General Circuit Realizing Compact Revocable Attribute-Based Encryption from Multilinear Maps

Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay(✉)

Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur 721302, India
{pratishdatta,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. This paper demonstrates *new technique* for managing *revocation* in the context of *attribute-based encryption* (ABE) and presents two *selectively* secure *directly revocable* ABE (RABE) constructions

– supporting decryption policies realizable by *polynomial size* Boolean circuits of *arbitrary fan-out* and
– featuring *compactness* in the sense that the number of revocation controlling components in ciphertexts and decryption keys are *constant*.

In fact, our RABE schemes are the *first* to achieve these parameters. Both our constructions utilize *multilinear maps*. The size of public parameter in our first construction is *linear* to the maximum number of users supported by the system while in the second construction we reduce it to *logarithmic*.

Keywords: RABE for circuits · Polynomial size circuits · Multilinear map

1 Introduction

In recent times, the cost effectiveness and greater flexibility of cloud technology has triggered an emerging trend among individuals and organizations to outsource potentially sensitive private data to the "cloud", an external large and powerful server. Attribute-based encryption (ABE), a noble paradigm for public key encryption in which ciphertexts are encrypted for entities possessing specific decryption credentials, has been extensively deployed to realize complex access control functionalities in cloud environment. ABE comes in two flavors, namely, key-policy and ciphertext-policy. However, in spite of its promising properties, the adoption of ABE in cloud management requires further refinements.

A crucial feature of ABE systems is the *expressiveness* of the supported decryption policies. Recently, few independent seminal works [4,10] have extended the class of admissible policies for ABE to arbitrary polynomial size Boolean circuits of unbounded fan-out in contrast to circuits of fan-out one realized by all ABE constructions prior to their works.

DOI: 10.1007/978-3-319-23318-5_19

The other significant requirement in the context of ABE is user *revocation*, a tool for *changing* the users' decryption rights. Over time many users' private keys might get compromised, users might leave or be dismissed due to the revealing of malicious activities. In the literature several revocation mechanisms have been proposed in ABE setting [1–3,11,14–16]. The *direct* revocation technique [1,2,14,15], that controls revocation by specifying a revocation list directly during encryption, does not involve any additional proxy server [16] or key update phase [1,3,11]. Consequently, the non-revoked users remain unaffected and revocation can take effect instantly without requiring to wait for the expiration of the current time period.

However, in all the above *revocable* ABE (RABE) systems the decryption policies were restricted to circuits of fan-out one, paving the way for a "backtracking" attack [10] on the policy circuits by unauthorized users, thereby completely breaking the confidentiality of ciphertexts. Further, all currently available standard model RABE constructions supporting direct revocation mode [1,2,14] essentially follow the tree-based revocation mechanism of Naor et al. [12], as a result of which the number of components for managing user revocation contained in the ciphertexts and decryption keys are respectively $O(\widehat{r} \log \frac{N_{\max}}{\widehat{r}})$ and $O(\log N_{\max})$, where $N_{\max}$ is the maximum number of users supported by the system and $\widehat{r}$ is the number of revoked users.

Our Contribution: In this paper, we apply the revocation technique introduced in [5] and its improved variant [6] in the ABE setting and propose two RABE schemes for *general circuit* realizable decryption policies supporting *direct* revocation and featuring *constant* number of components for enforcing revocation in the ciphertexts and decryption keys.

More precisely, we integrate the revocation strategy of [5,6] with the ABE scheme of [10]. As an outcome, we develop the *first* RABE constructions that support the *most expressive* form of decryption policies achieved so far for ABE, namely, arbitrary polynomial size circuits having unbounded fan-out with bounded depth and input length. Although the basic conception may sound simple, its exact realization involves many subtleties that we address with innovative ideas. Our schemes employ multilinear map for which some approximate candidates have recently been proposed [7,8,10]. Both our schemes support direct revocation and are proven secure in the *selective revocation list model* under the *Multilinear Diffie-Hellman Exponent* [4] and the *Compressed Multilinear Diffie-Hellman Exponent* assumptions [13], which are multilinear equivalents of the *Bilinear Diffie-Hellman Exponent* assumption. Our security analyses do not use random oracles or generic multilinear group framework. We emphasize that selective security can be a reasonable trade-off for performance in some circumstances. Moreover, applying a standard *complexity leveraging* argument, as in [4], our selectively secure constructions can be made *adaptively* secure.

Our first RABE scheme, which is a blend of the revocation technique of [5] and an improved version of the ABE construction proposed in [10], has ciphertext consisting of only 3 group elements (or encodings). The decryption keys comprise of $\ell + 4q + 1$ group elements in the worst case, ℓ and q being the input length and number of gates in the policy circuits. This is the same as all currently available vanilla ABE constructions for general circuits based on multilinear maps [4,10]. Consequently, we achieve very short ciphertext size without imposing any extra overhead on the decryption key for the added revocation functionality. To the best of our knowledge, our work is the *first* to achieve this property.

However, the number of group elements in the public parameters in our first RABE construction is linear to $N_{\max}$. In order to overcome this bottleneck, we modify our first construction by replacing the revocation method with that of [6] taking advantage of a multilinear map of (possibly) slightly higher multilinearity level compared to the one used in the first scheme. We reduce the number of group elements in the public parameters to $\log N_{\max}$ in our second RABE scheme. This is comparable with the previous standard model RABE constructions supporting direct revocation [1,2,14]. However, we retain the same property for ciphertext and decryption keys, i.e., the number of ciphertext and decryption key components do not grow with $N_{\max}$.

Finally, while both our RABE schemes are of *key-policy* variety, using the notion of *universal circuits*, as in [10], both our constructions can be extended to realize *ciphertext-policy* style RABE for arbitrary bounded size circuits achieving the same parameters.

2 Preliminaries

■ **Circuit Notation**: We adopt the same notations for circuits as in [10]. First note that without loss of generality we can consider only those circuits which are *monotone*, where gates are either OR or AND having fan-in two, and *layered* (see [10] for details). Our circuits will have a single output gate. A circuit will be represented as a six-tuple $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$. Here, ℓ, q respectively denote the length of the input, the number of gates, and d represents the depth of the circuit which is one plus the length of the shortest path from the output wire to any input wire. We designate the set of input wires as $\mathsf{Input} = \{1, \ldots, \ell\}$, the set of gates as $\mathsf{Gates} = \{\ell + 1, \ldots, \ell + q\}$, the total set of wires in the circuit as $W = \mathsf{Input} \cup \mathsf{Gates} = \{1, \ldots, \ell + q\}$, and the wire $\ell + q$ as the output wire. Let $\mathbb{A}, \mathbb{B} : \mathsf{Gates} \to W \backslash \{\ell + q\}$ be functions. For all $w \in \mathsf{Gates}$, $\mathbb{A}(w)$ and $\mathbb{B}(w)$ respectively identify w's first and second incoming wires. Finally, $\mathsf{GateType} : \mathsf{Gates} \to \{\mathsf{AND}, \mathsf{OR}\}$ defines a functions that identifies a gate as either an AND or an OR gate. We follow the convention that $w > \mathbb{B}(w) > \mathbb{A}(w)$ for any $w \in \mathsf{Gates}$.

We also define a function $\mathsf{depth} : W \to \{1, \ldots, d\}$ such that if $w \in \mathsf{Input}$, $\mathsf{depth}(w) = 1$, and in general $\mathsf{depth}(w)$ of wire w is equal to one plus the length of the shortest path from w to an input wire. Since our circuit is layered, we have, for all $w \in \mathsf{Gates}$, if $\mathsf{depth}(w) = t$ then $\mathsf{depth}(\mathbb{A}(w)) = \mathsf{depth}(\mathbb{B}(w)) = t - 1$.

