{A}\mathbf{k}^{\mathsf{T}}]_T \end{pmatrix}
$$

.

• For each  $i \in [L]$ , each public key pk $_i \in \mathcal{D}_i$  is that

$$
\mathsf{pk}_{i} = ([\mathbf{A}\mathbf{U}_{i}]_{1}, \{ [\mathbf{U}_{i}\mathbf{Br}_{i}^{\mathsf{T}}]_{2} \}_{j \in [L] \setminus \{i\}}).
$$

It corresponds to secret key sk<sub>i</sub> =  $U_i$ .

• The challenge ciphertext is

$$
ct^* = \left( \underbrace{[\mathbf{s}\mathbf{A} \mathbf{A}]_1, \left[ \underbrace{\sum_{j \in [L]} (\mathbf{s}\mathbf{A}\mathbf{V}_j + \mathbf{s}\mathbf{A}\mathbf{U}_j)}_{\mathbf{c}_1} \right]_1, \underbrace{[\mathbf{s}\mathbf{A}\mathbf{k}^\mathsf{T}]_T \cdot m}_{C} \right)
$$

,

where  $\mathbf{s} \leftarrow \mathbb{Z}_p^{1 \times k}$ .

 $-$  Game $_1$  : Identical to Game $_0$  except that we replace **sA** in challenge ciphertext with  $\mathbf{c} \leftarrow \mathbb{Z}_p^{1\times (k+1)}$ . Then the challenge ciphertext is  $\overline{1}$ 

$$
ct^* = \left( \underbrace{\begin{bmatrix} \boxed{c} & 1 \\ \ddots & \ddots \end{bmatrix}}_{c_0} \frac{\sum_{j \in [L]} (\boxed{c}V_j + cU_j)}_{c_1} \right)_{1} \cdot \underbrace{\begin{bmatrix} c & k^T \end{bmatrix}_{T} \cdot m}_{C} \right).
$$

Observe that we have Game<sub>0</sub>  $\approx_c$  Game<sub>1</sub>, which follows the MDDH assumption, ensuring that ([**A**]<sub>1</sub>, [**sA**]<sub>1</sub>)  $\approx_c$  $([A]_1, [c]_1)$  $([A]_1, [c]_1)$  $([A]_1, [c]_1)$  where  $A \leftarrow \mathbb{Z}_p^{k \times (k+1)}, s \leftarrow \mathbb{Z}_p^{1 \times k}$  and  $c \leftarrow \mathbb{Z}_p^{1 \times (k+1)}.$  See Lemma 1 for more details.

**–** Game2,ℓ,<sup>1</sup> (ℓ ∈ []) : Identical to Game2,ℓ−1,<sup>3</sup> except that we replace [**Br**<sup>⊤</sup> ℓ , **V**ℓ**Br**<sup>⊤</sup> ℓ + **k** ⊤ ]<sup>2</sup> in crs with the following form

$$
\left[\ \overline{\mathbf{d}_{\ell}^{\mathsf{T}}},\mathbf{V}_{\ell}\ \overline{\mathbf{d}_{\ell}^{\mathsf{T}}}+\mathbf{k}^{\mathsf{T}}\right]_{2},
$$

where  $\mathbf{d}_\ell \gets \mathbb{Z}_p^{1\times (k+1)}$  . We have Game<sub>2, $\ell-1,3\approx_c$  Game $_{2,\ell,1}$ , which follows the MDDH assumption (  $[\mathbf{B}]_2, [\mathbf{Br}_{\ell}^{\scriptscriptstyle{\text{T}}}]_2)\approx_c$ </sub>  $([\mathbf{B}]_2, [\mathbf{d}_\ell^\mathsf{T}]_2)$  $([\mathbf{B}]_2, [\mathbf{d}_\ell^\mathsf{T}]_2)$  $([\mathbf{B}]_2, [\mathbf{d}_\ell^\mathsf{T}]_2)$ , where  $\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$ ,  $\mathbf{r}_\ell \leftarrow \mathbb{Z}_p^{1\times (k+1)}$  and  $\mathbf{d}_\ell \leftarrow \mathbb{Z}_p^{1\times k}$ . See Lemma 2 for more details.

 $-$  Game $_{2,\ell,2}$  : Identical to Game $_{2,\ell,1}$  except that we change replace  $[$ **d** $_\ell^r$ , **V** $_\ell$ **d** $_\ell^r$  + **k** $^\top$ ] $_2$  in crs with the following form

$$
[\mathbf{d}_\ell^\top,\mathbf{V}_\ell\mathbf{d}_\ell^\top+\alpha\mathbf{c}^\perp+\mathbf{k}^\top]_2,
$$

where  $\alpha_\ell\gets\Z_p$  and  $\mathbf{c}^\perp\gets\Z_p^{2k+1}$  such that  $\mathbf{A}\mathbf{c}^\perp=0,\mathbf{c}\mathbf{c}^\perp=1.$  Note that Game<sub>2,0</sub> is identical to Game<sub>1</sub>; we have Game<sub>2,ℓ,1</sub>  $\approx$ <sub>s</sub> Game<sub>2,ℓ,2</sub>, see Lemma [3](#page-15-1) for more details.

 $-$  Game<sub>2,ℓ,3</sub> : Identical to Game<sub>2,ℓ,2</sub> except that we change replace  $[\bm{d}^{\tau}_{\ell},\bm{V}_{\ell}\bm{d}^{\tau}_{\ell}+\alpha\bm{c}^{\bot}+\bm{k}^{\tau}]_2$  in crs with the following form

$$
\left[\overline{\mathbf{Br}^{\mathsf{T}}_{\ell}},\mathbf{V}_{\ell}\overline{\mathbf{Br}^{\mathsf{T}}_{\ell}}+\alpha\mathbf{c}^{\perp}+\mathbf{k}^{\mathsf{T}}\right]_{2}.
$$

Notice that we can prove Game<sub>[2](#page-14-0), $\ell$ ,2  $\approx$ <sub>c</sub> Game<sub>2, $\ell$ </sub>,3 via Lemma 2 as well.</sub>

- Game<sub>3</sub> : Identical to Game<sub>2,L</sub> except that we replace the ciphertext into the following form:

$$
ct^* = \left( \underbrace{c}_{c_0} \underbrace{1}_{c_1} \left[ \underbrace{\sum_{j \in [L]} (cV_j + cU_j)}_{c_1} \right], \underbrace{\overline{g_T^*}}_{C} \right).
$$

where  $g_T^*$  is sampled uniformly over  $G_T$ . We claim that Game $_{2,L}$   $\approx_s$  Game $_3$  which follows the following the statistical argument:

$$
(\mathbf{A}\mathbf{k}^{\!\scriptscriptstyle{\top}},\mathbf{k}^{\!\scriptscriptstyle{\top}} + \alpha \mathbf{c}^{\perp},\mathbf{c}\mathbf{k}^{\!\scriptscriptstyle{\top}}) \approx_s (\mathbf{A}\mathbf{k}^{\!\scriptscriptstyle{\top}},\mathbf{k}^{\!\scriptscriptstyle{\top}},\mathbf{c}\mathbf{k}^{\!\scriptscriptstyle{\top}} - \alpha)
$$

where  $[\mathbf{ck}^{\dagger} - \alpha]_T$  is uniform, namely  $g_T^*$ .

<span id="page-13-0"></span>In the following, we use Adv ${}^l_{\mathcal{A}}(\lambda)$  to denote the advantage of  $\mathcal A$  in Game ${}_i.$ 

**Lemma 1 (**Game<sub>0</sub>  $\approx_c$  Game<sub>1</sub>). For any efficient adversary A, there exists algorithm  $\mathcal{B}_1$  with close running time to A *such that*

$$
|\mathsf{Adv}_{\mathcal{A}}^0(\lambda)-\mathsf{Adv}_{\mathcal{A}}^1(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MDDH}}(\lambda)+\mathsf{negl}(\lambda).
$$

*Proof.* Recall that the difference between two games is that we replace  $[sA]_1$  in Game<sub>0</sub> with  $[c]_1$ , where  $A \leftarrow$  $\mathbb{Z}_p^{k\times (k+1)}$ ,  $\mathbf{s}\leftarrow\mathbb{Z}_p^{1\times k}$  and  $\mathbf{c}\leftarrow\mathbb{Z}_p^{1\times (k+1)}$ . This follows from  $(k,k+1,1)$ -MDDH assumption, which ensures that:

$$
([A]_1, [sA]_1) \approx_c ([A]_1, [c]_1).
$$

On input  $([A]_1, [t]_1)$  where  $t = sA$  or  $t = c$ , algorithm  $B_1$  works as follows:

**Setup.** Sample

$$
\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k}, \mathbf{k} \leftarrow \mathbb{Z}_p^{1\times (k+1)}, \{\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(k+1)\times (k+1)}, \mathbf{r}_i \leftarrow \mathbb{Z}_p^{1\times k}\}_{i\in[L]}.
$$

**Output** 

$$
\text{crs} = \begin{pmatrix} [\mathbf{A}]_1, \{ [\mathbf{A}\mathbf{V}_i]_1, [\mathbf{B}\mathbf{r}_i^{\mathsf{T}}, \mathbf{V}_i \mathbf{B}\mathbf{r}_i^{\mathsf{T}} + \mathbf{k}^{\mathsf{T}}]_2 \}_{i \in [L]},\\ \{ [\mathbf{V}_i \mathbf{B}\mathbf{r}_j^{\mathsf{T}}]_2 \}_{j \in [L], i \in [L] \setminus \{j\}}, [\mathbf{A}\mathbf{k}^{\mathsf{T}}]_T \end{pmatrix}.
$$

**Query.** Here, for all  $i \in [L]$  and each  $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathcal{D}_i$  is generated honestly as :

$$
\mathsf{pk}_{i} = (\left[\mathbf{A}\mathbf{U}_{i}\right]_{1}, \left\{\left[\mathbf{U}_{i}\mathbf{B}\mathbf{r}_{j}^{\mathsf{T}}\right]_{2}\right\}_{j \in [L] \setminus \{i\}})
$$

and  $sk_i = U_i$  where  $U_i \leftarrow \mathbb{Z}_p^{(k+1)\times (k+1)}$ . **Challenge.** On input challenge m<sup>∗</sup> , output

$$
ct^* = \left( [\mathbf{t}]_1, \left[ \sum_{j \in [L]} (\mathbf{t} \mathbf{V}_j + \mathbf{t} \mathbf{U}_j) \right]_1, [\mathbf{t} \mathbf{k}^\top]_T \cdot \mathsf{m}^* \right).
$$

Observe that when  $t = sA$ , the simulation is identical to Game<sub>0</sub>; when  $t = c$ , the simulation is identical to Game<sub>1</sub>. This readily proves the lemma. ⊓⊔

<span id="page-14-0"></span>**Lemma 2** (Game<sub>2,ℓ-1,3</sub>  $\approx$ <sub>c</sub> Game<sub>2,ℓ,1</sub>). *For any efficient adversary*  $\mathcal{A}$ , there exists algorithm  $\mathcal{B}_2$  with close running *time to* A *such that*

$$
|\mathsf{Adv}_{\mathcal{A}}^{2,\ell-1,3}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{2,\ell,1}(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{MDDH}}(\lambda)+\mathsf{negl}(\lambda).
$$

*Proof.* This follows from the  $(k, k + 1, 1)$ -MDDH assumption:

$$
([\mathbf{B}]_2, [\mathbf{Br}^\mathsf{T}_\ell]_2) \approx_c ([\mathbf{B}]_2, [\mathbf{d}^\mathsf{T}_\ell]_2)
$$

 $\mathbf{w}$ here  $\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k}, \mathbf{r}_\ell \leftarrow \mathbb{Z}_p^{1\times k}$  and  $\mathbf{d}_\ell \leftarrow \mathbb{Z}_p^{1\times (k+1)}.$  On input  $[\mathbf{B}]_2, [\mathbf{t}^\intercal]_2$  where  $\mathbf{t}^\intercal = \mathbf{B}\mathbf{r}^\intercal_\ell$  or  $\mathbf{t}^\intercal = \mathbf{d}^\intercal_\ell$ , the algorithm  $\mathcal{B}_2$ works as follow:

**Setup.** Sample

$$
\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \ \mathbf{k} \leftarrow \mathbb{Z}_p^{1 \times (k+1)}, \ \mathbf{c} \leftarrow \mathbb{Z}_p^{1 \times (k+1)}, \ \alpha \leftarrow \mathbb{Z}_p, \{\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}, \ \mathbf{r}_i \leftarrow \mathbb{Z}_p^{1 \times k}\}_{i \in [L]}.
$$

Output

$$
\text{crs} = \left( \begin{array}{c} [\mathbf{A}\mathbf{k}^{\text{T}}]_{T} [\mathbf{A}]_{1}, \{ [\mathbf{A}\mathbf{V}_{i}]_{1} \}_{i \in [L]}, \{ [\mathbf{V}_{i}\mathbf{B}\mathbf{r}_{j}^{\text{T}}]_{2} \}_{j \in [L], i \in [L] \setminus \{j\}}, \\ \{ [\mathbf{B}\mathbf{r}_{i}^{\text{T}}, \mathbf{V}_{i}\mathbf{B}\mathbf{r}_{i}^{\text{T}} + \mathbf{k}^{\text{T}} + \alpha \mathbf{c}^{\perp}]_{2} \}_{i < \ell}, \left[ [\mathbf{t}_{\ell}, \mathbf{V}_{\ell}\mathbf{t}_{\ell}^{\text{T}} + \mathbf{k}^{\text{T}}]_{2} \right]_{i} \\ \{ [\mathbf{B}\mathbf{r}_{i}^{\text{T}}, \mathbf{V}_{i}\mathbf{B}\mathbf{r}_{i}^{\text{T}} + \mathbf{k}^{\text{T}}]_{2} \}_{i > \ell} \end{array} \right)
$$

,

<span id="page-15-2"></span>**Query.** Here, for all  $i \in [L]$  and each  $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathcal{D}_i$  is generated honestly as:

 $-$  if  $i \neq \ell$ , the pk<sub>i</sub> is that

 $([\mathbf{A}\mathbf{U}_{i}]_1, \{[\mathbf{U}_{i}\mathbf{B}\mathbf{r}_{i}^{\mathsf{T}}]_2\}_{j\in[L]\backslash\{i,\ell\}}, [\mathbf{U}_{i}\mathbf{t}^{\mathsf{T}}]_2);$ 

 $-$  if  $i = \ell$ , the pk<sub> $\ell$ </sub> is that

$$
([\mathbf{A}\mathbf{U}_{\ell}]_1, \{[\mathbf{U}_{\ell} \mathbf{B}\mathbf{r}_{j}^{\mathsf{T}}]_2\}_{j \in [L] \setminus \{\ell\}}).
$$

where  $\mathbf{U}_i \leftarrow \mathbb{Z}_p^{(k+1)\times (k+1)}$ .