We will abuse notation and let $f(x)$ be the evaluation of the circuit f on input $x \in \{0, 1\}^\ell$, and $f_w(x)$ be the value of wire w of the circuit f on input x.

2.1 The Notion of RABE for General Circuits

■ **Syntax of RABE for circuits**: Consider a circuit family $\mathbb{F}_{\ell,d}$ that consists of all circuits f with input length ℓ and depth d characterizing decryption rights. A (key-policy) revocable attribute-based encryption (RABE) scheme for circuits in $\mathbb{F}_{\ell,d}$ with message space $\mathbb{M}$ consists of the following algorithms:

$\mathbf{RABE.Setup}(1^\lambda, \ell, d, N_{\max})$: The trusted key generation center takes as input a security parameter 1^λ, the length ℓ of Boolean inputs to decryption circuits, the allowed depth d of the decryption circuits, and the maximum number $N_{\max}$ of users supported by the system. It publishes the public parameters PP along with the empty user list $\mathsf{UL} = \varnothing$, while keeps the master secret key MK to itself.

$\mathbf{RABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, \mathsf{UL}, \mathsf{ID}, f)$: On input the public parameters PP, the master secret key MK, the current user list UL, and a user identity ID together with the decryption policy circuit description $f \in \mathbb{F}_{\ell,d}$ of that user, the key generation center provides a decryption key $\mathsf{SK}_{f,\mathsf{ID}}$ to the user and publishes the user list UL updated with the information of the newly joined user.

$\mathbf{RABE.Encrypt}(\mathsf{PP}, \mathsf{UL}, x, \mathsf{RL}, M)$: Taking in the public parameters PP, the current user list UL, a descriptor input string $x \in \{0,1\}^\ell$, a set of revoked user identities RL, and a message $M \in \mathbb{M}$, the encrypter prepares a ciphertext $\mathsf{CT}_{x,\mathsf{RL}}$.

$\mathbf{RABE.Decrypt}(\mathsf{PP}, \mathsf{UL}, \mathsf{CT}_{x,\mathsf{RL}}, \mathsf{SK}_{f,\mathsf{ID}})$: A user takes as input the public parameters PP, the current user list UL, a ciphertext $\mathsf{CT}_{x,\mathsf{RL}}$ encrypted for x along with a list RL of revoked user identities, and its decryption key $\mathsf{SK}_{f,\mathsf{ID}}$ corresponding to its decryption policy circuit $f \in \mathbb{F}_{\ell,d}$ as well as user identity ID. It attempts to decrypt the ciphertext and outputs the message $M \in \mathbb{M}$ if successful; otherwise, it outputs the distinguished symbol $\bot$.

■ **Correctness:** The correctness of RABE for general circuits is defined as follows: For all $(\mathsf{PP}, \mathsf{UL}, \mathsf{MK}) \leftarrow \mathsf{RABE.Setup}(1^\lambda, \ell, d, N_{\max})$, $\mathsf{SK}_{f,\mathsf{ID}} \leftarrow \mathsf{RABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, \mathsf{UL}, \mathsf{ID}, f)$ for any ID and $f \in \mathbb{F}_{\ell,d}$, $\mathsf{CT}_{x,\mathsf{RL}} \leftarrow \mathsf{RABE.Encrypt}(\mathsf{PP}, \mathsf{UL}, x, \mathsf{RL}, M)$ for any $x \in \{0,1\}^\ell, \mathsf{RL}$ and $M \in \mathbb{M}$, $\big([f(x) = 1] \wedge [\mathsf{ID} \notin \mathsf{RL}] \implies \mathsf{RABE.Decrypt}(\mathsf{PP}, \mathsf{UL}, \mathsf{CT}_{x,\mathsf{RL}}, \mathsf{SK}_{f,\mathsf{ID}}) = M\big)$.

■ **Security Model:** The security of RABE under *selective revocation list model* against *chosen plaintext attacks* (CPA) is defined in terms of the following experiment between a probabilistic challenger $\mathcal{B}$ and a probabilistic polynomial-time adversary $\mathcal{A}$:

Init: $\mathcal{A}$ commits to a challenge descriptor input string $x^* \in \{0,1\}^\ell$ along with a challenge revoked user identity list RL^*.

Setup: $\mathcal{B}$ creates a user list UL including all users with identities in RL^* in it; generates a master secret key MK together with the public parameters PP by running $\mathsf{RABE.Setup}(1^\lambda, \ell, d, N_{\max})$; keeps MK to itself; and gives PP, UL to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ adaptively requests a polynomial number of decryption keys for circuit description $f \in \mathbb{F}_{\ell,d}$ along with user identity ID of its choice subject to the restriction that $[f(x^*) = 0] \vee [\mathsf{ID} \in \mathsf{RL}^*]$. $\mathcal{B}$ returns the corresponding decryption keys $\mathsf{SK}_{f,\mathsf{ID}}$ along with the updated user list UL to $\mathcal{A}$ by executing $\mathsf{RABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, \mathsf{UL}, ID, f)$.

Challenge: $\mathcal{A}$ submits two equal length messages $M_0^*, M_1^* \in \mathbb{M}$. $\mathcal{B}$ flips a random coin $b \in \{0,1\}$ and hands the challenge ciphertext CT^* to $\mathcal{A}$ by performing $\mathsf{RABE.Encrypt}(\mathsf{PP}, \mathsf{UL}, x^*, \mathsf{RL}^*, M_b^*)$.

Phase 2: $\mathcal{A}$ may continue adaptively to make a polynomial number of decryption key queries as in **Phase 1** with the same constraint as above.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

Definition 1. *An* RABE *scheme for circuits is said to be secure under selective revocation list model against* CPA *if the advantage of all probabilistic polynomial time adversaries $\mathcal{A}$ in the above game,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RABE,SRL\text{-}CPA}}(\lambda) = |\mathsf{Pr}[b' = b] - 1/2|$, *is at most negligible.*

2.2 Multilinear Maps and Complexity Assumptions

A (leveled) multilinear map [7–9] consists of the following two algorithms:

(I) $\mathcal{G}^{\mathsf{MLM}}(1^\lambda, \kappa)$: It takes as input a security parameter 1^λ and a positive integer κ indicating the number of allowed pairing operations. It outputs a sequence of groups $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ each of large prime order $p > 2^\lambda$ together with the canonical generators g_i of $\mathbb{G}_i$. We call $\mathbb{G}_1$ the source group, $\mathbb{G}_\kappa$ the target group, and $\mathbb{G}_2, \ldots, \mathbb{G}_{\kappa-1}$ intermediate groups. Let $\mathsf{PP}_{\mathsf{MLM}} = (\overrightarrow{\mathbb{G}}, g_1, \ldots, g_\kappa)$ be the description of the multilinear group with canonical generators.

(II) $e_{i,j}(g, h)$ (for $i, j \in \{1, \ldots, \kappa\}$ with $i + j \leq \kappa$): On input two elements $g \in \mathbb{G}_i$ and $h \in \mathbb{G}_j$ with $i + j \leq \kappa$, it outputs an element of $\mathbb{G}_{i+j}$ such that $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$ for $a, b \in \mathbb{Z}_p$. We often omit the subscripts and just write e. We can also generalize e to multiple inputs as $e(\chi^{(1)}, \ldots, \chi^{(t)}) = e(\chi^{(1)}, e(\chi^{(2)}, \ldots, \chi^{(t)}))$.

We refer g_i^a as a level-i encoding of $a \in \mathbb{Z}_p$. The scalar a itself is referred to as a level-0 encoding of a. Then the map e combines a level-i encoding of an element $a \in \mathbb{Z}_p$ and a level-j encoding of another element $b \in \mathbb{Z}_p$, and produces level-$(i + j)$ encoding of the product ab.

Assumption 1 [$(\boldsymbol{\kappa}, \boldsymbol{N})$-Multilinear Diffie-Hellman Exponent: $(\boldsymbol{\kappa}, \boldsymbol{N})$-MDHE [4]]. *The (κ, N)-Multilinear Diffie-Hellman Exponent ((κ, N)-*MDHE*) problem is to guess $\widetilde{b} \in \{0,1\}$ given $\varrho_{\widetilde{b}} = (\mathsf{PP}_{\mathsf{MLM}}, \vartheta_1, \ldots, \vartheta_N, \vartheta_{N+2}, \ldots, \vartheta_{2N}, \Upsilon, \tau_1, \ldots, \tau_{\kappa-2}, \Re_{\widetilde{b}})$ generated by $\mathcal{G}_{\widetilde{b}}^{(\kappa,N)\text{-}\mathsf{MDHE}}(1^\lambda)$, where $\mathcal{G}_{\widetilde{b}}^{(\kappa,N)\text{-}\mathsf{MDHE}}(1^\lambda)$ operates as follows: It runs $\mathcal{G}^{\mathsf{MLM}}(1^\lambda, \kappa)$ to generate $\mathsf{PP}_{\mathsf{MLM}}$ of order p; picks random $\alpha, \varsigma, \psi_1, \ldots, \psi_{\kappa-2} \in \mathbb{Z}_p$; computes $\vartheta_j = g_1^{\alpha^{(j)}}$ for $j = 1, \ldots, N, N+2, \ldots, 2N$, $\Upsilon = g_1^{\varsigma}, \tau_i = g_1^{\psi_i}$ for $i = 1, \ldots, \kappa - 2$; sets $\Re_0 = g_\kappa^{\alpha^{(N+1)}\varsigma \prod_{i=1}^{\kappa-2} \psi_i}$ while $\Re_1 =$ some random element in $\mathbb{G}_\kappa$; and finally returns $\varrho_{\widetilde{b}} = (\mathsf{PP}_{\mathsf{MLM}}, \vartheta_1, \ldots, \vartheta_N, \vartheta_{N+2}, \ldots, \vartheta_{2N}, \Upsilon, \tau_1, \ldots, \tau_{\kappa-2}, \Re_{\widetilde{b}})$.*

The (κ, N)-MDHE *assumption is that the advantage of all probabilistic polynomial time algorithms* $\mathcal{B}$ *in solving the above problem,* $\mathsf{Adv}_{\mathcal{B}}^{(\kappa,N)\text{-}\mathsf{MDHE}}(\lambda) = |\mathsf{Pr}[\mathcal{B}(1^{\lambda}, \varrho_0) \rightarrow 1] - \mathsf{Pr}[\mathcal{B}(1^{\lambda}, \varrho_1) \rightarrow 1]|$ *is at most negligible.*

Assumption 2 $[(\boldsymbol{n}, \boldsymbol{k}, \boldsymbol{l})$ **-Compressed Multilinear Diffie-Hellman Exponent:** $(\boldsymbol{n}, \boldsymbol{k}, \boldsymbol{l})$-$\mathsf{cMDHE}$ [13]]. *The* (n, k, l)*-Compressed Multilinear Diffie-Hellman Exponent* ((n, k, l)-cMDHE) *problem is to guess* $\widetilde{b} \in \{0, 1\}$ *given* $\varrho_{\widetilde{b}} = (\mathsf{PP}_{\mathsf{MLM}}, \xi_0, \ldots, \xi_n, \tau_1, \ldots, \tau_k, \varUpsilon, \Re_{\widetilde{b}})$ *generated by* $\mathcal{G}_{\widetilde{b}}^{(n,k,l)\text{-}\mathsf{cMDHE}}(1^{\lambda})$*, where* $\mathcal{G}_{\widetilde{b}}^{(n,k,l)\text{-}\mathsf{cMDHE}}(1^{\lambda})$ *operates as follows: It runs* $\mathcal{G}^{\mathsf{MLM}}(1^{\lambda}, \kappa = n + k + l - 1)$ *to generate* $\mathsf{PP}_{\mathsf{MLM}}$ *of order* p*; picks random* $\alpha, \varsigma, \psi_1, \ldots, \psi_k \in \mathbb{Z}_p$*; computes* $\xi_{\iota} = g_1^{\alpha^{(2^{\iota})}}$ *for* $\iota = 0, \ldots, n, \tau_h = g_1^{\psi_h}$ *for* $h = 1, \ldots, k$*,* $\varUpsilon = g_l^{\varsigma}$*; sets* $\Re_0 = g_{\kappa}^{\alpha^{(2^n - 1)}\varsigma \prod_{h=1}^{k} \psi_h}$ *while* $\Re_1 =$ *some random element of* $\mathbb{G}_{\kappa}$*; and finally returns* $\varrho_{\widetilde{b}} = (\mathsf{PP}_{\mathsf{MLM}}, \xi_0, \ldots, \xi_n, \tau_1, \ldots, \tau_k, \varUpsilon, \Re_{\widetilde{b}})$.

The (n, k, l)-cMDHE *assumption is that the advantage of all probabilistic polynomial time algorithms* $\mathcal{B}$ *in solving the above problem,* $\mathsf{Adv}_{\mathcal{B}}^{(n,k,l)\text{-}\mathsf{cMDHE}}(\lambda) = |\mathsf{Pr}[\mathcal{B}(1^{\lambda}, \varrho_0) \rightarrow 1] - \mathsf{Pr}[\mathcal{B}(1^{\lambda}, \varrho_1) \rightarrow 1]|$ *is at most negligible.*

3 RABE-I

■ **The Construction**:

RABE.Setup$(1^{\lambda}, \ell, d, N_{\max})$: The trusted key generation center takes as input a security parameter 1^{λ}, the length ℓ of Boolean inputs to the decryption circuits, the allowed depth d of decryption circuits, and the maximum number $N_{\max}$ of users supported by the system. Let $\mathcal{N} = \{1, \ldots, N_{\max}\}$ be the set of user key indices. It proceeds as follows:

1. It runs $\mathcal{G}^{\mathsf{MLM}}(1^{\lambda}, \kappa = \ell + d + 1)$ to obtain $\mathsf{PP}_{\mathsf{MLM}} = \big(\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_{\kappa}), g_1, \ldots, g_{\kappa}\big)$ of prime order $p > 2^{\lambda}$.
2. It selects random $(a_{1,0}, a_{1,1}), \ldots, (a_{\ell,0}, a_{\ell,1}) \in \mathbb{Z}_p^2$, and computes $A_{i,\beta} = g_1^{a_{i,\beta}}$ for $i = 1, \ldots, \ell$; $\beta \in \{0, 1\}$.
3. It selects random $\alpha, \gamma, \theta \in \mathbb{Z}_p$ and computes $\vartheta_j = g_1^{\alpha^{(j)}}$ for $j = 1, \ldots, N_{\max}, N_{\max} + 2, \ldots, 2N_{\max}, Y = g_1^{\gamma}$, $Z = g_{d-1}^{\theta}$, $\varOmega = g_{d+1}^{\alpha^{(N_{\max}+1)}\theta}$.
4. It initializes the user list UL, which would consist of ordered pairs (ID, u) such that ID is the identity of an user who has participated in the system and $u \in \mathcal{N}$ is the unique index assigned to ID by the key generation center at the time of subscription, as an empty set, i.e., it sets $\mathsf{UL} = \varnothing$.
5. Finally it publishes the public parameters $\mathsf{PP} = \big(\mathsf{PP}_{\mathsf{MLM}}, \{A_{i,\beta}\}_{i=1,\ldots,\ell;\beta\in\{0,1\}}, \{\vartheta_j\}_{j=1,\ldots,N_{\max},N_{\max}+2,\ldots,2N_{\max}}, Y, Z, \varOmega\big)$ along with the empty user list UL, while keeps the master secret key $\mathsf{MK} = (\alpha, \gamma, \theta)$ to itself.