**Challenge.** On input challenge m<sup>∗</sup> , output ct<sup>∗</sup> as

$$
ct^* = \left( [\mathbf{c}]_1, \left[ \sum_{j \in [L]} (\mathbf{c} \mathbf{V}_j + \mathbf{c} \mathbf{U}_j) \right]_1, [\mathbf{c} \mathbf{k}^\top]_T \cdot \mathsf{m}^* \right).
$$

Observe that when  $f = Br_{\ell}^{\tau}$ , the simulation is identical to Game<sub>2,ℓ−1,3</sub>; when  $f = d_{\ell}^{\tau}$ , the simulation is identical to  $\mathsf{Game}_{2,\ell,1}$ . □

<span id="page-15-1"></span>**Lemma 3 (**Game<sub>2,ℓ,1</sub>  $\approx$ <sub>s</sub> Game<sub>2,ℓ,2</sub>). *For any efficient adversary A*, there exists algorithm  $B_3$  with close running time *to* A *such that*

$$
|\mathsf{Adv}_{\mathcal{A}}^{2,\ell,1}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{2,\ell,2}(\lambda)|\leq \mathsf{negl}(\lambda).
$$

*Proof.* Here, it only considers honest case, then we have Game<sub>2,ℓ,1</sub>  $\approx$ <sub>s</sub> Game<sub>2,ℓ,2</sub> by the following argument:

$$
\begin{cases}\n\mathbf{A}, \mathbf{B}_{\ell}, \mathbf{c}^{\perp}, \mathbf{d}_{\ell}^{\top}, \mathbf{A}\mathbf{V}_{\ell}, \mathbf{V}_{\ell}\mathbf{B}, \mathbf{V}_{\ell}\mathbf{d}_{\ell}^{\top} + bc^{\perp}\alpha; & // \text{crs} \\
\mathbf{A}\mathbf{U}_{\ell}; \quad \mathbf{c}, \mathbf{c}\mathbf{V}_{\ell} + \mathbf{c}\mathbf{U}_{\ell} & // \text{pk}_{i}; \text{ct}^* \\
& \approx \n\begin{cases}\n\mathbf{A}, \mathbf{B}_{\ell}, \mathbf{c}^{\perp}, \mathbf{d}_{\ell}^{\top}, \mathbf{A}\mathbf{V}_{\ell}, \mathbf{V}_{\ell}\mathbf{B}, \mathbf{V}_{\ell}\mathbf{d}_{\ell}^{\top} + \mathbf{v}_{\ell}\mathbf{c}^{\perp} + bc^{\perp}\alpha; \\
\mathbf{A}\mathbf{U}_{\ell}; \quad \mathbf{c}, \mathbf{c}\mathbf{V}_{\ell} + \mathbf{c}\mathbf{U}_{\ell} + \mathbf{v}_{\ell}\mathbf{c}^{\perp} + u_{\ell}\mathbf{c}^{\perp} \\
& \approx \n\end{cases}\n\begin{cases}\n\mathbf{A}, \mathbf{B}_{\ell}, \mathbf{c}^{\perp}, \mathbf{d}_{\ell}^{\top}, \mathbf{A}\mathbf{V}_{\ell}, \mathbf{V}_{\ell}\mathbf{B}, \mathbf{V}_{\ell}\mathbf{d}_{\ell}^{\top} + \mathbf{v}_{\ell}\mathbf{c}^{\perp} + be^{\perp}\alpha; \\
\mathbf{A}\mathbf{U}_{\ell}; \quad \mathbf{c}, \mathbf{c}\mathbf{V}_{\ell} + \mathbf{c}\mathbf{U}_{\ell} + \mathbf{v}_{\ell}\mathbf{c}^{\perp} + u_{\ell}\mathbf{c}^{\perp}\n\end{cases}
$$

where  $b \in \{0, 1\}$ . Here, we have

**–** The first  $\approx$ <sub>s</sub> follows that:

 $\mathbf{V}_{\ell} \mapsto \mathbf{V}_{\ell} + \mathbf{c}^{\perp} \mathbf{v}_{\ell} \mathbf{d}^{\perp}$  and  $\mathbf{U}_{\ell} \mapsto \mathbf{U}_{\ell} + \mathbf{c}^{\perp} u_{\ell} \mathbf{d}^{\perp}$ 

where  $\mathbf{c}^{\perp} \in \mathbb{Z}_p^{k+1}$  and  $\mathbf{d}^{\perp} \in \mathbb{Z}_p^{1 \times (k+1)}$  such that  $\mathbf{A} \mathbf{c}^{\perp} = 0, \mathbf{c} \mathbf{c}^{\perp} = 1, \mathbf{d}^{\perp} \mathbf{B} = 0, \mathbf{d}^{\perp} \mathbf{d}_{\ell} = 1$ .

– The second  $\approx_s$  holds by the fact that  $v_\ell$  in crs also seems to be sampled randomly because  $u_\ell$  hides extra  $v_\ell$  in challenge ciphertext.

This readily proves the lemma. ⊓⊔

# <span id="page-15-0"></span>**4 Lattice-Based Global Slotted RBE**

In this section, we give a global slotted RBE construction based on function-binding hash function (relying on LWE assumption) [\[FWW23\]](#page-28-5) and witness encryption (relying on evasive LWE assumption) [\[Wee22\]](#page-29-2). This construction is adapted from slotted RABE (with a public randomized aggregation procedure) in [\[FWW23\]](#page-28-5). Concretely, we initially construct a global slotted RBE construction that achieves adaptive security subject to the restriction that adversary does not make any corruption queries. Then we use the "two-key" technology [\[GW09\]](#page-28-11) to remove this restriction and obtain a global slotted RBE that achieves the adaptive security (as defined in Section [2\)](#page-7-0) in the random oracle.

#### <span id="page-16-0"></span>**4.1 Construction without Corruption**

Assume a public key encryption PKE = (Setup, Enc, Dec) with all parameters of size poly( $\lambda$ ), a function-binding hash function FBH = (Setup, Hash, Open, Ver) with block size  $m_{in} = \lambda + \log L$  [\[FWW23\]](#page-28-5), and a witness encryption WE = (Enc, Dec) for a NP language  $\mathcal L$  with witness relation  $\mathcal R$  [\[VWW22,](#page-29-4)[Tsa22\]](#page-28-16) defined as follows:

$$
\mathcal{R}((\text{hk}, \text{pk}, \text{dig}), (i, \text{ct}, r, \pi)) = 1
$$
  
\n
$$
\Leftrightarrow \text{ct} = \text{PKE}.\text{Enc}(\text{pk}, 1; r) \land \text{FBH}.\text{Ver}(\text{hk}, \text{dig}, \{i\}, \{(i, (i, \text{ct}))\}, \pi) = 1
$$

our construction is as follows:

- $-$  Setup $(1^{\lambda},1^L)$  : Run  $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{PKE}.\mathsf{Setup}(1^{\lambda})$  and  $\mathsf{hk} \leftarrow \mathsf{FBH}.\mathsf{Setup}(1^{\lambda},L)$ , and output the common reference string  $\text{crs} = (\text{pk}, \text{hk}).$
- $-$  Gen(crs, i) : Parse crs = (pk, hk). Sample  $r \leftarrow \{0,1\}^{\lambda}$  and run ct  $\leftarrow$  PKE.Enc(pk, 1; r), then output pk<sub>i</sub> = ct and  $sk_i = r$ .
- Ver(crs,  $i, \mathrm{pk}_i$ ) : Check whether pk<sub>i</sub> is a valid ciphertext of PKE. If so, output 1; otherwise, output 0.
- $-$  Agg(crs,  $\{i, \mathrm{pk}_i\}_{i \in [L]})$  : Parse crs = (pk, hk). Then run

$$
\mathsf{dig} \leftarrow \mathsf{FBH}.\mathsf{Hash}(\mathsf{hk},((1,\mathsf{pk}_1),\ldots,(L,\mathsf{pk}_L))),
$$

$$
\pi_i \leftarrow \textsf{FBH.Open}(\textsf{hk}, ((1, \textsf{pk}_1), \dots, (L, \textsf{pk}_L)), \{i\}), \forall i = 1, \dots, L.
$$

Output the master public key mpk = (crs, dig) and the helper secret key hsk<sub>j</sub> =  $(j, \pi_j, \text{pk}_j)$  for all  $j \in [L]$ .

**–** Enc(mpk, m) : Parse mpk = ( (pk, hk), dig). Output the ciphertext

$$
ct \leftarrow \mathsf{WE}.\mathsf{Enc}(1^{\lambda}, (\mathsf{hk}, \mathsf{pk}, \mathsf{dig}), \mathsf{m}).
$$

 $-$  Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) : Parse hsk<sub>i\*</sub> =  $(i^*, \pi_{i^*}, \text{pk}_{i^*})$ . Output

$$
m = WE.Dec(ct, (i^*, pk_{i^*}, sk_{i^*}, \pi_{i^*})).
$$

**Optimal Compactness and Unbounded Users.** Note that  $|crs| = |pk| + |hk| = poly(\lambda, \log L)$ ,  $|mpk| = |crs| + |diq|$ poly( $\lambda$ , log L),  $|h$ sk<sub>i</sub> $| = |j| + |\pi_j| + |pk_j| = poly(\lambda, \log L)$ , and the runtime for algorithms PKE.Enc and FBH.Ver are at most poly( $\lambda$ , log L), so above construction supports optimal compactness and unbounded users.

**Correctness.** For all  $\lambda, L \in \mathbb{N}$ , all  $C$ , and all  $i^* \in [L]$ , all crs  $\leftarrow$  Setup $(1^{\lambda}, 1^L, C)$  where crs = (pk, hk), all (pk<sub>i\*</sub>, sk<sub>i\*</sub>)  $\leftarrow$ Gen(crs, i<sup>\*</sup>), all {pk<sub>i</sub>}<sub>i∈[L]\{i\*}</sub> such that Ver(crs, i, pk<sub>i</sub>) = 1, and all message m, we have mpk = (crs, dig) and hsk<sub>i\*</sub> =  $(i^*, \pi_{i^*}, \mathsf{pk}_{i^*}),$  where

$$
dig \leftarrow \text{FBH.Hash}(hk, ((1, pk_1), \dots, (L, pk_L))),
$$

$$
\pi_{i^*} \leftarrow \text{FBH.Dpen}(hk, ((1, pk_1), \dots, (L, pk_L)), \{i^*\}).
$$

Then the ciphertext is computed as

 $\mathsf{ct} \leftarrow \mathsf{WE}.\mathsf{Enc}(1^{\lambda}, (\mathsf{hk}, \mathsf{pk}, \mathsf{dig}), \mathsf{m}).$ 

For decryption, we state that hsk $_{i^*} = (i^*, \pi_{i^*}, \text{pk}_{i^*})$  along with sk $_{i^*}$  is a valid witness for the statement (hk, pk, dig), because pk<sub>i\*</sub> = PKE.Enc(pk, 1; sk<sub>i\*</sub>), and FBH.Ver(hk, dig, {i\*}, {(i\*, (i\*, pk<sub>i\*</sub>))},  $\pi_{i^*}$ ) = 1 by the completeness of FBH. Thus, by the correctness of witness encryption, we have

$$
WE.Dec(ct, (i^*, pk_{i^*}, sk_{i^*}, \pi_{i^*})) = m.
$$

#### **4.2 Security**

<span id="page-17-0"></span>**Theorem 2.** *Assume* PKE = (Setup, Enc, Dec) *is a public key encryption with semantic security,* FBH = (Setup, Hash, Open, Ver) *is a function-binding hash function with function hiding and function bind properties, and* WE = (Enc, Dec) *is a secure witness encryption, our global slotted RBE achieves the adaptive IND security without corruption.*

**Game Sequence.** We prove Theorem [2](#page-17-0) via a sequence of games as follows:

- **–** H0: Real Game.
- $-$  H<sub>1</sub>: this game is identical to H<sub>0</sub> except that for each query to oracle OGen(crs, i) where crs = (pk, hk), the challenger generates  $(\mathsf{pk}_i, \mathsf{sk}_i)$  as follows:

$$
pk_i = PKEEnc(pk, 0; r), sk_i = r
$$

where  $r \leftarrow \{0,1\}^{\lambda}$ . We can prove H $_0 \approx_c H_1$  via Lemma [4.](#page-17-1)

–  $\rm{H}_{2}:$  this game is identical to  $\rm{H}_{1}$  except that we replace FBH.Setup( $1^\lambda,1^L$ ) with <code>FBH.SetupBind( $1^\lambda,1^L,f_g)$ </code> . Here, the function  $f_g$  is defined as

$$
f_g((1, \text{ct}_1), \ldots, (L, \text{ct}_L)) = \bigvee_{i \in [L]} g(i, \text{ct}_i),
$$

where

$$
g(i, ct_i) = \begin{cases} 1, & \text{when } PKE.Dec(sk, ct_i) = 1, \\ 0, & \text{otherwise.} \end{cases}
$$

sk is the secret key corresponding to public key pk of PKE. We can prove  $H_1 \approx_c H_2$  via Lemma [5.](#page-17-2)

 $-$  H<sub>3</sub>: this game is identical to H<sub>2</sub> except that the challenge ciphertext is defined as

$$
\mathsf{ct}^* \leftarrow \mathsf{WE}.\mathsf{Enc}(1^{\lambda}, (\mathsf{hk}, \mathsf{pk}, \mathsf{dig}), \overline{\mathsf{m}^*}),
$$

where m $<sup>∗</sup>$  is a random message. This can be proved via Lemma [6.](#page-17-3)</sup>

<span id="page-17-1"></span>**Lemma 4 (**H<sub>0</sub>  $\approx_c$  H<sub>1</sub>). *For any adversary* A, there exists algorithm  $\mathcal{B}_1$  with close running time to A such that

$$
|\mathsf{Adv}_{\mathcal{A}}^0(\lambda)-\mathsf{Adv}_{\mathcal{A}}^1(\lambda)|\leq q\cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{PKE}}(\lambda)+\mathsf{negl}(\lambda).
$$