RABE.KeyGen$(\mathsf{PP}, \mathsf{MK}, \mathsf{UL}, \mathsf{ID}, f)$: The key generation center takes the public parameters PP, the master secret key MK, the current user list UL, and the user identity ID together with the description $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ of the decryption circuit from a user as input. Our circuit has $\ell + q$ wires $\{1, \ldots, \ell + q\}$ where $\{1, \ldots, \ell\}$ are ℓ input wires, $\{\ell + 1, \ldots, \ell + q\}$ are q gates (OR or AND gates), and the wire $\ell + q$ is designated as the output wire. It proceeds as follows:

1. It first assigns an index $u \in \mathcal{N}$ such that $(\cdot, u) \notin \mathsf{UL}$ to ID and updates UL by adding the pair (ID, u).
2. It chooses random $r_1, \ldots, r_{\ell+q} \in \mathbb{Z}_p$ where we think of randomness r_w as being associated with wire $w \in \{1, \ldots, \ell + q\}$. It produces the "header" component $\mathcal{K} = g_d^{\alpha^{(u)}\theta\gamma - r_{\ell+q}}$.
3. It generates key components for every wire w. The structure of the key component depends upon the category of w, i.e., whether w is an Input wire, OR gate, or AND gate. We describe below how it generates the key components in each case.
   - *Input wire*: If $w \in \{1, \ldots, \ell\}$ then it corresponds to the w-th input. It computes the key component $\mathcal{K}_w = e(A_{w,1}, g_1)^{r_w} = g_2^{r_w a_{w,1}}$.
   - *OR gate*: Suppose that wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{OR}$, and $t = \mathsf{depth}(w)$. It picks random $\mu_w, \nu_w \in \mathbb{Z}_p$ and creates the key component
$$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)}}, K_{w,4} = g_t^{r_w - \nu_w r_{\mathbb{B}(w)}}\big).$$
   - *AND gate*: Let wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{AND}$, and $t = \mathsf{depth}(w)$. It selects random $\mu_w, \nu_w \in \mathbb{Z}_p$ and forms the key component
$$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)} - \nu_w r_{\mathbb{B}(w)}}\big).$$
4. It provides the decryption key $\mathsf{SK}_{f,\mathsf{ID}} = \big(f, \mathsf{ID}, \mathcal{K}, \{\mathcal{K}_w\}_{w \in \{1,\ldots,\ell+q\}}\big)$ to the user and publishes the updated user list UL.

RABE.Encrypt$(\mathsf{PP}, \mathsf{UL}, x, \mathsf{RL}, M)$: Taking as input the public parameters PP, the current user list UL, a descriptor input string $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$, a list RL of revoked user identities, and a message $M \in \mathbb{G}_\kappa$, the encrypter forms the ciphertext as follows:

1. It first defines the revoked user key index set $\mathsf{RI} \subseteq \mathcal{N}$ corresponding to RL using UL, i.e., if $\mathsf{ID} \in \mathsf{RL}$ and $(\mathsf{ID}, j) \in \mathsf{UL}$ it includes j in RI. It then determines $\mathsf{SI} = \mathcal{N} \backslash \mathsf{RI}$.
2. It picks random $s \in \mathbb{Z}_p$ and computes
$$C_M = e(\Omega, A_{1,x_1}, \ldots, A_{\ell,x_\ell})^s M = g_\kappa^{\alpha^{(N_{\max}+1)}\theta s \delta(x)} M,$$
$$C = g_1^s,\ C' = \big(Y \prod_{j \in \mathsf{SI}} \vartheta_{N_{\max}+1-j}\big)^s = \big(g_1^\gamma \prod_{j \in \mathsf{SI}} g_1^{\alpha^{(N_{\max}+1-j)}}\big)^s,$$
   where we define $\delta(x) = \prod_{i=1}^{\ell} a_{i,x_i}$ for the ease of exposition.
3. It outputs the ciphertext $\mathsf{CT}_{x,\mathsf{RL}} = (x, \mathsf{RL}, C_M, C, C')$.

RABE.Decrypt$(\mathsf{PP}, \mathsf{UL}, \mathsf{CT}_{x,\mathsf{RL}}, \mathsf{SK}_{f,\mathsf{ID}})$: A user, on input the public parameters PP, the current user list UL, a ciphertext $\mathsf{CT}_{x,\mathsf{RL}} = (x, \mathsf{RL}, C_M, C, C')$ encrypted for descriptor input string $x = x_1 \dots x_\ell \in \{0,1\}^\ell$ and a list of revoked user identities RL, along with its decryption key $\mathsf{SK}_{f,\mathsf{ID}} = \big(f, \mathsf{ID}, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\dots,\ell+q\}}\big)$ for its decryption circuit $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ as well as its user identity ID, where $u \in \mathcal{N}$ is the index assigned to ID (say), outputs $\perp$, if $[f(x) = 0] \vee [\mathsf{ID} \in \mathsf{RL}]$. Otherwise, (if $[f(x) = 1] \ \wedge \ [\mathsf{ID} \notin \mathsf{RL}]$) it proceeds as follows:

1. First, as a "header" computation it computes

$$D = e(A_{1,x_1}, \dots, A_{\ell,x_\ell}) = g_\ell^{\delta(x)},\ \widehat{E} = e(\mathcal{K}, D, C) = g_\kappa^{(\alpha^{(u)}\theta\gamma - r_{\ell+q})s\delta(x)},$$

extracting $\{A_{i,x_i}\}_{i=1,\dots,\ell}$ from PP.

2. Next, it performs the bottom-up evaluation of the circuit. For every wire w with corresponding $\mathsf{depth}(w) = t$, if $f_w(x) = 0$, nothing is computed for that wire, otherwise (if $f_w(x) = 1$), it attempts to compute $E_w = g_{\ell+t+1}^{r_w s\delta(x)}$ as follows. The user proceeds iteratively starting with computing E_1 and moves forward *in order* to finally compute $E_{\ell+q}$. Note that computing these values in order ensures that the computation on a wire w with $\mathsf{depth}(w) = t-1$ that evaluates to 1 will be defined before the computation on a wire w with $\mathsf{depth}(w) = t$. The computation procedure depends on whether the wire is an Input wire, OR gate, or AND gate.
   - *Input wire*: If $w \in \{1, \dots, \ell\}$ then it corresponds to the w-th input and $t = \mathsf{depth}(w) = 1$. Suppose that $x_w = f_w(x) = 1$. Extracting $\mathcal{K}_w$ from its decryption key $\mathsf{SK}_{f,\mathsf{ID}}$, the user computes

$$E_w = e(\mathcal{K}_w, A_{1,x_1}, \dots, A_{w-1,x_{w-1}}, A_{w+1,x_{w+1}}, \dots, A_{\ell,x_\ell}, C) = g_{\ell+1+1}^{r_w s\delta(x)}.$$

   - *OR gate*: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $t = \mathsf{depth}(w)$. Assume that $f_w(x) = 1$. Then the user checks whether $f_{\mathbb{A}(w)}(x) = 1$, i.e., the first input of gate w evaluated to 1, and if so, then the user extracts $K_{w,1}, K_{w,3}$ from $\mathcal{K}_w$ included in $\mathsf{SK}_{f,\mathsf{ID}}$ and computes

$$E_w = e(E_{\mathbb{A}(w)}, K_{w,1}) e(K_{w,3}, D, C) = g_{\ell+t+1}^{r_w s\delta(x)}.$$

     Note that $E_{\mathbb{A}(w)}$ is already computed at this stage in the bottom-up circuit evaluation as $\mathsf{depth}(\mathbb{A}(w)) = t - 1$.
     Alternatively, if $f_{\mathbb{A}(w)}(x) = 0$ then it must be the case that $f_{\mathbb{B}(w)}(x) = 1$ as $f_w(x) = 1$, and it computes

$$E_w = e(E_{\mathbb{B}(w)}, K_{w,2}) e(K_{w,4}, D, C) = g_{\ell+t+1}^{r_w s\delta(x)}$$

     extracting $K_{w,2}, K_{w,4}$ from $\mathcal{K}_w$ contained in $\mathsf{SK}_{f,\mathsf{ID}}$.
   - *AND gate*: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $t = \mathsf{depth}(w)$. Suppose that $f_w(x) = 1$. Then $f_{\mathbb{A}(w)}(x) = f_{\mathbb{B}(w)}(x) = 1$. The user computes

$$E_w = e(E_{\mathbb{A}(w)}, K_{w,1}) e(E_{\mathbb{B}(w)}, K_{w,2}) e(K_{w,3}, D, C) = g_{\ell+t+1}^{r_w s\delta(x)}$$

     extracting $K_{w,1}, K_{w,2}, K_{w,3}$ from $\mathcal{K}_w$ in $\mathsf{SK}_{f,\mathsf{ID}}$.
     The user finally computes $E_{\ell+q} = g_\kappa^{r_{\ell+q} s\delta(x)}$, as $f(x) = f_{\ell+q}(x) = 1$.

3. It determines the revoked user key index set $\mathsf{RI} \subseteq \mathcal{N}$ corresponding to RL using UL and obtains $\mathsf{SI} = \mathcal{N} \backslash \mathsf{RI}$ which contains all the non-revoked user key indices. Note that since $\mathsf{ID} \notin \mathsf{RL}$, $u \in \mathsf{SI}$. The user retrieves the message by the following computation:
$$C_M \widehat{E} E_{\ell+q} e\big(\prod_{j \in \mathsf{SI}\backslash\{u\}} \vartheta_{N_{\max}+1-j+u}, Z, D, C\big) e\big(\vartheta_u, Z, D, C'\big)^{-1} = M.$$

■ **Security Analysis**:

Theorem 1. $\mathsf{RABE\text{-}I}$ *is secure in the selective revocation list model against* CPA *as per the security model of Sect. 2.1 if the* $(\ell + d + 1, N_{\max})$-MDHE *assumption holds for the underlying multilinear group generator* $\mathcal{G}^{\mathsf{MLM}}$*, described in Sect. 2.2, where* $\ell, d,$ *and* $N_{\max}$ *denote respectively the input length of decryption circuits, depth of the decryption circuits, and the maximum number of users supported by the system.*

Proof. Suppose that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that attacks $\mathsf{RABE\text{-}I}$ as per the selective revocation list model under CPA with a non-negligible advantage. We construct a probabilistic algorithm $\mathcal{B}$ that attempts to solve an instance of the $(\ell + d + 1, N_{\max})$-MDHE problem using $\mathcal{A}$ as a subroutine. $\mathcal{B}$ is given a challenge instance
$$\varrho_{\widetilde{b}} = (\mathsf{PP_{MLM}}, \vartheta_1, \ldots, \vartheta_{N_{\max}}, \vartheta_{N_{\max}+2}, \ldots, \vartheta_{2N_{\max}}, \Upsilon, \tau_1, \ldots, \tau_{\ell+d-1}, \Re_{\widetilde{b}})$$
$$\text{where } \{\vartheta_j = g_1^{\alpha^{(j)}}\}_{j=1,\ldots,N_{\max},N_{\max}+2,\ldots,2N_{\max}}, \{\tau_i = g_1^{\psi_i}\}_{i=1,\ldots,\ell+d-1}, \Upsilon = g_1^{\varsigma}$$
such that $\alpha, \varsigma, \psi_i$ are random elements of $\mathbb{Z}_p$, and $\Re_{\widetilde{b}}$ is $g_{\ell+d+1}^{\alpha^{(N_{\max}+1)}\varsigma \prod_{i=1}^{\ell+d-1} \psi_i}$ or some random element in $\mathbb{G}_{\ell+d+1}$ according as $\widetilde{b}$ is 0 or 1. $\mathcal{B}$ plays the role of the challenger in the CPA security game as per the selective revocation list model of Sect. 2.1 and interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ declares the challenge input string $x^* = x_1^* \ldots x_\ell^* \in \{0,1\}^\ell$ along with the challenge revocation list RL^* to $\mathcal{B}$. Let $\mathcal{N} = \{1, \ldots, N_{\max}\}$ be the set of user key indices. $\mathcal{B}$ first initializes the user list $\mathsf{UL} = \varnothing$. Next for each $\mathsf{ID} \in \mathsf{RL}^*$ it selects an index $j \in \mathcal{N}$ such that $(\cdot, j) \notin \mathsf{UL}$ and adds (ID, j) to UL. Let $\mathsf{RI}^* \subseteq \mathcal{N}$ be the revoked set of user key indices corresponding to RL^* and $\mathsf{SI}^* = \mathcal{N} \backslash \mathsf{RI}^*$.

Setup: $\mathcal{B}$ chooses random $z_1, \ldots, z_\ell, \varphi \in \mathbb{Z}_p$ and sets
$$A_{i,\beta} = \tau_i = g_1^{\psi_i}, \text{ if } \beta = x_i^*, \; A_{i,\beta} = g_1^{z_i}, \text{ if } \beta \neq x_i^*, \text{ for } i = 1, \ldots, \ell; \; \beta \in \{0,1\},$$
$$Y = g_1^{\varphi}\big(\prod_{j \in \mathsf{SI}^*} \vartheta_{N_{\max}+1-j}\big)^{-1} = g_1^{\gamma}, \; Z = e(\tau_{\ell+1}, \ldots, \tau_{\ell+d-1}) = g_{d-1}^{\theta},$$
$$\Omega = e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+d-1}) = g_{d+1}^{\alpha^{(N_{\max}+1)}\theta}.$$
Note that the above setting corresponds to (possibly) *implicitly* letting
$$a_{i,\beta} = \psi_i, \text{ if } \beta = x_i^*, \; a_{i,\beta} = z_i, \text{ if } \beta \neq x_i^*, \text{ for } i = 1, \ldots, \ell; \; \beta \in \{0,1\},$$
$$\gamma = \varphi - \sum_{j \in \mathsf{SI}^*} \alpha^{(N_{\max}+1-j)}, \; \theta = \prod_{h=\ell+1}^{\ell+d-1} \psi_h = \Gamma(\ell+1, \ell+d-1),$$

where we define $\Gamma(v_1, v_2) = \prod_{h=v_1}^{v_2} \psi_h$ for positive integers v_1, v_2, with the convention that $\Gamma(v_1, v_2) = 1$ if $v_1 > v_2$, for the purpose of enhancing readability in subsequent discussion. $\mathcal{B}$ hands the public parameters

$$\mathsf{PP} = \big(\mathsf{PP}_{\mathsf{MLM}}, \{A_{i,\beta}\}_{i=1,\ldots,\ell;\beta\in\{0,1\}}, \{\vartheta_j\}_{j=1,\ldots,N_{\max},N_{\max}+2,\ldots,2N_{\max}}, Y, Z, \Omega\big)$$

along with the user list UL to $\mathcal{A}$.