*where is the bound of the number of quires to oracle* OGen*.*

*Proof.* The proof follows the semantic security of public-key encryption PKE. Roughly speaking, since the adversary has no idea about the secret key sk, the challenger is able to change all ciphertexts in a one-by-one fashion, without raising any doubt. Here, we omit details. ⊓⊔

<span id="page-17-2"></span>**Lemma 5 (**H<sub>1</sub>  $\approx$ <sub>c</sub> H<sub>2</sub>). *For any adversary* A, there exists algorithm  $B_2$  with close running time to A such that

$$
|\mathsf{Adv}_{\mathcal{A}}^1(\lambda)-\mathsf{Adv}_{\mathcal{A}}^2(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{FBH-CFH}}(\lambda)+\mathsf{negl}(\lambda).
$$

*Proof.* The proof follows the computational function hiding property of FBH. The challenger can replace algorithm FBH.Setup with FBH.SetupBind without raising any doubt. ⊓⊔

<span id="page-17-3"></span>**Lemma 6 (**H<sub>2</sub>  $\approx$  H<sub>3</sub>). For any adversary A, there exists algorithm  $\mathcal{B}_3$  with close running time to A such that

$$
|\mathsf{Adv}_{\mathcal{A}}^1(\lambda)-\mathsf{Adv}_{\mathcal{A}}^2(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}_3}^{\mathsf{WE}}(\lambda)+\mathsf{negl}(\lambda).
$$

<span id="page-18-0"></span>*Proof.* The proof follows the security of witness encrytion WE and the statistical function binding property of FBH. Recall the witness relation R:

$$
\mathcal{R}((\text{hk}, \text{pk}, \text{dig}), (i, \text{ct}, r, \pi)) = 1
$$
  
\n
$$
\Leftrightarrow \text{ct} = \text{PKE}.\text{Enc}(\text{pk}, 1; r) \land \text{FBH}.\text{Ver}(\text{hk}, \text{dig}, \{i\}, \{(i, (i, \text{ct}))\}, \pi) = 1.
$$

If some witness  $(i, \text{ct}, r, \pi)$  corresponding to instance (hk, pk, dig), it must hold that PKE.Dec(sk, ct) = 1, then we have  $f_g((1, ct_1), \ldots, (L, ct_L)) = 1$ . However, since  $g(i, ct_i) = 0$  for all  $i \in [L]$  in  $H_2$ , it means  $f_g((1, ct_1), \ldots, (L, ct_L)) =$ 0. Combining with statistical function binding property, it is clear that adversary is able to pass the relation  $\mathcal R$ with only negligible probability. Therefore, the challenger can exploit the security of WE to replace the challenge ciphertext ct $^* \leftarrow \mathsf{WE}.\mathsf{Enc}(1^\lambda,(\mathsf{hk},\mathsf{pk},\mathsf{dig}),\mathsf{m}_b)$  with ct $^* \leftarrow \mathsf{WE}.\mathsf{Enc}(1^\lambda,(\mathsf{hk},\mathsf{pk},\mathsf{dig}),\mathsf{m}^*$ ). ⊓⊔

#### **4.3 Final Construction**

Assume a hash function H  $: \{0,1\}^* \to \{0,1\}^L$  that can be modeled as random oracle, a global slotted RBE without corruption gsRBE<sub>wc</sub> = (Setup, Gen, Ver, Agg, Enc, Dec) that all parameters are of size poly( $\lambda$ ), our final construction is as follows:

- $-$  Setup $(1^{\lambda},1^L)$  : Run crs  $\leftarrow$  gsRBE<sub>wc</sub>.Setup $(1^{\lambda})$  and output the common reference string crs.
- **–** Gen(crs, ) : Sample two pairs of public key and secret

 $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{gsRBE}_{\mathsf{wc}}.\mathsf{Gen}(\mathsf{crs}, i), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{gsRBE}_{\mathsf{wc}}.\mathsf{Gen}(\mathsf{crs}, i).$ 

Then sample a random bit  $\beta \leftarrow \{0, 1\}$  and  $s \leftarrow \{0, 1\}^{\lambda}$ , output p $k_i = (pk_0, pk_1, s)$ , s $k_i = (\beta, sk_\beta)$ .

- Ver(crs, i, pk<sub>i</sub>): Parse pk<sub>i</sub> = (pk<sub>i.0</sub>, pk<sub>i.1</sub>, s<sub>i</sub>). If both pk<sub>i.0</sub> and pk<sub>i.1</sub> pass the check of algorithm gsRBE<sub>wc</sub>.Ver, and  $s_i \in \{0,1\}^{\lambda}$ , output 1; otherwise, output 0.
- $-$  Agg(crs,  $\{i, \mathsf{pk}_i\}_{i \in [L]})$  : Parse  $\mathsf{pk}_i = (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, s_i)$ . Compute  $(h_1, \ldots, h_L) \leftarrow \mathsf{H}(\mathsf{crs}, (1, \mathsf{pk}_1), \ldots, (L, \mathsf{pk}_L))$  and let  $\overline{h}_i = 1 - h_i$  for all  $i \in [L]$ . Then run

$$
(\mathsf{mpk}_0, \{\mathsf{hsk}_{j,0}\}_{j \in [L]}) \leftarrow \mathsf{gsRBE}_{\mathsf{wc}}.\mathsf{Agg}(\mathsf{crs}, ((1, \mathsf{pk}_{1,h_1}), \dots, (L, \mathsf{pk}_{L,h_L}))),
$$
  

$$
(\mathsf{mpk}_1, \{\mathsf{hsk}_{j,1}\}_{j \in [L]}) \leftarrow \mathsf{gsRBE}_{\mathsf{wc}}.\mathsf{Agg}(\mathsf{crs}, ((1, \mathsf{pk}_{1,\overline{h}_1}), \dots, (L, \mathsf{pk}_{L,\overline{h}_L}))).
$$

Output the master public key mpk = (mpk<sub>0</sub>, mpk<sub>1</sub>) and the helper secret key hsk<sub>j</sub> =  $(h_j, h$ sk<sub>j,0</sub>, hsk<sub>j,1</sub>) for all  $j \in [L]$ .

- $-$  Enc(mpk, m) : Parse mpk = (mpk<sub>0</sub>, mpk<sub>1</sub>), output ct = (ct<sub>0</sub>, ct<sub>1</sub>), where ct<sub>0</sub>  $\leftarrow$  gsRBE<sub>wc</sub>.Enc(mpk<sub>0</sub>, m) and  $ct_1 \leftarrow$  gsRBE<sub>wc</sub>.Enc(mpk<sub>1</sub>, m).
- $-$  Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) : Parse hsk<sub>i\*</sub> = ( $h_{i*}$ , hsk<sub>i\*,0</sub>, hsk<sub>i\*,1</sub>), sk<sub>i\*</sub> = ( $\beta_{i*}$ , sk<sub> $\beta_{i*}$ </sub>) and ct = (ct<sub>0</sub>, ct<sub>1</sub>). If  $\beta_{i*} = h_{i*}$ , output  $m = gSRBE_{wc}$ .Dec(hsk<sub>i\*,0</sub>, sk<sub>i\*, $\beta_{i^*}$ </sub>, ct<sub>0</sub>); otherwise, output  $m = gSRBE_{wc}$ .Dec(hsk<sub>i\*,1</sub>, sk<sub>i\*, $\beta_{i^*}$ </sub>, ct<sub>1</sub>).

**Optimal Compactness and Unbounded Users.** Note that the final construction consists of two instances for  $gSRBE_{wc}$ , so it will meet optimal compactness (resp., unbounded users) as long as  $gSRBE_{wc}$  meets optimal compactness (resp., unbounded users).

 $\bf{Correctness.}$  Informally, the public key pk $_{i^*,\beta_{i^*}}$  corresponding to secret key sk $_{i^*,\beta_{i^*}}$  is either registered into mpk $_0$ (when  $\beta_{i^*}=\dot h_{i^*}$ ) or mpk $_1$  (when  $\beta_{i^*}=\bar h_{i^*}$ ), then it can recover message m properly following the correctness of gsRBE<sub>wc</sub>.

As for security, this construction achieves the adaptive IND security in random oracle model and it allows the query to corruption oracle OCor. The proof strategy is analogous to [\[FWW23\]](#page-28-5). The difference lies on the fact that we do not require any challenge policy, so our construction is naturally adaptively IND secure.

Finally, we obtain an Q-bound slotted RFE for circuits via the compiler in Section [5](#page-19-0) and Section [6,](#page-23-0) and we can modify the compiler to achieve unbounded users by eliminating all  $\{\bar{\sf pk}_{i,w,b}\}$  of crs and just setting  $\overline{\sf pk}_{i,w,b} = \sf pk_{i,w,b}$ only when  $C_i[w] = b$ . Here, due to the good traits of FBH, algorithm sRBE.Agg still works as usual when the number of registered public keys is less than  $L$ .

# <span id="page-19-0"></span>**5 1-Bound Slotted RFE for Circuits**

With global slotted RBE, we present a slotted RFE scheme for circuits with adaptive 1-bound SIM security.

#### <span id="page-19-1"></span>**5.1 Construction**

For some circuit family  $C : X \to Z$ , let  $\bigcup(\cdot, \cdot)$  be the universal circuit such that  $\bigcup(C, x) = C(x)$  for any circuit  $C \in C$ and input  $x \in X$ . Assume a garbled circuit scheme GC = (Garble, Eval) where *n* is the input length of the circuit, and a global slotted registered broadcast encryption gsRBE = (Setup, Gen, Ver, Agg, Enc, Dec), then our 1-bound slotted RFE for circuits (set  $Q = 1$  as default) works as follows:

 $-$  Setup $(1^{\lambda},1^L,C)$  : Run gsRBE.Setup $(1^{\lambda},1^L)$   $2n$  times and obtain  $\{crs_{w,b}\}_{w\in[n],b\in\{0,1\}}.$  For all  $w\in[n]$  and  $b\in[n]$  $\{0,1\}$ , run gsRBE.Gen $(\text{crs}_{w,b}, i)$  for all  $i \in [L]$ , omit secret keys and obtain valid public keys  $\{\widehat{\mathsf{pk}}_{i,w,b}\}_{i \in [L], w \in [n], b \in \{0,1\}}$ . Output

$$
\text{crs} = (\{\text{crs}_{w,b}\}_{w \in [n], b \in \{0,1\}}, \{\widehat{\text{pk}}_{i,w,b}\}_{i \in [L], w \in [n], b \in \{0,1\}}).
$$

 $\vdash$  Gen(crs, *i*) : For all  $w \in [n]$  and  $b \in \{0, 1\}$ , run (pk<sub>i wh</sub>, sk<sub>*i,w*,b</sub>) ← gsRBE.Gen(crs<sub>*w*,b</sub>, *i*). Output

 $pk_i = (\{pk_{i,w,b}\}_{w \in [n],b \in \{0,1\}}), \, sk_i = (\{sk_{i,w,b}\}_{w \in [n],b \in \{0,1\}}).$ 

- $-$  Ver(crs, i, pk<sub>i</sub>): For all  $w \in [n]$  and  $b \in \{0, 1\}$ , run  $\beta_{w,b} \leftarrow$  gsRBE.Ver(crs<sub>w,b</sub>, i, pk<sub>i,w,b</sub>), and if  $\beta_{w,b} = 0$ , output 0 and abort. Otherwise, output 1.
- **–** Agg(crs, {pk<sub>i</sub>, C<sub>i</sub>}<sub>i∈[L]</sub>) : Parse C<sub>i</sub> = (C<sub>i</sub>[1], ..., C<sub>i</sub>[n]) ∈ {0, 1}<sup>n</sup> and pk<sub>i</sub> = ({pk<sub>i,w,b</sub>}<sub>w∈[n],b∈{0,1}</sub>). For all *i* ∈ [L] and all  $w \in [n], b \in \{0, 1\}$ , set

$$
\overline{\mathsf{pk}}_{i,w,b} := \begin{cases} \mathsf{pk}_{i,w,b}, & \text{when } \mathsf{C}_i[w] = b, \\ \widehat{\mathsf{pk}}_{i,w,b}, & \text{otherwise.} \end{cases}
$$

Then run (mpk<sub>w,b</sub>, {hsk<sub>j,w,b</sub>}<sub>j∈[L]</sub>) ← gsRBE.Agg(crs<sub>w,b</sub>, {*i*,  $\overline{pk}_{i,w,b}$ }<sub>i∈[L]</sub>). Output

$$
mpk = (\{mpk_{w,b}\}_{w \in [n], b \in \{0,1\}}), \text{ hsk}_j = (C_j, \{hsk_{j,w,b}\}_{w \in [n], b \in \{0,1\}}).
$$

 $-$  Enc(mpk, x) : Let  $U[x]$  be the universal circuit with x hard-wired. Run

$$
(\tilde{\mathsf{U}}, \{\mathsf{lab}_{w,b}\}_{w \in [n], b \in \{0,1\}}) \leftarrow \mathsf{GC}.\mathsf{Garble}(1^\lambda, \mathsf{U}[x]).
$$

For all  $w \in [n], b \in \{0, 1\}$ , run

$$
\mathsf{ct}_{w,b} \leftarrow \mathsf{gsRBE}.\mathsf{Enc}(\mathsf{mpk}_{w,b}, \mathsf{lab}_{w,b}).
$$

Output  $ct = (\tilde{U}, \{ct_{w,b}\}_{w \in [n], b \in \{0,1\}}).$ 

*–* Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) : Parse hsk<sub>i\*</sub> = (C<sub>i\*</sub>, {hsk<sub>i\*,w,b</sub>}<sub>w∈[n],b∈{0,1}</sub>) and sk<sub>i\*</sub> = ({sk<sub>i\*,w,b</sub>}<sub>w∈[n],b∈{0,1}</sub>). For all  $w \in [n]$ , let  $b_w = C_{i^*}[w]$  and run

 $m_{w,b_w} \leftarrow$  gsRBE.Dec(hs $k_{i^*,w,b_w}$ , s $k_{i^*,w,b_w}$ , c $t_{w,b_w}$ ).

Thus, we output

$$
z \leftarrow \mathsf{GC}.\mathsf{Eval}(\tilde{\mathsf{U}}, \{m_{w,b_w}\}_{w \in [n]}).
$$

*Remark.* The above generic construction can be instantiated by any global slotted RBE scheme, but it does not support unbounded users, i.e., crs not scaling with  $L$ . Indeed, if we adopt the construction in Section  $4$ , we can improve it into the one supporting unbounded users by removing all public keys  $\{\overline{pk}_{i,w,b}\}$  in algorithm Setup and directly letting  $\overline{pk}_{i,w,b} = pk_{i,w,b}$  when  $C_i[w] = b$ .

**Completeness and Compactness.** For completeness, it follows the underlying slotted RBE. In other words, if slotted RBE in above construction meets completeness, then it holds that

$$
Pr[gsRBE.Ver(crs_{w,b}, i, pk_i) = 1] = 1
$$

for all  $w \in [n]$  and all  $b \in \{0, 1\}$ . Thus, the completeness of our construction follows readily.

For compactness, thanks to the optimal compactness of gsRBE, our 1-bound slotted RFE scheme has the following properties:

$$
|mpk| = 2n \cdot poly(\lambda, \log L), |hsk_j| = 2n \cdot poly(\lambda, \log L)
$$

where  $n$  is related to circuit family  $C$ . Thus, our construction meets the compactness requirement.

**Correctness.** For all  $\lambda, L \in \mathbb{N}$ , all C, and all  $i^* \in [L]$ , all crs  $\leftarrow$  Setup( $1^{\lambda}, 1^L, C$ ), all  $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Gen}(\mathsf{crs}, i^*)$ , all  $\{pk_i\}_{i\in[L]\setminus\{i^*\}}$  such that Ver(crs, *i*,  $pk_i$ ) = 1, all  $x \in X$  and  $C_1, \ldots, C_L \in C$ , we have  $sk_{i^*} = (\{sk_{i,w,b}\}_{w\in[n],b\in\{0,1\}})$  and

$$
ct = \left(\tilde{U}, \begin{pmatrix} gSRBE.Enc(mpk_{1,0}, lab_{1,0}) \cdots gSRBE.Enc(mpk_{n,0}, lab_{n,0}) \\ gSRBE.Enc(mpk_{1,1}, lab_{1,1}) \cdots gSRBE.Enc(mpk_{n,1}, lab_{n,1})\end{pmatrix}\right).
$$

Here, each mpk $_{w.b}$  is generated from  $\{i,\overline{\mathsf{pk}}_{i,w,b}\}_{i\in [L]}$ . Note that in algorithm Enc, we have

$$
\overline{\mathsf{pk}}_{i,w,b} := \begin{cases} \mathsf{pk}_{i,w,b}, & \text{when } \mathsf{C}_i[w] = b, \\ \widehat{\mathsf{pk}}_{i,w,b}, & \text{otherwise.} \end{cases}
$$

Here, sk $_{i^*,w,b}$  is the secret key of  $\overline{\mathsf{pk}}_{i^*,w,b}$  only when C $_{i^*}[w]=b.$  Thus, after computing all m $_{w,b},$  nothing else can be obtained except for  $\{lab_{i^*,w,b_w}\}_{w\in[n]}$ . Then it follows the correctness of garbled circuits to compute  $z = U(C_{i^*}, x) =$  $C_{i^*}(x)$ . Therefore, the correctness follows readily.

#### **5.2 Security**

<span id="page-20-0"></span>**Theorem 3.** *Assume* GC = (Garble, Eval) *is a secure garbled circuits scheme and* gsRBE = (Setup, Gen, Ver, Agg, Enc, Dec) *is a global slotted RBE scheme with optimal compactness which achieves the IND security defined in Section [2.3,](#page-8-0) our construction achieves the 1-bound SIM security defined in Section [2.4.](#page-9-1)*

*Proof.* let C<sup>\*</sup> be the circuit corresponding to the unique corrupted user. Just as the security analysis presented in Section [1.2,](#page-2-1) our proof strategy follows the security of underlying global slotted RBE and garbled circuits. Concretely, we randomize all labels  $\{[\widetilde{ab}_{w,1-C^*[w]}\}_{w \in [n]}$  one by one (from Game $_{1,\kappa-1}$  to Game $_{1,\kappa}$ ), via the IND security of global slotted RBE. Then we can simulate rest labels (from Game<sub>1,n</sub> to Game<sub>2</sub>) via the security of garbled circuits. In final game, the challenge ciphertext will only disclose  $C^*(x^*)$  and nothing else. Here, we define the simulator  $\widetilde{Enc}$  that works as follows:

**−**  $\widetilde{Enc}(\text{mpk}, (i^*, C^*, C^*(x^*), sk_{i^*}))$ : Parse mpk =  $(\{\text{mpk}_{w,b}\}_{w \in [n], b \in \{0,1\}})$ . Run

ĺ

$$
\tilde{C}^*, \{\widetilde{\mathsf{lab}}_{w,C^*[w]}\}_{w\in[n]}\Big\rangle \leftarrow \widetilde{\mathsf{Garble}}(1^\lambda,C^*(x)).
$$

Then sample  $\widetilde{\mathsf{lab}}_{w,1-C^*[w]} \leftarrow \{0,1\}^\lambda$  for all  $w \in [n]$ . Set

$$
ct^* = \left(\tilde{C}^*, \begin{pmatrix} ct^*_{1,0} \cdots ct^*_{n,0} \\ ct^*_{1,1} \cdots ct^*_{n,1} \end{pmatrix}\right)
$$
  
=  $\left(\tilde{C}^*, \begin{pmatrix} gSRBEEEnc(mpk_{1,0}, l\overline{ab}_{1,0}) \cdots gSRBEEEnc(mpk_{n,0}, l\overline{ab}_{n,0}) \\ gSRBEEEnc(mpk_{1,1}, l\overline{ab}_{1,1}) \cdots gSRBEEEnc(mpk_{n,1}, l\overline{ab}_{n,1})\end{pmatrix}\right).$ 

The algorithm  $\widetilde{\mathsf{Enc}}$  actually does not need sk $_{i^*}$ , so our resulting construction can resist single malicious user.

**Game Sequence.** We prove Theorem [3](#page-20-0) via a sequence of games as follows:

- **–** Game0: this game is identical to the real experiment of adaptive 1-SIM security. Recall that
	- crs has the form:

$$
\text{crs} = (\{\text{crs}_{w,b}\}_{w \in [n], b \in \{0,1\}}, \{\widehat{\text{pk}}_{i,w,b}\}_{i \in [L], w \in [n], b \in \{0,1\}}).
$$

where crs<sub>w,b</sub> and  $\bar{pk}_{i,w,b}$  are generated from the underlying slotted RBE algorithms gsRBE.Setup and gsRBE.Gen, respectively.

• For each  $i \in [L]$ ,  $(\mathsf{pk}_i, \mathsf{sk}_i)$  are in the form:

$$
\mathsf{pk}_{i} = (\{\mathsf{pk}_{i,w,b}\}_{w \in [n], b \in \{0,1\}}), \ \mathsf{sk}_{i} = (\{\mathsf{sk}_{i,w,b}\}_{w \in [n], b \in \{0,1\}}).
$$

where ( $pk_{i,w,b}$ , sk<sub>i,w,b</sub>) are sampled from algorithm gsRBE.Gen.

• The master public key mpk and helper secret key hsk<sub>i</sub> has the form

$$
mpk = \begin{pmatrix} mpk_{1,0} \cdots mpk_{n,0} \\ mpk_{1,1} \cdots mpk_{n,1} \end{pmatrix}, \text{ hsk}_j = \begin{pmatrix} c_j, \begin{pmatrix} hsk_{j,1,0} \cdots hsj_{j,n,0} \\ hsk_{j,1,1} \cdots hsj_{j,n,1} \end{pmatrix} \end{pmatrix},
$$

where for each  $w \in [n], b \in \{0, 1\}$ , mpk<sub>w,b</sub> is computed as

$$
\overline{\mathsf{pk}}_{i,w,b} := \begin{cases} \mathsf{pk}_{i,w,b}, & \text{when } \mathsf{C}_i[w] = b \\ \widehat{\mathsf{pk}}_{i,w,b}, & \text{otherwise.} \end{cases}
$$

Then obtain  $(\text{mpk}_{w,b}, {\{\text{hsk}_{j,w,b}\}}_{j \in [L]}) \leftarrow \text{gsRBE.Agg}(crs_{w,b}, {i, \overline{\text{pk}}_{i,w,b}\}}_{i \in [L]})$ .

• The challenge ciphertext ct<sup>∗</sup> has the form

$$
ct^* = \left(\tilde{U}, \begin{pmatrix} ct_{1,0}^* \cdots ct_{n,0}^* \\ ct_{1,1}^* \cdots ct_{n,1}^* \end{pmatrix}\right)
$$
  
=  $\left(\tilde{U}, \begin{pmatrix} gSRBE.Enc(mpk_{1,0}, lab_{1,0}) \cdots gSRBE.Enc(mpk_{n,0}, lab_{n,0}) \\ gSRBE.Enc(mpk_{1,1}, lab_{1,1}) \cdots gSRBE.Enc(mpk_{n,1}, lab_{n,1})\end{pmatrix}\right),$ 

where  $(\tilde{\mathsf{U}}, \{\mathsf{lab}_{w,b}\}_{w \in [n], b \in \{0,1\}}) \leftarrow \mathsf{GC}.\mathsf{Garble}(1^\lambda, \mathsf{U}[x^*]).$ 

– Game $_{1,\kappa}$ ( $\kappa\in[n]$ ): Game $_{1,\kappa}$  is identical to Game $_0$  except that for each  $w\leq\kappa,$  set  $b_w$  = C $^*[w]$  and  $\overline{b}_w$  = 1 − C $^*[w]$ , we have

$$
\begin{aligned}\n\mathsf{ct}_{w,b_{w}}^{*} &\leftarrow \mathsf{gsRBE}.\mathsf{Enc}(\mathsf{mpk}_{w,b_{w}}, s_{w,b_{w}}, \mathsf{lab}_{w,b_{w}}), \\
\mathsf{ct}_{w,\overline{b}_{w}}^{*} &\leftarrow \mathsf{gsRBE}.\mathsf{Enc}(\mathsf{mpk}_{w,\overline{b}_{w}}, s_{w,\overline{b}_{w}}, \widetilde{\mathsf{lab}}_{w,\overline{b}_{w}}),\n\end{aligned}
$$

where  $\widetilde{{\rm lab}}_{w,\overline{b}_w}$  is randomly sampled from  $\{0,1\}^\lambda.$ 

 $-$  Game<sub>2</sub>: this game is identical to Game<sub>1,n</sub> except that it replaces Enc with  $\widetilde{Enc}$  to generate the challenge ciphertext ct<sup>∗</sup> .

**Lemma 7 (**Game1,−<sup>1</sup> ≈ Game1,**).** *For any efficient adversary* A*, there exists an algorithm* B<sup>1</sup> *with close running time to* A *such that*

$$
|{\mathsf{Adv}}^{1,\kappa-1}_{\mathcal{A}}(\lambda)-{\mathsf{Adv}}^{1,\kappa}_{\mathcal{A}}(\lambda)|\leq 2\cdot {\mathsf{Adv}}^{\mathsf{gSRBE}}_{\mathcal{B}_1}(\lambda)+{\mathsf{negl}}(\lambda).
$$

*Proof.* Initially,  $\mathcal{B}_1$  receives upper bound L from A. Then  $\mathcal{B}_1$  flips a coin  $\beta \leftarrow \{0,1\}$  and sends L to the challenger of gsRBE. Then  $\mathcal{B}_1$  proceeds following phases:

Setup. After receiving the crs<sub>k, *ß*</sub> from the challenger of gsRBE, initialize 2n – 1 slotted RBE instances by itself, and obtain  $\{crs_{w,b}\}_{w\in[n],b\in\{0,1\}}$ . For all  $w \in [n]$  and all  $b \in \{0,1\}$ , if  $w = \kappa$  and  $b = \beta$ , query the oracle OGen $(i)$  of gsRBE to obtain  $\widehat{\mathsf{pk}}_{i,\kappa,\beta}$  for all  $i \in [L]$ ; otherwise, run gsRBE.Gen(crs $_{w,b}, i$ ) by itself for all  $i \in [L]$ . Then omit all secret keys and obtain public keys  $\{\widehat{\mathsf{pk}}_{i,w,b}\}_{i \in [L], w \in [\![ n ]\!], b \in \{0,1\}}.$  Output

$$
\text{crs} = \big(\{\text{crs}_{w,b}\}_{w \in [n], b \in \{0,1\}}, \{\widehat{\text{pk}}_{i,w,b}\}_{i \in [L], w \in [n], b \in \{0,1\}}\big).
$$

**Query.** Here, A can query oracles as below:

- $−$  OGen(i) : For all  $w ∈ [n]$  and all  $b ∈ {0, 1}$ , if  $w = \kappa$  and  $b = \beta$ , query the oracle OGen(i) of gsRBE to obtain  $pk_{i,k,\beta}$ ; otherwise, run  $(pk_{i,w,b}, sk_{i,w,b}) \leftarrow$  gsRBE.Gen(crs<sub>w,b</sub>, i) by itself. Output pk =  $({pk}_{i,w,b} \}_{w \in [n],b \in \{0,1\}})$ and set  $\mathcal{D}_i[pk] = \{sk_{i,w,b}\}_{w \in [n] \setminus \{k\}, b \in \{0,1\}} \cup \{sk_{i,k,1-\beta}\}.$
- **–** OCor(*i*, pk) : Parse pk = ({pk<sub>*i,w,b*</sub>}<sub>*w*∈[*n*],*b*∈{0,1}</sub>). Query the oracle OCor(*i*, pk<sub>*k,β*</sub>) of gsRBE to obtain sk<sub>*i,k,β*</sub>. Then return  $\mathcal{D}_i[pk] \cup \{sk_{i,k,\beta}\}\)$ . Update  $C = C \cup \{(i,pk)\}\$ .
- **Challenge.**  $B_1$  receives challenge public keys  $\{pk_i^*\}_{i \in [L]}$  where  $pk_i^* = \{pk_{i,w,b}^*\}_{w \in [n], b \in \{0,1\}}$ . Combining with challenge circuits  $\{C_i^*\}_{i\in[L]}$ , we assume that the unique corrupted user registering the circuit  $C^*\in\{C_i^*\}_{i\in[L]}$ , and C<sup>\*</sup> is linked to the public key  $pk^* \in \{pk_i^*\}_{i \in [L]}$ . If  $C^*[k] = \beta$ , then abort the experiment immediately; otherwise, it means that all public keys that has registered in gsRBE are not corrupted. Thus, for all  $i \in [L]$  and all  $w \in [n], b \in \{0, 1\}$ , set

$$
\overline{\mathsf{pk}}_{i,w,b} := \begin{cases} \mathsf{pk}^*_{i,w,b}, & \text{when } \mathsf{C}_i[w] = b, \\ \widehat{\mathsf{pk}}_{i,w,b}, & \text{otherwise.} \end{cases}
$$