Phase 1 and **Phase 2**: Both the key query phases are executed in the same manner by $\mathcal{B}$. So, we describe them once here. $\mathcal{A}$ adaptively queries a decryption key for a circuit $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ and user identity ID to $\mathcal{B}$ subject to the restriction that $[f(x^*) = 0] \ \vee\ [\mathsf{ID} \in \mathsf{RL}^*]$. $\mathcal{B}$ answers the query as follows:

Case (I) ($\mathsf{ID} \in \mathsf{RL}^*$): $\mathcal{B}$ retrieves the index $u \in \mathcal{N}$ already assigned to ID in the initialization phase from UL. $\mathcal{B}$ forms the decryption key components $\mathcal{K}_w$ corresponding to all the wires $w \in \{1, \ldots, \ell+q\}$ of the circuit f exactly as in the real scheme. Next $\mathcal{B}$ sets the "header" component $\mathcal{K}$ of the decryption key as

$$\mathcal{K} = e(\vartheta_u, Z)^{\varphi}\Big[\prod_{j\in\mathsf{SI}^*} e(\vartheta_{N_{\max}+1-j+u}, Z)\Big]^{-1} g_d^{-r_{\ell+q}},$$

where $r_{\ell+q} \in \mathbb{Z}_p$ is the randomness associated with the wire $\ell+q$ already selected by $\mathcal{B}$ at the time of computing the decryption key component $\mathcal{K}_{\ell+q}$. The above simulation of $\mathcal{K} = g_d^{\alpha^{(u)}\theta\gamma - r_{\ell+q}}$ is valid since

$$\begin{aligned}\alpha^{(u)}\theta\gamma - r_{\ell+q} &= \alpha^{(u)}\Gamma(\ell+1, \ell+d-1)\big[\varphi - \sum_{j\in\mathsf{SI}^*} \alpha^{(N_{\max}+1-j)}\big] - r_{\ell+q}\\ &= \alpha^{(u)}\Gamma(\ell+1, \ell+d-1)\varphi - \Gamma(\ell+1, \ell+d-1)\sum_{j\in\mathsf{SI}^*} \alpha^{(N_{\max}+1-j+u)} - r_{\ell+q}.\end{aligned}$$

Further, notice that since $\mathsf{ID} \in \mathsf{RL}^*$, $u \notin \mathsf{SI}^*$. Hence, none of the $\alpha^{(N_{\max}+1-j+u)}$ in the preceding equation matches $\alpha^{(N_{\max}+1)}$, enabling $\mathcal{B}$ to simulate $\mathcal{K}$ as above using the available information.

Case (II) ($\mathsf{ID} \notin \mathsf{RL}^*$): In this case $\mathcal{B}$ assigns an index $u \in \mathcal{N}$ such that $(\cdot, u) \notin \mathsf{UL}$ to ID and adds (ID, u) to UL. Now observe that due to the restriction on $\mathcal{A}$'s decryption key queries we must have $f(x^*) = 0$ in this case. As in [10], we will think of the simulation as having some invariant property on the depth of the wire we are looking at. Consider a wire w with $\mathsf{depth}(w) = t$. $\mathcal{B}$ views r_w, the randomness associated with the wire w, as follows: If $f_w(x^*) = 0$, then $\mathcal{B}$ will *implicitly* view r_w as the term $-\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-1)$ plus some additional known randomization term. Otherwise (if $f_w(x^*) = 1$), $\mathcal{B}$ will view r_w as 0 plus some additional known randomization term. We keep this property intact for the bottom-up key simulation of the circuit. This makes $\mathcal{B}$ to view $r_{\ell+q}$ as $-\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+d-1)$ plus some additional known randomization term since $f_{\ell+q}(x^*) = f(x^*) = 0$. Then $\mathcal{B}$ can simulate the "header" component $\mathcal{K}$ by cancelation as will be explained shortly.

The bottom-up simulation of the key component for each wire w by $\mathcal{B}$ varies depending on whether w is an Input wire, OR gate, or AND gate as follows:

- *Input wire*: Consider $w \in \{1, \ldots, \ell\}$, i.e., an input wire. Hence, $\mathsf{depth}(w) = 1$.
  - If $x^*_w = 1$ then $\mathcal{B}$ picks random $r_w \in \mathbb{Z}_p$ (as is done honestly) and sets the key component $\mathcal{K}_w = e(\tau_w, g_1)^{r_w} = g_2^{r_w a_{w,1}}$.
  - Otherwise, if $x^*_w = 0$ then $\mathcal{B}$ *implicitly* lets $r_w = -\alpha^{(N_{\max}+1)} + \eta_w = -\alpha^{(N_{\max}+1)} \Gamma(\ell+1, \ell+1-1) + \eta_w$, where $\eta_w \in \mathbb{Z}_p$ is randomly selected by $\mathcal{B}$, and sets the key component $\mathcal{K}_w = e(\vartheta_{N_{\max}}, \vartheta_1)^{-z_w} g_2^{\eta_w z_w} = g_2^{r_w a_{w,1}}$.

- *OR gate*: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $t = \mathsf{depth}(w)$. Then, $\mathsf{depth}(\mathbb{A}(w)) = \mathsf{depth}(\mathbb{B}(w)) = t - 1$ as our circuit is layered.
  - If $f_w(x^*) = 1$ then $\mathcal{B}$ chooses random $\mu_w, \nu_w, r_w \in \mathbb{Z}_p$ as in the real scheme, and forms the key component as

$$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)}}, K_{w,4} = g_t^{r_w - \nu_w r_{\mathbb{B}(w)}}\big).$$

    Lets have a closer look to the simulation of $K_{w,3}$ and $K_{w,4}$ in $\mathcal{K}_w$ by $\mathcal{B}$ above. Since $f_w(x^*) = 1$, the $\mathbb{A}(w)$ and $\mathbb{B}(w)$ gates might evaluate to 1 or 0 upon input x^* with the only restriction that both of them cannot be 0 at the same time. Consider the case of $K_{w,3}$. Observe that if $f_{\mathbb{A}(w)}(x^*) = 1$, then $r_{\mathbb{A}(w)}$ is a random element in $\mathbb{Z}_p$ already selected by $\mathcal{B}$ at this stage due to the bottom-up key simulation. Thus, in this case $\mathcal{B}$ can simulate $K_{w,3}$ exactly as in the real scheme. Now, let $f_{\mathbb{A}(w)}(x^*) = 0$. Therefore, $r_{\mathbb{A}(w)}$ has been implicitly set as $-\alpha^{(N_{\max}+1)} \Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)}$ by $\mathcal{B}$ in the course of its bottom-up key simulation, where $\eta_{\mathbb{A}(w)} \in \mathbb{Z}_p$ is randomly chosen by $\mathcal{B}$. Thus, in this case $\mathcal{B}$ can create $K_{w,3}$ as

$$K_{w,3} = e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+t-2})^{\mu_w} g_t^{r_w - \mu_w \eta_{\mathbb{A}(w)}} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)}}.$$