After that, run  $(\tilde{\mathsf{U}}, \{\mathsf{lab}_{w,b}\}_{w\in[n], b\in\{0,1\}}) \leftarrow \mathsf{GC}.\mathsf{Garble}(1^\lambda, \mathsf{U}[x^*])$  and for each  $w \leq \kappa$ , pick  $\widetilde{\mathsf{lab}}_{w,1-\mathsf{C}^*[w]} \leftarrow \{0,1\}^\lambda.$ For all  $w \in [n], b \in \{0,1\}$ , if  $w = \kappa$  and  $b = \beta$ , send  $(\{i, \overline{\mathsf{pk}}_{i,\kappa,\beta}\}_{i \in [L]}, \mathsf{lab}_{\kappa,\beta}, \widetilde{\mathsf{lab}}_{\kappa,\beta})$  to the challenger, and obtain  $(\text{mpk}_{\kappa,\beta}, \{\text{hsk}_{j,\kappa,\beta}\}_{j\in[L]})$ ; otherwise, run  $(\text{mpk}_{w,b}, \{\text{hsk}_{j,w,b}\}_{j\in[L]}) \leftarrow \text{gsRBE.Agg}(crs_{w,b}, \{i, \text{pk}_{i,w,b}\}_{i\in[L]})$ . Set

$$
mpk = (\{mpk_{w,b}\}_{w \in [n], b \in \{0,1\}}), \; hsk_j = (C_j^*, \{hsk_{j,w,b}\}_{w \in [n], b \in \{0,1\}}).
$$

Then  $\mathcal{B}_1$  receives the challenge ciphertext ct $_{\kappa,\beta}^*$  and computes other ct $_{w.b}^*$  as follows:

$$
\mathsf{ct}_{w,b}^* = \begin{cases} \mathsf{gsRBE}.\mathsf{Enc}(\mathsf{mpk}_{w,b}, \widetilde{\mathsf{lab}}_{w,b}), \text{ when } w < \kappa \wedge \mathsf{C}^*[w] = 1 - b, \\ \mathsf{gsRBE}.\mathsf{Enc}(\mathsf{mpk}_{w,b}, \mathsf{lab}_{w,b}), \text{ otherwise.} \end{cases}
$$

Finally, return the challenge ciphertext

$$
\mathsf{ct}^* = \left(\tilde{\mathsf{U}}, \begin{pmatrix} \mathsf{ct}_{1,0}^* & \cdots & \mathsf{ct}_{n,0}^* \\ \mathsf{ct}_{1,1}^* & \cdots & \mathsf{ct}_{n,1}^* \end{pmatrix}\right).
$$

<span id="page-23-2"></span>Observe that if ct ${}^*_{\kappa,\beta}$  is generated under message lab ${}_{\kappa,\beta},$   $\mathcal{B}_1$  simulates Game $_{1,\kappa}$ ; otherwise, it simulates Game $_{1,\kappa-1}.$ Thus, this readily proves the lemma. ⊓⊔

**Lemma 8 (**Game<sub>1,n</sub>  $\approx_c$  Game<sub>2</sub>). *For any efficient adversary A*, there exists an algorithm  $B_2$  with close running time *to* A *such that*

$$
|{\mathsf{Adv}}^{1,n}_{\mathcal{A}}(\lambda)-{\mathsf{Adv}}^2_{\mathcal{A}}(\lambda)|\leq {\mathsf{Adv}}^{\mathsf{GC}}_{{\mathcal{B}}_2}(\lambda)+{\mathsf{negl}}(\lambda).
$$

*Proof.* The only difference between Game<sub>1,n</sub> and Game<sub>2</sub> is the generation of ct<sup>∗</sup>. Obviously, if the underlying garbled circuits scheme is secure, the output of algorithm Enc in Game $_{1,n}$  is indistinguishable from the output of algorithm  $\widehat{\textsf{Enc}}$  in Game<sub>2</sub> from the view of A. □

**Analysis for Honest and Corrupt Cases.** Without loss of generality, we assume that all challenge circuits are different from each other. Let  $C^*$  be the corrupted circuit for slot  $i^*$ . For some honest slot  $i$ , there must exist at least a different bit (assume index  $w'$ ) between  $C^*_i$  and  $C^*$ . In this way, even if  $\cal H$  owns the secret key of slot  $i^*$ , it will only obtain  $\{\widetilde{\mathsf{lab}}_{w,C_i^*[w]}\}_{w \in [n] \setminus \{w'\}} \cup \{\widetilde{\mathsf{lab}}_{w',C_i^*[w']}\}.$  Here,  $\{\widetilde{\mathsf{lab}}_{w,C_i^*[w]}\}_{w \in [n] \setminus \{w'\}}$  are simulated using  $1^\lambda$  and  $C^*(x^*),$ while lab $_{w',C^*_l[w']}$  is a random string independent from other labels. According to the privacy of garbled circuits,  ${\cal A}$ cannot extract any useful information about  $C_i^*(x^*)$ .

For corrupt case, A can only obtain  $\left(\tilde{\sf C}^*, \{\widetilde{\sf lab}_{w, {\sf C}^*[w]}\}_{w\in[n]}\right)$  that are simulated using just  $1^\lambda$  and  ${\sf C}^*(x^*).$  Thus, only  $C^*(x^*)$  is revealed. At last, the proof of Theorem [3](#page-20-0) is completed.  $□$ 

# <span id="page-23-0"></span>**6 Q-Bound Slotted RFE for Circuits**

In this section, we roughly follow the approach of [\[GVW12\]](#page-28-9) in order to upgrade the construction in Section [5](#page-19-0) from 1 bound security to Q-bound security. Here, we only present an Q-bound secure RFE for NC1 circuits because it can be trivially bootstrapped into RFE for all polynomial-sized circuits using computational randomized encodings [\[AIK06\]](#page-27-19). With slotted RFE for circuits, we can convert it into a full-fledged RFE construction via "power-of-two" approach (presented in the full version).

#### **6.1 Construction**

Let  $C := NC1$  be a circuit family with circuits of degree D and size n. Our construction is additionally parameterized with S, N, t and v. For any circuit C  $\in C$  and set  $\Delta \subseteq [S]$ , we define a new circuit G  $\in G$  takes as input  $\mathbf{x} = (x, r_1, \ldots, r_S)$ as follows:

<span id="page-23-1"></span>
$$
G(\mathbf{x}) = C(x) + \sum_{j \in \Delta} r_j.
$$
 (1)

Assume a 1-bound slotted registered function encryption osRFE = (Setup, Gen, Ver, Agg, Enc, Dec), our Q-bound slotted RFE for circuit family  $C$  works as follows:

–  ${\sf Setup}(1^{\lambda},1^L,1^Q,C)$  : Initialize  $N$  1-bound slotted RFE instances by running osRFE.Setup( $1^{\lambda},1^L,G$ ) for  $N$  times, and obtain  $\{crs_k\}_{k\in[N]}$ . Output

$$
crs = (\{crs_k\}_{k \in [N]}).
$$

**−** Gen(crs, *i*) : Parse crs = ({crs<sub>*k*</sub>} $_{k \in [N]}$ ), then it proceeds as follows:

• Sample uniformly random set  $\Gamma_i \subseteq [N]$  of size  $tD + 1$  and random set  $\Delta_i \subseteq [S]$  of size  $v$ , where  $\Delta_i$  can be translated into a bit string  $\delta_i$  of length v log S. Set  $n' = n + v \log S$ ;

<span id="page-24-0"></span>• For all  $k \in [N]$ , run  $(\mathsf{pk}_{k,i}, \mathsf{sk}_{k,i}) \leftarrow \mathsf{osRFE.Gen}(\mathsf{crs}_k, i)$ . Here, based on the construction presented in Section [5,](#page-19-0) for each  $k \in [N]$ , we have  $pk_{k,i} = (\{pk_{k,i,w,b}\}_{w \in [n'],b \in \{0,1\}})$  and  $sk_{k,i} = (\{sk_{k,i,w,b}\}_{w \in [n'],b \in \{0,1\}})$ , where  $(pk_{k,i,w,b}, sk_{k,i,w,b})$  are sampled from the key-generation algorithm of global slotted RBE. Then omit secret keys sk<sub>k,i</sub> for all  $k \notin \Gamma_i$ , and for all  $k \in \Gamma_i$ , update

$$
sk_{k,i} = (\{sk_{k,i,w,b}\}_{w \le n} \cup \{sk_{k,i,w,\delta_i[w]}\}_{w > n}).
$$

Output

$$
pk_i = (\{pk_{k,i}\}_{k \in [N]})
$$
,  $sk_i = (\{sk_{k,i}\}_{k \in \Gamma_i}, \Gamma_i, \Delta_i)$ .

- $-$  Ver(crs,  $i, p k_i$ ) : For all  $k \in [N]$ , run  $\beta_k \leftarrow$  osRFE.Ver(crs $_k$ ,  $i, p k_{k,i}$ ), and if  $\beta_k = 0$ , output 0 and abort; otherwise, output 1.
- **–** Agg(crs, {pk<sub>i</sub>, C<sub>i</sub>}<sub>i∈[L]</sub>) : Parse pk<sub>i</sub> = ({pk<sub>k,i</sub>}<sub>k∈[N]</sub>) for each  $i \in [L]$ . For all  $k \in [N]$ , run (mpk<sub>k</sub>, {hsk<sub>k,j</sub>}<sub>j∈[L]</sub>) ← osRFE.Agg(crs<sub>k</sub>, {pk<sub>k,i</sub>, G<sub>i</sub>}<sub>i∈[L]</sub>), where G<sub>i</sub> is defined as in [\(1\)](#page-23-1) given constant C<sub>i</sub> and Δ<sub>i</sub>. Here, algorithm osRFE.Agg works as in construction [5.1](#page-19-1) except that all submitted public keys associated with  $\Delta_i$  are directly registered into  ${mpk_k}_{k\in[N]}$  to ensure the cover freeness. Then output

$$
mpk = (\{mpk_k\}_{k \in [N]}), bsk_j = (\{bsk_{k,j}\}_{k \in [N]})
$$

- $-$  Enc(mpk, x) : It proceeds as follows:
	- Sample a random degree *t* polynomial  $\mu(\cdot)$  whose constant term is *x*;
	- For all  $j \in [S]$ , sample a random degree Dt polynomial  $\zeta_i(\cdot)$  whose constant term is 0;
	- For all  $k \in [N]$ , compute  $\mathbf{x}_k = (\mu(k), \zeta_1(k), \ldots, \zeta_s(k))$  and run  $\mathsf{ct}_k \leftarrow \mathsf{osRFE}.\mathsf{Enc}(\mathsf{mpk}_k, \mathbf{x}_k).$ Then output

$$
ct = (ct_1, \ldots, ct_N).
$$

 $-$  Dec(hsk<sub>i\*</sub>, sk<sub>i\*</sub>, ct) : Parse hsk<sub>i\*</sub> = ({hsk<sub>k,i\*</sub>}<sub>k∈[N]</sub>), sk<sub>i\*</sub> = ({sk<sub>k,i\*</sub>}<sub>k∈Γ<sub>i\*</sub>, Γ<sub>i\*</sub>, Δ<sub>i\*</sub>). For all  $k \in Γ$ <sub>i\*</sub>, run</sub>

$$
z_k \leftarrow \text{osRFE.Dec}(hsk_{k,i^*}, sk_{k,i^*}, ct_k).
$$

Then use  $\{z_k\}_{k\in\Gamma_{l^*}}$  to recover a degree Dt polynomial  $\psi(\cdot)$  such that  $\psi(k)=z_k$ . Output  $\psi(0)$ .

*Remark.* For the range of parameters S, N, t and v, we let  $v = O(\lambda)$ . Suppose there are Q corrupted users whose slot numbers are collected in set  $\{c_1,\ldots,c_O\}\subseteq[L]$ , then we set  $t=O(Q^2\lambda)$  and  $N=O(D^2Q^2t)$  to ensure small pairwise intersections [\[GVW12\]](#page-28-9). In other words, for all  $\Gamma_{c_1}, \ldots, \Gamma_{c_O}$ , it holds that

$$
\Pr\left[\left|\bigcup_{i\neq j}(\Gamma_{c_i}\cap\Gamma_{c_j})\right|>t\right]\leq \text{negl}(\lambda).
$$

Besides, we set  $S = O(Q^2v)$  to ensure cover freeness [\[GVW12\]](#page-28-9). In other words, for all  $\Delta_{c_1},\ldots,\Delta_{c_O}$  and all  $i\in [Q],$  it holds that

$$
\Pr\left[\Delta_{c_i}\setminus\left(\bigcup_{j\neq i}\Delta_{c_j}\right)=\emptyset\right]\leq \mathsf{negl}(\lambda).
$$

 « ¬ On the other hand, we can trivially bootstrap above construction into RFE for all polynomial-sized circuits instead of NC1 circuits. For any polynomial-sized circuit  $C_i$ , this can be done by modifying the definition of circuit  $G_i$  into generating a randomized encoding of  ${\sf C}_i$  that is computable by a constant-degree circuit with fresh randomness generated from weak pseudo-random function. Here, we omit more details and only discuss the NC1 case.

<span id="page-25-1"></span>**Completeness and Compactness.** For completeness, it follows the underlying 1-bound slotted RFE. For compactness, combining the compactness analysis of osRFE, it holds that

$$
|mpk| = N \cdot 2n \cdot poly(\lambda, \log L), |hsk_j| = N \cdot 2n \cdot poly(\lambda, \log L)
$$

where N depends on the corruption bound  $Q$ , and n is related to circuit family C. Thus, our construction meets the compactness requirement.

**Correctness.** By the correctness of underlying 1-bound slotted RFE, for all  $k \in \Gamma_{i^*}$  we have

$$
\psi(k) = G_{i^*}(\mathbf{x}_k) = C_{i^*}(\mu(k)) + \sum_{j \in \Delta_{i^*}} \zeta_j(k).
$$

Since  $|\Gamma_{i^*}| = Dt + 1$ , we can recover the polynomial  $\psi(\cdot)$  of degree Dt, and then evaluate  $\psi(0) = C_{i^*}(\mu(0)) = C_{i^*}(x)$ . Indeed, above computation exactly corresponds to BGW MPC protocol [\[WOG88\]](#page-29-5). Therefore, the correctness follows readily.