    A similar argument holds for $K_{w,4}$.
  - On the other hand, if $f_w(x^*) = 0$ then $\mathcal{B}$ picks random $\sigma_w, \zeta_w, \eta_w \in \mathbb{Z}_p$, *implicitly* sets $\mu_w = \psi_{\ell+t-1} + \sigma_w, \nu_w = \psi_{\ell+t-1} + \zeta_w$, along with $r_w = -\alpha^{(N_{\max}+1)} \Gamma(\ell+1, \ell+t-1) + \eta_w$, and creates the key component $\mathcal{K}_w = (K_{w,1}, K_{w,2}, K_{w,3}, K_{w,4})$ as follows:

$$\begin{aligned}
K_{w,1} = &\ \tau_{\ell+t-1} g_1^{\sigma_w} = g_1^{\mu_w}, K_{w,2} = \tau_{\ell+t-1} g_1^{\zeta_w} = g_1^{\nu_w},\\
K_{w,3} = &\ e(\tau_{\ell+t-1}, g_{t-1})^{-\eta_{\mathbb{A}(w)}} e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+t-2})^{\sigma_w} g_t^{\eta_w - \sigma_w \eta_{\mathbb{A}(w)}}\\
= &\ g_t^{\eta_w - \psi_{\ell+t-1} \eta_{\mathbb{A}(w)} - \sigma_w(-\alpha^{(N_{\max}+1)} \Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)})} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)}},\\
K_{w,4} = &\ e(\tau_{\ell+t-1}, g_{t-1})^{-\eta_{\mathbb{B}(w)}} e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+t-2})^{\zeta_w} g_t^{\eta_w - \zeta_w \eta_{\mathbb{B}(w)}}\\
= &\ g_t^{\eta_w - \psi_{\ell+t-1} \eta_{\mathbb{B}(w)} - \zeta_w(-\alpha^{(N_{\max}+1)} \Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{B}(w)})} = g_t^{r_w - \nu_w r_{\mathbb{B}(w)}}.
\end{aligned}$$

Note that since $f_w(x^*) = 0$, $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 0$. Therefore, $\mathcal{B}$'s bottom-up key simulation has implicitly set $r_{\mathbb{A}(w)} = -\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)}$, where $\eta_{\mathbb{A}(w)} \in \mathbb{Z}_p$ is randomly picked by $\mathcal{B}$. Hence,

$$\begin{aligned} & r_w - \mu_w r_{\mathbb{A}(w)} \\ & = \eta_w - \psi_{\ell+t-1}\eta_{\mathbb{A}(w)} - \sigma_w\big(-\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)}\big) \end{aligned} \quad (1)$$

establishing that the distribution of simulated $K_{w,3}$ by $\mathcal{B}$ is identical to that in the actual construction. Analogous argument holds for $K_{w,4}$.

- *AND gate*: Consider wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $t = \mathsf{depth}(w)$. Then $\mathsf{depth}(\mathbb{A}(w)) = \mathsf{depth}(\mathbb{B}(w)) = t-1$ for the reason that our circuit is layered.

  – Let $f_w(x^*) = 1$. Then $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 1$. $\mathcal{B}$ picks random $\mu_w, \nu_w, r_w \in \mathbb{Z}_p$ and forms the key component

  $$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_t^{r_w - \mu_w r_{\mathbb{A}(w)} - \nu_w r_{\mathbb{B}(w)}}\big)$$

  exactly as in the real scheme. Observe that, since $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 1$, $r_{\mathbb{A}(w)}$ and $r_{\mathbb{B}(w)}$ are random elements of $\mathbb{Z}_p$ already chosen by $\mathcal{B}$ in the course of the bottom-up simulation.

  – Alternatively, let $f_w(x^*) = 0$. Then, $f_{\mathbb{A}(w)}(x^*) = 0$ or $f_{\mathbb{B}(w)}(x^*) = 0$. If $f_{\mathbb{A}(w)}(x^*) = 0$, then $\mathcal{B}$ selects $\sigma_w, \zeta_w, \eta_w \in \mathbb{Z}_p$, *implicitly* defines $\mu_w = \psi_{\ell+t-1} + \sigma_w, \nu_w = \zeta_w$, and $r_w = -\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-1) + \eta_w$, and determines the decryption key component $\mathcal{K}_w = (K_{w,1}, K_{w,2}, K_{w,3})$ by setting

$$\begin{aligned} K_{w,1} = {} & \tau_{\ell+t-1} g_1^{\sigma_w} = g_1^{\mu_w}, K_{w,2} = g_1^{\zeta_w} = g_1^{\nu_w}, \\ K_{w,3} = {} & e(\tau_{\ell+t-1}, g_{t-1})^{-\eta_{\mathbb{A}(w)}} e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+t-2})^{\sigma_w} g_t^{\eta_w - \sigma_w \eta_{\mathbb{A}(w)}} \cdot \\ & \big(g_t^{r_{\mathbb{B}(w)}}\big)^{-\zeta_w} = g_t^{\eta_w - \psi_{\ell+t-1}\eta_{\mathbb{A}(w)} - \sigma_w(-\alpha^{(N_{\max}+1)}\Gamma(\ell+1,\ell+t-2)+\eta_{\mathbb{A}(w)}) - \zeta_w r_{\mathbb{B}(w)}} \\ = {} & g_t^{r_w - \mu_w r_{\mathbb{A}(w)} - \nu_w r_{\mathbb{B}(w)}}. \end{aligned}$$

  The simulated $K_{w,3}$ by $\mathcal{B}$ above is identically distributed as that in the original construction. This follows from the fact that, the $\mathbb{A}(w)$ gate being evaluated to 0, $r_{\mathbb{A}(w)}$ has already been implicitly set as $r_{\mathbb{A}(w)} = -\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)}$ by $\mathcal{B}$ upon selecting random $\eta_{\mathbb{A}(w)} \in \mathbb{Z}_p$ in the course of the bottom-up key simulation. Therefore, as in Eq. (1), we have

  $$\begin{aligned} & r_w - \mu_w r_{\mathbb{A}(w)} \\ & = \eta_w - \psi_{\ell+t-1}\eta_{\mathbb{A}(w)} - \sigma_w\big(-\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+t-2) + \eta_{\mathbb{A}(w)}\big). \end{aligned}$$

  Notice that $g_t^{r_{\mathbb{B}(w)}}$ is always computable by $\mathcal{B}$ from the available information regardless of whether the $\mathbb{B}(w)$ gate evaluates to 1 or 0 upon input x^*. If $f_{\mathbb{B}(w)}(x^*) = 1$, then $r_{\mathbb{B}(w)}$ is a random element of $\mathbb{Z}_p$ chosen by $\mathcal{B}$

itself during the bottom-up simulation process. Hence, the computation of $g_t^{r_{\mathbb{B}(w)}}$ is straightforward in this case. Otherwise, if $f_{\mathbb{B}(w)}(x^*) = 0$, then $\mathcal{B}$ has already set $r_{\mathbb{B}(w)}$ as $r_{\mathbb{B}(w)} = -\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+d-2) + \eta_{\mathbb{B}(w)}$ at this stage by selecting random $\eta_{\mathbb{B}(w)} \in \mathbb{Z}_p$. Therefore, in this case $\mathcal{B}$ can compute $g_t^{r_{\mathbb{B}(w)}}$ as $g_t^{r_{\mathbb{B}(w)}} = e(\vartheta_{N_{\max}}, \vartheta_1, \tau_{\ell+1}, \ldots, \tau_{\ell+t-2})^{-1} g_t^{\eta_{\mathbb{B}(w)}}$.
The case where $f_{\mathbb{B}(w)}(x^*) = 0$ and $f_{\mathbb{A}(w)}(x^*) = 1$ can be argued analogously, with the roles of μ_w and ν_w reversed.