#### **6.2 Security**

<span id="page-25-0"></span>**Theorem 4.** *Assume* osRFE = (Setup, Gen, Ver, Agg, Enc, Dec) *is a slotted RFE scheme which achieves the 1-bound SIM security, the above construction achieves the Q-bound SIM security.*

*Proof.* Our proof strategy is analogous to [\[GVW12\]](#page-28-9). Suppose the adversary  $\mathcal A$  colludes with  $Q$  corrupted users whose slot indices are collected in set  $\{c_1,\ldots,c_Q\}\subseteq[L]$ . Let  $\cal T$  denote the set  $\bigcup_{i\neq j}(\Gamma_{c_i}\cap\Gamma_{c_j})$ , and note that  $\cal A$  has no idea about other  $\Gamma_i \notin \{\Gamma_{c_1},\ldots,\Gamma_{c_0}\}\$ under honest slots. With challenge ciphertext  $ct^* = (ct^*_1,\ldots,ct^*_N)$  and secret key  $sk_i = (\{sk_{k,i}\}_{k \in \Gamma_i}, \Gamma_i, \Delta_i)$  for  $k \in [N]$ , then the proof strategy is organized as follows:

- $-$  If  $k \notin \mathcal{T}$ , it means there exists at most a set  $\Gamma_{c_i}$  such that  $k \in \Gamma_{c_i}$  and  $k \notin \Gamma_{c_j}$  for other *j*. In particular,  $ct_k^*$  can be just decrypted correctly by a corrupted user with slot  $i$ . Thus, it can rely on the 1-bound SIM security of underlying construction;
- Otherwise, it means that A holds more than one secret keys that are used to decrypt ct $^*_k$  correctly. In this way, 1-bound SIM security would be broken, and the security would in turn rely on the underlying MPC protocol. In this case,  $\cal{A}$  will obtain no valid information about the challenge message  $x^*$  as long as small pairwise intersections and cover freeness hold.

Then we define the simulator  $\widetilde{\mathsf{Enc}}$  that works as follows:

- $\widetilde{\mathsf{Enc}}(\mathsf{mpk}, \mathcal{K})$ : Parse  $\mathcal{K} = \{(c_i, C^*_{c_i}, C^*_{c_i}(x^*), \mathsf{sk}_{c_i})\}_{i \in [Q]}$ . Here, we obtain  $\Gamma_{c_1}, \ldots, \Gamma_{c_Q}, \Delta_{c_1}, \ldots, \Delta_{c_Q}$  from  $\mathsf{sk}_i$ . Then it proceeds as follows:
	- Sample a uniformly random degree *t* polynomial  $\mu(\cdot)$  whose constant term is 0;
	- $\bullet$  For all  $j\in [Q]$ , fix some  $a_j^*\in \Delta_{c_j}\setminus \left(\bigcup_{j\neq k}\Delta_{c_k}\right)$  based on the cover freeness. For other  $a\in (\Delta_{c_1}\cup\dots\cup\Delta_{c_Q})\setminus\{0\}$  $\{a_j^*\}_{j\in [Q]}$ , sample a uniformly random degree  $Dt$  polynomial  $\zeta_a(\cdot)$  whose constant term is 0. For all  $j\in [Q]$ , pick a random degree  $Dt$  polynomial  $\psi_{c_j}(\cdot)$  whose constant term is  ${\sf C}_{c_j}(x^*)$  and adjust the evaluation of  $\zeta_{a_j^*}$ such that for all  $k \in \mathcal{T}$ , we have

$$
\psi_{c_j}(k) = C_{c_j}(\mu(k)) + \sum_{a \in \Delta_{c_j}} \zeta_a(k).
$$

• For all  $k \notin \mathcal{T}$ , suppose there is at most a set  $\Gamma_{c_i}$  such that  $k \in \Gamma_{c_i}$  and  $k \notin \Gamma_{c_i}$  for all  $j \neq i$ , then we simulate  $ct_k^*$  as follows:

$$
\mathsf{ct}_k^* \leftarrow \mathsf{osRFE}.\widetilde{\mathsf{Enc}}(\mathsf{mpk}_k, (c_i, \mathsf{G}_{c_i}, \psi_{c_i}(k), \mathsf{sk}_{c_i})).
$$

• For all  $k \in \mathcal{T}$ , we generate  $\mathsf{ct}_{k}^{*}$  as in real experiment:

$$
\mathsf{ct}_k^* \leftarrow \mathsf{osRFE}.\mathsf{Enc}(\mathsf{mpk}_k, (\mu(k), \zeta_1(k), \ldots, \zeta_s(k))).
$$

Finally, output

$$
ct^* = (ct_1^*, \ldots, ct_N^*).
$$

**Game Sequence.** We prove Theorem [4](#page-25-0) via a sequence of games as follows:

- **–** Game<sub>0</sub>: Real Game.
- **–** Game<sub>1</sub>: this game is identical to Game<sub>0</sub> except that it samples  $\zeta_1, \ldots, \zeta_n \psi_1, \ldots, \psi_Q$  as in Enc and simulates all  $\{\mathsf{ct}_k^*\}_{k\notin\mathcal{T}}$  as in algorithm Enc.
- $-$  Game<sub>2</sub>: this game is identical to Game<sub>1</sub> except that it replaces Enc with  $\widetilde{Enc}$  to generate the challenge ciphertext ct∗ .

**Lemma 9 (**Game<sub>0</sub>  $\approx_c$  Game<sub>1</sub>). *For any adversary* A, there exists an algorithm B with close running time to A such *that*

$$
|\mathsf{Adv}_{\mathcal{A}}^0(\lambda)-\mathsf{Adv}_{\mathcal{A}}^1(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{osRFE}}(\lambda)+\mathsf{negl}(\lambda).
$$

*Proof.* Informally, thanks to cover freeness, we observe that the distribution of  $\zeta_1, \ldots, \zeta_s, \psi_1, \ldots, \psi_Q$  in  $\widetilde{Enc}$  are essentially identical to those in Enc, and then we can follow the 1-bound SIM security of underlying slotted RFE scheme to simulate all  $\{ct_k^*\}_{k\notin \mathcal{T}}$  as in Enc. Thus, it holds that Game<sub>0</sub> is computationally indistinguishable from Game1. ⊓⊔

**Lemma 10 (Game<sub>1</sub>**  $\approx$ <sub>s</sub> Game<sub>2</sub>). *For any adversary*  $A$ *, we have* 

$$
|\mathsf{Adv}^1_{\mathcal{A}}(\lambda)-\mathsf{Adv}^2_{\mathcal{A}}(\lambda)|\leq \mathsf{negl}(\lambda).
$$

*Proof.* The only difference between Game<sub>1</sub> and Game<sub>2</sub> is the distributions of  $\mu$ . We claim that the distributions of  $\{\mu(k)\}_{k \in \mathcal{T}}$  in Game<sub>1</sub> are essentially identical to those in Game<sub>2</sub> as long as small pairwise intersections holds, i.e.,  $|\mathcal{T}| \leq t$ . Thus, this readily proves the lemma. □

**Acknowledgements.** Thank Yin Zhu for his useful advice. We also thank all anonymous reviewers for their helpful comments. This work was supported in part by National Natural Science Foundation of China (62372180, 62325209, U2336203), Innovation Program of Shanghai Municipal Education Commission (2021-01-07-00-08-E00101), the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (22510750100), the Australian Research Council Discovery Early Career Researcher Award DE240100282, the Major Program (JD) of Hubei Province (No. 2023BAA027), and the Key Research and Development Science and Technology of Hainan Province (GHYF2022010).

#### **References**

- <span id="page-26-1"></span>ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, December 2019. [8](#page-7-1)
- <span id="page-26-0"></span>ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, April 2019. [8](#page-7-1)
- <span id="page-27-15"></span>ACF+20. Shweta Agrawal, Michael Clear, Ophir Frieder, Sanjam Garg, Adam O'Neill, and Justin Thaler. Ad hoc multi-input functional encryption. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 40:1–40:41. LIPIcs, January 2020. [8](#page-7-1)
- <span id="page-27-5"></span>AFNV19. Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 701–731. Springer, Heidelberg, August 2019. [2](#page-1-1)
- <span id="page-27-18"></span>AGT21. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 224–255. Springer, Heidelberg, November 2021. [8](#page-7-1)
- <span id="page-27-19"></span>AIK06. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *computational complexity*, 15(2):115–162, 2006. [24](#page-23-2)
- <span id="page-27-9"></span>AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. [2](#page-1-1)
- <span id="page-27-7"></span>ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016. [2](#page-1-1)
- <span id="page-27-12"></span>AMVY21. Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from LWE. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 239–269, Virtual Event, August 2021. Springer, Heidelberg. [7](#page-6-0)
- <span id="page-27-8"></span>AR17. Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205. Springer, Heidelberg, November 2017. [2](#page-1-1)
- <span id="page-27-17"></span>ATY23. Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multi-input fe (and more) for attribute-weighted sums. In *Annual International Cryptology Conference*, pages 464–497. Springer, 2023. [8](#page-7-1)
- <span id="page-27-6"></span>AV19. Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, December 2019. [2,](#page-1-1) [7](#page-6-0)
- <span id="page-27-4"></span>AW17. Shashank Agrawal and David J. Wu. Functional encryption: Deterministic to randomized functions from simple assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 30–61. Springer, Heidelberg, April / May 2017. [2](#page-1-1)
- <span id="page-27-11"></span>BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. [4,](#page-3-0) [8](#page-7-1)
- <span id="page-27-3"></span>BLM+24. Pedro Branco, Russell WF Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy KY Woo. Traitor tracing without trusted authority from registered functional encryption. *Cryptology ePrint Archive*, 2024. [2,](#page-1-1) [3,](#page-2-2) [6,](#page-5-1) [7](#page-6-0)
- <span id="page-27-1"></span>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. [1,](#page-0-0) [2](#page-1-1)
- <span id="page-27-10"></span>BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015. [2](#page-1-1)
- <span id="page-27-14"></span>CDG+18. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multiclient functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018. [8](#page-7-1)
- <span id="page-27-16"></span>CDSG+20. Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 747–775. Springer, Heidelberg, August 2020. [8](#page-7-1)
- <span id="page-27-13"></span>CGJS15. Nishanth Chandran, Vipul Goyal, Aayush Jain, and Amit Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive, Report 2015/1017, 2015. <https://eprint.iacr.org/2015/1017>. [8](#page-7-1)
- <span id="page-27-2"></span>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. [2](#page-1-1)
- <span id="page-27-0"></span>DP23. Pratish Datta and Tapas Pal. Registration-based functional encryption. *IACR Cryptol. ePrint Arch.*, 2023:457, 2023. [1,](#page-0-0) [2,](#page-1-1) [3,](#page-2-2) [34](#page-33-0)
- <span id="page-28-15"></span>EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. [8](#page-7-1)
- <span id="page-28-0"></span>FFM+23. Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 98–133. Springer, 2023. [1,](#page-0-0) [2,](#page-1-1) [3,](#page-2-2) [7,](#page-6-0) [34](#page-33-0)
- <span id="page-28-5"></span>FWW23. Cody Freitag, Brent Waters, and David J Wu. How to use (plain) witness encryption: Registered abe, flexible broadcast, and more. In *Annual International Cryptology Conference*, pages 498–531. Springer, 2023. [2,](#page-1-1) [4,](#page-3-0) [6,](#page-5-1) [7,](#page-6-0) [16,](#page-15-2) [17,](#page-16-0) [19,](#page-18-0) [31](#page-30-0)
- <span id="page-28-13"></span>GGL24. Rachit Garg, Rishab Goyal, and George Lu. Dynamic collusion functional encryption and multi-authority attributebased encryption. In *IACR International Conference on Public-Key Cryptography*, pages 69–104. Springer, 2024. [7](#page-6-0)
- <span id="page-28-12"></span>GGLW22. Rachit Garg, Rishab Goyal, George Lu, and Brent Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 736–763. Springer, Heidelberg, May / June 2022. [7](#page-6-0)
- <span id="page-28-17"></span>GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013. [32](#page-31-0)
- <span id="page-28-1"></span>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. [1](#page-0-0)
- <span id="page-28-3"></span>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. [2](#page-1-1)
- <span id="page-28-2"></span>GV20. Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 621–651. Springer, Heidelberg, August 2020. [2](#page-1-1)
- <span id="page-28-9"></span>GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. [2,](#page-1-1) [4,](#page-3-0) [7,](#page-6-0) [24,](#page-23-2) [25,](#page-24-0) [26](#page-25-1)
- <span id="page-28-11"></span>GW09. Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, Heidelberg, April 2009. [6,](#page-5-1) [16](#page-15-2)
- <span id="page-28-4"></span>HLWW23. Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 511–542. Springer, Heidelberg, April 2023. [2,](#page-1-1) [3,](#page-2-2) [6,](#page-5-1) [7,](#page-6-0) [32,](#page-31-0) [34](#page-33-0)
- <span id="page-28-7"></span>JLLW23. Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In *Annual International Cryptology Conference*, pages 233–262. Springer, 2023. [2](#page-1-1)
- <span id="page-28-10"></span>KMW23. Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 407–441. Springer, 2023. [6](#page-5-1)
- <span id="page-28-6"></span>KW17. Sam Kim and David J. Wu. Access control encryption for general policies from standard assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 471–501. Springer, Heidelberg, December 2017. [2](#page-1-1)
- <span id="page-28-14"></span>QWW18. Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018. [7](#page-6-0)
- <span id="page-28-8"></span>SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. [2,](#page-1-1) [4](#page-3-0)
- <span id="page-28-16"></span>Tsa22. Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Heidelberg, August 2022. [17](#page-16-0)
- <span id="page-29-4"></span>VWW22. Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Heidelberg, December 2022. [17](#page-16-0)
- <span id="page-29-2"></span>Wee22. Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Heidelberg, May / June 2022. [3,](#page-2-2) [16](#page-15-2)
- <span id="page-29-5"></span>WOG88. Avi Wigderson, MB Or, and S Goldwasser. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC'88)*, pages 1–10, 1988. [26](#page-25-1)
- <span id="page-29-3"></span>Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986. [4,](#page-3-0) [8](#page-7-1)
- <span id="page-29-1"></span>ZLZ+24. Ziqi Zhu, Jiangtao Li, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered functional encryptions from pairings. In *EUROCRYPT*, 2024. [2,](#page-1-1) [3](#page-2-2)
- <span id="page-29-0"></span>ZZGQ23. Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered abe via predicate encodings. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 66–97. Springer, 2023. [2,](#page-1-1) [4,](#page-3-0) [6,](#page-5-1)

[7](#page-6-0)

# **Appendices**

# <span id="page-30-0"></span>**A Other Cryptographic Tools**

#### **A.1 Public-Key Encryption**

**Algorithms.** A public-key encryption (PKE for short) scheme consists of three efficient algorithms as follows:

- $-$  Setup $(1^\lambda)\to$  (pk, sk). It takes as input security parameter  $1^\lambda$ , outputs public key pk and secret key sk.
- $-$  Enc(pk, m;  $r)\to$  ct. It takes as input public key pk and a message m. Then sample randomness  $r\gets\{0,1\}^\lambda$  to generate a ciphertext ct. Output ct;
- **–** Dec(sk, ct) → m/⊥. It takes as input secret key sk and ciphertext ct, outputs a message m or an empty symbol ⊥.

**Correctness.** For all  $\lambda$  and all m, we have

$$
Pr[Dec(sk, Enc(pk, m)) = m | (pk, sk) \leftarrow Setup(1^{\lambda})] = 1.
$$

**Semantic Security.** For all  $\lambda$  and all efficient adversaries  $\mathcal{A}$ , the advantage

$$
\begin{bmatrix}\n\mathbf{p} \\
\mathbf{p} \\
\mathbf{p}' = b\n\end{bmatrix}\n\begin{bmatrix}\n(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda}) \\
(\mathsf{m}_{0}^{*}, \mathsf{m}_{1}^{*}) \leftarrow \mathcal{A}(1^{\lambda}, \mathsf{pk}) \\
b \leftarrow \{0, 1\}; \mathsf{ct}^{*} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_{b}^{*})\n\end{bmatrix}\n-\frac{1}{2}\n\begin{bmatrix}\n\mathsf{pk} \\
\mathsf{pk} \\
\
$$

is negligible in  $\lambda$ .

#### **A.2 Function-Binding Hash Function**

**Algorithms.** A function-binding hash function [\[FWW23\]](#page-28-5) (FBH for short) consists of four efficient algorithms as follows:

- $-$  Setup $(1^\lambda,n)\to$  hk. It takes as input security parameter  $1^\lambda$  and the bound of input block number  $n$ , then outputs a hash key hk where  $|hk| = poly(\lambda, m_{in}, \log n)$ . Here,  $m_{in} = m_{in}(\lambda)$  is the size of the input block, and set digest size  $d = d(\lambda)$  and opening size  $p = p(\lambda)$ .
- $-$  SetupBind $(1^\lambda,n,f)\to$  hk. It takes as input security parameter  $1^\lambda$ , the number of input blocks  $n$  and a function  $f:(\{0,1\}^{m_{\rm in}})^*\to\{0,1\}^{m_{\rm out}},$  then outputs a hash key hk. Here,  $m_{\rm out}=m_{\rm out}(\lambda)$  is the size of function output and f is implementable by a circuit of size at most  $s(\lambda) \cdot \text{poly}(k)$ .
- **–** Hash(hk, (blk1, . . . , blk)) → dig. It takes as input the hash key hk and a sequence of input blocks (blk1, . . . , blk) ∈  $(\{0,1\}^{m_{\rm in}})^k$  where  $k\leq n,$  then deterministically outputs a digest dig  $\in\{0,1\}^d.$
- $-$  Open(hk, (blk<sub>1</sub>, ..., blk<sub>k</sub>), S)  $\to \pi$ . It takes as input the hash key hk, a sequence of input blocks (blk<sub>1</sub>, ..., blk<sub>n</sub>) and a subset  $S \subseteq [k]$ , then outputs an opening proof  $\pi \in \{0,1\}^p$ .
- $-$  Ver(hk, dig, S,  $\{(i, \text{blk}_i)\}_{i \in S}$ , π)  $\rightarrow$  0/1. It takes as input hk, dig, S,  $\{(i, \text{blk}_i)\}_{i \in S}$  and π, then outputs a bit indicating whether it accepts or rejects.

**Efficiency.** For all  $\lambda$ , all  $n$ ,  $k$  satisfying  $k \leq n$ , all  $f$ , all  $hk \leftarrow$  Setup $(1^{\lambda}, n)$ , all tuple of inputs  $(blk_1, \ldots, blk_k) \in$  $(\{0,1\}^{m_{\rm in}})^k$ , all dig  $\leftarrow$  Hash $(\mathsf{hk}, (\mathsf{blk}_1, \dots, \mathsf{blk}_k)),$  all subset  $S \subseteq [k]$  and all opening  $\mathsf{proof}\pi \leftarrow \mathsf{Open}(\mathsf{hk}, (\mathsf{blk}_1, \dots, \mathsf{blk}_k), S),$ it is required that

Time(Setup(1<sup>$$
\lambda
$$</sup>, n)) = Time(SetupBind(1 <sup>$\lambda$</sup> , n, f)) = poly( $\lambda$ , s( $\lambda$ ), log n),  
|dig| = poly( $\lambda$ , m<sub>out</sub>( $\lambda$ ), log k), and | $\pi$ | = |S| · poly( $\lambda$ , m<sub>out</sub>( $\lambda$ ), log k).

<span id="page-31-0"></span>**Perfect Completeness.** For all  $\lambda$ , all  $n, k$  satisfying  $k \leq n$ , and all input blocks  $(\text{blk}_1, \ldots, \text{blk}_k) \in (\{0, 1\}^{m_{\text{in}}})^k$  and subset  $S \subseteq [k]$ , we have

$$
\Pr\left[\text{Ver(hk, dig, S, {(i, blki)}i \in S, \pi) = 1 \left| \text{dif } \left( \frac{hk}{\sigma} \right) \text{Hash}(hk, (blk_1, ..., blk_k)) \right| = 1.
$$
\n
$$
\pi \leftarrow \text{Open}(hk, (blk_1, ..., blk_k), S)
$$

**Computational Function Hiding.** For all efficient adversaries A, it is required that the advantage

 $\sim$ 

$$
\Pr\left[b' = b \begin{bmatrix} f \leftarrow \mathcal{A}(1^{\lambda}) \\ h k_0 \leftarrow \text{Setup}(1^{\lambda}, n) \\ h k_1 \leftarrow \text{SetupBind}(1^{\lambda}, n, f) \\ b \leftarrow \{0, 1\}; b' \leftarrow \mathcal{A}(h k_b) \end{bmatrix} - \frac{1}{2} \right\right]
$$

is negligible in  $\lambda$ .

**Statistical Function Binding.** For all stateful unbounded adversaries  $A$ , it is required that the advantage

$$
\Pr\left[\begin{array}{c}(\exists\{(j, \text{blk}_j^*)\}_{j\in[k]\setminus S},\\ \text{s.t. } f(\text{blk}_1^*, \ldots, \text{blk}_k^*) = f(\text{blk}_1, \ldots, \text{blk}_k)\end{array}\middle|\begin{array}{c}f \leftarrow \mathcal{A}(1^{\lambda})\\ \text{hk} \leftarrow \text{SetupBind}(1^{\lambda}, n, f)\\(\{\text{blk}_t\}_{i\in[k]}, S, \{(i, \text{blk}_t^*)\}_{i\in S}, \pi) \leftarrow \mathcal{A}(\text{hk})\end{array}\right]\right]
$$
\n
$$
\wedge (\text{Ver}(\text{hk}, \text{dig}, S, \{(i, \text{blk}_t^*)\}_{i\in S}, \pi) = 1) \left[\begin{array}{c}f \leftarrow \mathcal{A}(1^{\lambda})\\(\{\text{blk}_t\}_{i\in[k]}, S, \{(i, \text{blk}_t^*)\}_{i\in S}, \pi) \leftarrow \mathcal{A}(\text{hk})\end{array}\right]
$$

Í

۔<br>.is negligible in λ

#### **A.3 Witness Encryption**

**Algorithms.** A witness encryption [\[GGSW13\]](#page-28-17) (WE for short) for a NP language  $\mathcal L$  with witness relation  $\mathcal R: X\times W\to Y$ {0, 1} consists of four efficient algorithms as follows:

- $-$  Enc $(1^\lambda,x,$  m $)\to$  ct. It takes as input the security parameter  $1^\lambda$ , an instance  $x\in X$  for the language  ${\cal L}$ , and a message m, then outputs a ciphertext ct.
- $–$  Dec(ct,  $w$ )  $→$  m/ $⊥$ . It takes as input a ciphertext ct and a witness  $w ∈ W$ , then outputs a message m or an empty symbol ⊥.

**Correctness.** For all  $\lambda \in \mathbb{N}$ , all  $(x, w)$  such that x is a valid instance of L (namely  $\mathcal{R}(x, w) = 1$ ) and all messages m, we have

$$
Pr[Dec(Enc(1λ, x, m), w) = m] = 1.
$$

**Security.** For all  $\lambda \in \mathbb{N}$  and all efficient adversaries  $\mathcal{A}$ , the security requires the advantage

$$
\Pr\left[b'=b\left| \begin{array}{l} (x^*, m_0^*, m_1^*) \leftarrow \mathcal{A}(1^{\lambda}) \\ b \leftarrow \{0, 1\}; \text{ct}^* \leftarrow \text{Enc}(1^{\lambda}, x^*, m_b^*) \end{array} \right| - \frac{1}{2} \right| \right]
$$

is negligible in  $\lambda$  when  $x^* \in X$  is an invalid instance of language  $L.$ 

# **B Q-Bound RFE for Circuits**

We exploit the "power-of-two" approach from [\[HLWW23\]](#page-28-4) to generically convert  $Q$ -bound slotted RFE to  $Q$ -bound RFE.

**Construction.** Suppose a full-fledged RFE mostly supports  $L = 2^{\ell}$  users and endures  $Q \ll L$  corrupted users, this approach needs  $\ell$  + 1 copies of slotted RFE with 1, 2, 4,  $\cdots$  ,  $2^\ell$  slots. And the public state aux =  $(\mathcal{D}_1, \mathcal{D}_2,$  mpk) consists of the following terms:

- $\mathcal{D}_1[k,i] = (\mathsf{pk}, y)$ : where  $k \in [0,\ell]$  and  $i \in [2^k]$ . This dictionary assigns a user's  $(\mathsf{pk}, y)$  to the slot  $i$  of the  $2<sup>k</sup>$ -slotted RFE scheme.
- $\mathcal{D}_2[k,n]$  = hsk: where  $k\in[0,\ell]$  and  $n\in[L]$ . This dictionary assigns a hsk of slotted RFE to the  $2^k$ -slotted RFE scheme and the user index  $n$ .
- $-$  mpk = (ctr, mpk<sub>0</sub>,  $\cdots$  , mpk $_{\ell}$ ) denotes the current master public key. Where  $\{m$ pk $_{k}\}_{k\in[0,\ell]}$  denote the master public keys of  $\ell + 1$  copies of slotted RFE, and ctr denotes the number of currently registered users. When no registered user, we initially set mpk =  $(0, \perp, \cdots, \perp)$ .

When no registered user, we initially set aux =  $(\emptyset, \emptyset, \perp)$ . Assuming a Q-bound slotted RFE for circuits sRFE = (Setup, Gen, Ver, Agg, Enc, Dec), a full-fledged RFE for circuits can be constructed as follows:

 $-$  Setup $(1^{\lambda}, 1^L, 1^Q, C)$  : Compute  $\ell = \log L$ . For all  $k \in [0, \ell]$ , run crs $_k \leftarrow$  sRFE.Setup $(1^{\lambda}, 1^{2^k}, 1^{\min(Q, 2^k)}, C)$ . Output

$$
crs = (crs_0, \cdots, crs_\ell).
$$

 $-$  Gen(crs, aux) : Fetch crs =  $\{crs_k\}_{k\in[0,\ell]}$  and aux =  $(\mathcal{D}_1,\mathcal{D}_2,$  mpk), where mpk =  $(\textsf{ctr}, \{ \textsf{mpk}_k\}_{k\in[0,\ell]}).$  For all  $k \in [0, \ell]$ , compute

$$
i_k = (\text{ctr mod } 2^k) + 1
$$

and run  $(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{sRFE}$ . Gen $(\mathsf{crs}_k, i_k)$ . Set ctr' = ctr and output

$$
pk = (ctr', pk_0, \dots, pk_\ell)
$$
 and  $sk = (ctr', sk_0, \dots, sk_\ell)$ .

- $-$  Reg(crs, aux, pk, C) : Fetch crs =  $\{crs_k\}_{k\in[0,\ell]}$ , aux =  $(D_1, D_2,$  mpk), and pk =  $(\textsf{ctr}', \{ \textsf{pk}_k\}_{k\in[0,\ell]})$ , where mpk = (ctr,  ${mpk_k}_{k\in[0,\ell]}$ ). For all  $k\in[0,\ell]$ , do the following operates:
	- Compute  $i_k = (\text{ctr mod } 2^k) + 1;$
	- Check if sRFE.Ver(crs<sub>k</sub>,  $i_k$ ,  $pk_k$ ) = 1 and ctr' = ctr. If the check passes, set ctr = ctr + 1, if the check fails,the algorithm halts and output (mpk, aux);
	- Update  $\mathcal{D}_1[k, i_k] = (pk, C);$
	- If  $i_k = 2^k$ : compute  $(\text{mpk}'_k, \{\text{hsk}_{k,j}\}_{j \in [2^k]}) \leftarrow \text{sRFE.Agg}(crs_k, \{\mathcal{D}_1[k,i]\}_{i \in [2^k]})$ . Update mpk $_k = \text{mpk}'_k$ , and for all  $j \in [2^k]$ , update  $\mathcal{D}_2[k,\text{ctr} - 2^k + j] = \text{hsk}_{k,j}$ .

Update the master public key mpk = (ctr, mpk $_0, \cdots$  , mpk $_\ell$ ) and aux =  $(\mathcal{D}_1, \mathcal{D}_2,$  mpk), output (mpk, aux).

 $-$  Upd(crs, aux, pk): Fetch crs = {crs<sub>k</sub>}<sub>k∈[0,ℓ]</sub>, aux =  $(D_1, D_2,$  mpk), and pk =  $(\text{ctr}', \{pk_k\}_{k \in [0,\ell]}),$  where mpk = (ctr,  ${mpk_k}_{k \in [0,\ell]}$ ). Output

$$
\mathsf{hsk} = \begin{cases} \underbrace{(\mathcal{D}_2[0,\text{ctr}'+1],\cdots,\mathcal{D}_2[\ell,\text{ctr}'+1]}_{\mathsf{hsk}_{\ell}}) & \text{if } \text{ctr} < \text{ctr} \\ \bot & \text{otherwise} \end{cases} \tag{2}
$$

 $-$  Enc(mpk, x): Fetch mpk = (ctr, {mpk<sub>k</sub>}<sub>k $\in [0,\ell]$ </sub>). For all  $k \in [0,\ell]$ , compute:

J.

$$
ct_k = \begin{cases} sRFE.Enc(mpk_k, x) & \text{if } mpk_k \neq \bot \\ \bot & \text{otherwise} \end{cases}
$$
 (3)

Output  $ct = (ctr, ct_0, \cdots, ct_\ell).$ 

- <span id="page-33-0"></span>– Dec(hsk, sk, ct): Fetch sk =  $(\text{ctr}', \{ \text{sk}_k \}_{k \in [0,\ell]})$ , hsk =  $\{ \text{hsk}_k \}_{k \in [0,\ell]}$  and ct =  $(\text{ctr}, \{ \text{ct}_k \}_{k \in [0,\ell]})$ . Proceed as follows:
	- If  $ctr' \geq ctr$ : output  $\perp$ ;
	- Otherwise, compute ctr =  $(a_\ell, \dots, a_0)_2$  and ctr' =  $(b_\ell, \dots, b_0)_2$ . We denote  $k_d$  as the maximum  $k \in [0, \ell]$ such that  $a_k \neq b_k$ . if mpk $k_{k_d} \neq \bot$  and hsk $k_{k_d} = \bot$ , output getupd;
	- Otherwise, output sRFE.Dec( $hsk_{k_d}$ , sk $k_d$ , ct).

**Analysis.** We would demonstrate the correctness, compactness, efficiency and security of the above construction via a series of theorems.

**Theorem 5 (Correctness).** *Suppose construction* sRFE *is complete and perfectly correct. Then our construction is perfectly correct.*

*Proof.* Similar to [\[HLWW23](#page-28-4)[,FFM](#page-28-0)+23[,DP23\]](#page-27-0), we omit details here. ⊓⊔

**Theorem 6 (Compactness).** *Suppose construction* sRFE *is compact. Then our construction is compact.*

*Proof.* Observe that  $|mpk| = |ctr| + \sum_{i\in [0,\ell]} |mpk_i|$  and  $|hsk| = \sum_{i\in [0,\ell]} |hsk_i|$ , where ctr is a  $\ell$ -bit number. According to the compactness of sRFE, we have  $|mpk_i| = poly(\lambda, C, Q, \log L)$  and  $|hsk_i| = poly(\lambda, C, Q, \log L)$  for all  $i \in [0, \ell]$ . Then it holds that  $|mpk| = poly(\lambda, C, Q, \log i)$  and  $|hsk| = poly(\lambda, C, Q, \log |\mathcal{R}|)$ . □

**Theorem 7 (Update Efficiency).** *Suppose construction* sRFE *is compact. Then our construction meets update efficiency.*

*Proof.* Observe that the number of invocations of Upd is at most  $\ell + 1 = O(\log |\mathcal{R}|)$  and Upd is only invoked when one of  $\{h s k_k\}_{k \in [0,\ell]}$  is ⊥. Thus, the number of invocations of Upd in ODec is at most  $O(\log |\mathcal{R}|)$ .

On the other hand,  $|h s k_k| = poly(\lambda, C, Q, \log |\mathcal{R}|)$  for  $k \in [0, \ell]$  according to the compactness of sRFE. Since aux maintains a dictionary  $\mathcal{D}_2$  mapping each index slot index k to its set of helper decryption keys, each invocation of Upd runs in poly(log |R|) time (in RAM model). □

**Theorem 8.** *Suppose construction* sRFE *meets the Q-bound SIM security. Then our construction meets the Q-bound SIM security.*

*Proof.* Analogous to [\[HLWW23\]](#page-28-4), suppose that there exists an adversary  $\mathcal A$  who breaks the  $Q$ -bound SIM security of sRFE with non-negligible advantage, we start by defining a sequence of hybrid experiments, each parameterized by an index  $v \in [0, \ell]$ :

**–** H : This game is identical to the real experiment defined in Section [2.5,](#page-10-0) except that for the challenge ciphertext  $ct^* = (ctr^*, ct^*_0, \ldots, ct^*_\ell)$ , the first v ciphertexts  $\{ct^*_k\}_{k \in [0,\nu]}$  are simulated by  $\widetilde{Enc}(mpk_k, K)$ , while remaining ciphertexts  $\{$ ct $^*_k\}_{k\in$ [ $v+1,\ell]}$  are generated by Enc(mpk $_k,x^*$ ). Here,  $\cal K$  is the set of all corrupted slot information in RFE. Concretely, this game proceeds as follows:

**Setup.** In this phase,  $A$  proceeds as follows:

- $\mathcal A$  chooses  $(L, Q)$  and send them to the challenger;
- The challenger samples crs  $\leftarrow$  Setup $(1^{\lambda}, 1^{\lambda}, 1^{\lambda}, 0)$ ;
- Then it initializes a counter  $t = 0$ , master public key mpk = $\bot$  and auxiliary input aux =  $(\mathcal{D}_1, \mathcal{D}_2,$  mpk) where  $\mathcal{D}_1 = \emptyset$ ,  $\mathcal{D}_2 = \emptyset$  and mpk =  $(0, \perp, \ldots, \perp)$ ; Set  $\mathcal{R} = \emptyset$ ,  $\mathcal{C} = \emptyset$ ,  $\mathcal{S} = \emptyset$  and a dictionary  $\mathcal{D}$  with  $\mathcal{D}[pk] = \emptyset$  for all possible pk;
- Finally, it sends crs to  $\mathcal{A}$ .

**Query.** In the query phase,  $A$  can make queries as follows:

- ORegHK(C):  $\mathcal A$  specifies a circuit C. The challenger sets  $t \leftarrow t+1$  and samples  $(\mathsf{pk}_t, \mathsf{sk}_t) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{aux})$ and  $(\textsf{mpk}',\textsf{aux}') \leftarrow \textsf{Reg}(\textsf{crs},\textsf{aux},\textsf{pk}_t,\textsf{C}).$  Then it updates mpk = mpk $',\textsf{aux} = \textsf{aux}',\mathcal{D}[\textsf{pk}_t] = \mathcal{D}[\textsf{pk}_t] \cup \mathcal{D}[\textsf{pk}_t]$  $\{C\}$ , append  $(\mathsf{pk}_t, \mathsf{sk}_t)$  to  $\mathcal R$  and return  $(t, \mathsf{mpk}, \mathsf{aux}, \mathsf{pk}_t)$ ;
- OCorHK(*i*): A specifies an index  $i \in [t]$ . let  $\mathcal{R}[i] = (pk, sk)$  and  $C = \mathcal{D}[pk]$ , append pk to C and return sk. This oracle can be queried at most  $Q$  times.

**Challenge.** In the challenge phase,  $\mathcal{A}$  submits the challenge message  $x^*$ . The challenger proceeds as follows:

- Let aux = (ctr,  $\mathcal{D}_1$ ,  $\mathcal{D}_2$ , mpk) where mpk = (ctr, mpk<sub>0</sub>, . . . , mpk<sub>ℓ</sub>);
- For each  $k \in [0, \ell]$ , if mpk $_k = \bot$ , then set ctr $_k = \bot$ ; otherwise, if  $k \leq \nu$ , compute  $ct_k^* \leftarrow \widetilde{Enc}(mpk_k, \mathcal{K})$ , and if  $k > v$ , compute  $ct_k^* \leftarrow Enc(mpk_k, x^*)$ ;
- The challenger replies to  $\mathcal{A}$  with  $ct^* = (ctr, ct_0, \ldots, ct_\ell)$ .

**Lemma 11** (H<sub> $\nu-1$  ≈ c H<sub>ν</sub>). Suppose construction SREF meets the Q-bound SIM security, For any adversary A, there*exists an algorithm* B *with close running time to* A *such that*

$$
|\mathsf{Adv}_{\mathcal{A}}^{\nu-1}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{\nu}(\lambda)|\leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{sRFE}}(\lambda)+\mathsf{negl}(\lambda).
$$

*Proof.* The algorithm  $\mathcal B$  will simulated  $H_{\nu-1}$  or  $H_{\nu}$  as follows:

**Setup.** In this phase,  $B$  proceeds as follows:

- When  $B$  receives  $(L, Q)$  from  $\mathcal{A}$ , it sends  $(2^{\nu}, \min(Q, 2^{\nu}))$  to the challenger of  $2^{\nu}$ -slotted RFE, then obtains  $crs_v;$
- $-$  For each  $k \in [0, l] \setminus \{v\}$ , the challenger samples crs $_k \leftarrow$  Setup $(1^{\lambda}, 1^{2^k}, 1^{\min(Q, 2^{\nu})}, C);$
- Then it initializes a counter  $t = 0$ , master public key mpk =⊥ and auxiliary input aux =  $(D_1, D_2,$  mpk) where  $\mathcal{D}_1 = \emptyset$ ,  $\mathcal{D}_2 = \emptyset$  and mpk =  $(0, \perp, \ldots, \perp)$ ; Set a dictionary  $\mathcal D$  with  $\mathcal D[pk] = \emptyset$  for all possible pk. In addition,  $B$  maintains a dictionary  $\mathcal{D}_s$  to track the secret keys associated with each ORegHK query, and initializes two ordered list  $S_1 = \perp, S_2 = (\perp, \ldots, \perp)$  to track the public keys and circuits aggregated as part of mpk $_{v}$ .
- **–** Finally, it sends  $\text{crs} = (\text{crs}_0, \ldots, \text{crs}_\ell)$  to  $\mathcal{A}$ .
- **Query.** In the query phase,  $B$  can simulated queries  $A$  makes as follows:
	- **–** OReqHK(C): When  $\mathcal{A}$  specifies a circuit C, let ctr be the current counter in aux. For each  $k \in [0, \ell]$ , compute  $i_k = (\text{ctr mod } 2^k) + 1$ . Then generate  $(\text{pk}_k, \text{sk}_k) \leftarrow \text{sRFE.Gen}(\text{crs}_k, i_k)$  for  $k \neq v$ . Next,  $\mathcal B$  makes a query  $i_v$  to OGen $(\cdot)$ , then obtain a public key pk<sub>v</sub>. It set pk = (ctr, pk<sub>0</sub>, . . . , pk<sub>ℓ</sub>). The challenger sets  $t \leftarrow t + 1$ and  $\mathcal{D}_s[t] = (\text{ctr}, \{ \text{sk}_k \}_{k \in [0,\ell] \setminus \{v\}})$ . Next,  $\mathcal B$  runs  $(\text{mpk}', \text{aux}') \leftarrow \text{Reg}(\text{crs}, \text{aux}, \text{pk}_t, \text{C})$  and updates mpk = mpk', aux = aux',  $\mathcal{D}[pk_t] = \mathcal{D}[pk_t] \cup \{C\}$ , append  $(pk_t, sk_t)$  to  $\mathcal R$  and return  $(t, mpk, aux, pk_t)$ . In addition,  $\mathcal B$  updates  $\mathcal S_2[i_v]=(t,\mathsf C).$  Moreover, if  $i_v=2^v,$   $\mathcal B$  set  $\mathcal S_1=\mathcal S_2.$
	- **–** OCorHK(*i*): When  $\mathcal{A}$  specifies an index  $i \in [t]$ ,  $\mathcal{B}$  looks up  $\mathcal{D}_s[i] = (\text{ctr}, {\{sk_k\}_{k \in [0,\ell] \setminus \{v\}}})$ . let  $\mathcal{R}[i] = (\text{pk}, \text{sk})$ , then it makes a corrupt query (*i*, pk) to OCor( $\cdot$ , $\cdot$ ) and obtains sk<sub>v</sub>. Next, append pk to C, set  $\mathcal{K} = \mathcal{K} \cup$  $\{(i, C, C(x^*), sk)\}\$ and return sk<sub>v</sub> = (ctr, sk<sub>0</sub>, . . . , sk<sub>l</sub>). This oracle can be queried at most Q times.
- **Challenge.** In the challenge phase,  $\mathcal A$  submits the challenge message  $x^*$ . Let mpk = (ctr, mpk<sub>0</sub>, ..., mpk<sub>ℓ</sub>) be the current master public key. For each  $k \in [0, \ell]$ ,  $\mathcal{B}$  proceeds as follows:
	- **−** If mpk<sub>k</sub> =⊥, then set  $ct_k^* = \bot$ ;
	- **−** If mpk<sub>k</sub>  $\neq \perp$  and  $k < v$ , then compute  $ct_k^*$  ←  $\widetilde{Enc}(mpk_k, \mathcal{K})$ ;
	- If mpk<sub>k</sub> ≠⊥ and  $k = v$ ,  $\hat{B}$  makes a challenge query to obtain ct $_v^*$ ;
	- **−** If mpk<sub>k</sub> ≠⊥ and  $k > v$ , then compute  $ct_k^*$  ← Enc(mpk<sub>k</sub>,  $x^*$ );
	- **−** The challenger replies to  $\mathcal{A}$  with  $ct^* = (ctr, ct^*_0, ..., ct^*_{\ell}).$

In above experiment,  $B$  perfectly simulates an execution of Q-bound RFE against  $A$ . Note that if 2 $^{\nu} \leq Q$ , it means that all secret keys in z<sup>v</sup>-slotted RFE can be corrupted. This will not influence the simulation of ct<sub>v</sub>, since other corrupt secret keys are not registered in  $z^{\nu}$ -slotted RFE, which means that adversary still owns at most  $2^{\nu}$  corrupted keys and then the simulation security of z<sup>v</sup>-slotted RFE will not be broken. Thus, in the case where ct $^*_\nu\leftarrow$  $s$ RFE.Enc(mpk<sub>v</sub>, x<sup>\*</sup>),  $\mathcal B$  simulates H<sub>v-1</sub>, while in the case where  $ct_v^* \leftarrow s$ RFE. $\widetilde{Enc}(mpk_v, \mathcal{K})$ ,  $\mathcal B$  simulates H<sub>v</sub>.