Since $f(x^*) = f_{\ell+q}(x^*) = 0$, $r_{\ell+q} = -\alpha^{(N_{\max}+1)}\Gamma(\ell+1, \ell+d-1) + \eta_{\ell+q}$, where $\eta_{\ell+q} \in \mathbb{Z}_p$ is randomly selected by $\mathcal{B}$. Also, since $\mathsf{ID} \notin \mathsf{RL}^*$, $u \in \mathsf{SI}^*$. These two facts allow $\mathcal{B}$ to compute the "header" component of the key as

$$\begin{aligned}\mathcal{K} = \; & e(\vartheta_u, Z)^{\varphi}\Big[\prod_{j\in\mathsf{SI}^*\setminus\{u\}} e(\vartheta_{N_{\max}+1-j+u}, Z)\Big]^{-1} g_d^{-\eta_{\ell+q}} \\ = \; & g_d^{\alpha^{(u)}\Gamma(\ell+1,\ell+d-1)\varphi - \Gamma(\ell+1,\ell+d-1)\sum_{j\in\mathsf{SI}^*\setminus\{u\}}\alpha^{(N_{\max}+1-j+u)} - \eta_{\ell+q}} \\ = \; & g_d^{\alpha^{(u)}\Gamma(\ell+1,\ell+d-1)\left[\varphi - \sum_{j\in\mathsf{SI}^*}\alpha^{(N_{\max}+1-j)}\right] - r_{\ell+q}} = g_d^{\alpha^{(u)}\theta\gamma - r_{\ell+q}}.\end{aligned}$$

$\mathcal{B}$ provides $\mathcal{A}$ the decryption key $\mathsf{SK}_{f,\mathsf{ID}} = \big(f, \mathsf{ID}, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\ldots,\ell+q\}}\big)$ along with the updated user list UL.

Challenge: $\mathcal{A}$ submits two challenge messages $M_0^*, M_1^* \in \mathbb{G}_{\ell+d+1}$ to $\mathcal{B}$. $\mathcal{B}$ flips a random coin $b \in \{0,1\}$, sets the challenge ciphertext

$$\mathsf{CT}^* = \big(x^*, \mathsf{RL}^*, C_M^* = \Re_{\widetilde{b}} M_b^*, C^* = \Upsilon = g_1^{\varsigma}, C^{'*} = \Upsilon^{\varphi} = (Y\prod_{j\in\mathsf{SI}^*}\vartheta_{N_{\max}+1-j})^{\varsigma}\big),$$

and gives it to $\mathcal{A}$.

Guess: $\mathcal{B}$ eventually receives back the guess $b' \in \{0,1\}$ from $\mathcal{A}$. If $b = b'$, $\mathcal{B}$ outputs $\widetilde{b}' = 1$; otherwise, it outputs $\widetilde{b}' = 0$.

Note that if $\widetilde{b} = 0$, then

$$C_m^* = \Re_{\widetilde{b}} M_b^* = g_{\ell+d+1}^{\alpha^{(N_{\max}+1)}\varsigma\Gamma(1,\ell+d-1)} M_b^* = g_{\kappa}^{\alpha^{(N_{\max}+1)}\theta\varsigma\delta(x^*)} M_b^*,$$

where $\delta(x^*) = \prod_{i=1}^{\ell} a_{i,x_i^*}$. Thus, we can see that the challenge ciphertext CT^* is properly generated by $\mathcal{B}$ in this case by *implicitly* letting $\underset{\sim}{s}$, the randomness used to prepare the ciphertext, as ς. On the other hand, if $\widetilde{b} = 1$, then $\Re_{\widetilde{b}}$ is a random element of $\mathbb{G}_{\ell+d+1}$, so that, the challenge ciphertext is completely random. Hence the result. $\square$

4 RABE-II

■ **The Construction**:
RABE.Setup$(1^{\lambda}, \ell, d, N_{\max})$: Taking as input a security parameter 1^{λ}, the length ℓ of Boolean inputs to the decryption circuits, the allowed depth d of decryption circuits, and the maximum number $N_{\max}$ of users supported by the system, the trusted key generation center proceeds as follows:

1. It chooses two positive integers n, m suitably such that $N_{\max} \leq \binom{n}{m}$. Let $\mathcal{N}$ denotes the set of all integers $j \in \{1, \dots, 2^n - 2\}$ of Hamming weight $\mathsf{HW}(j) = m$ when expressed as a bit string of length n. $\mathcal{N}$ is considered as the set of possible user key indices.
2. It executes $\mathcal{G}^{\mathsf{MLM}}(1^\lambda, \kappa = n + d + m - 1)$ to generate $\mathsf{PP}_{\mathsf{MLM}} = \big(\vec{\mathbb{G}} = (\mathbb{G}_1, \dots, \mathbb{G}_\kappa), g_1, \dots, g_\kappa\big)$ of prime order $p > 2^\lambda$.
3. It picks random $a_1, \dots, a_\ell \in \mathbb{Z}_p$ and computes $A_i = g_m^{a_i}$ for $i = 1, \dots, \ell$.
4. It chooses random $\alpha, \gamma, \theta \in \mathbb{Z}_p$ and computes $\xi_\iota = g_1^{\alpha^{(2^\iota)}}$ for $\iota = 0, \dots, n$, $Y = g_{n-1}^\gamma$, $Z = g_d^\theta$, $\Omega = g_\kappa^{\alpha^{(2^n-1)}\theta}$.
5. It initializes the user list UL, which would consist of ordered pairs (ID, u) such that ID is the identity of an user who has participated in the system and $u \in \mathcal{N}$ is the unique index assigned to ID by the key generation center at the time of subscription, as an empty set, i.e., it sets $\mathsf{UL} = \varnothing$.
6. It keeps the master secret key $\mathsf{MK} = (\alpha, \gamma, \theta)$ to itself while publishes the public parameters $\mathsf{PP} = \big(\mathsf{PP}_{\mathsf{MLM}}, n, m, \{A_i\}_{i=1,\dots,\ell}, \{\xi_\iota\}_{\iota=0,\dots,n}, Y, Z, \Omega\big)$ along with the empty user list UL.

RABE.KeyGen$(\mathsf{PP}, \mathsf{MK}, \mathsf{UL}, \mathsf{ID}, f)$: The key generation center intakes the public parameters PP, the master secret key MK, the current user list UL, and the user identity ID together with the description $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ of the decryption circuit from a user. Our circuit has $\ell + q$ wires $\{1, \dots, \ell + q\}$ where $\{1, \dots, \ell\}$ are ℓ input wires, $\{\ell + 1, \dots, \ell + q\}$ are q gates (OR or AND gates), and the wire $\ell + q$ is distinguished as the output wire. It proceeds as follows:

1. It first assigns to ID an index $u \in \mathcal{N}$ such that $(\cdot, u) \notin \mathsf{UL}$ and updates UL by adding the pair (ID, u).
2. It chooses random $r_1, \dots, r_{\ell+q} \in \mathbb{Z}_p$ where we think of randomness r_w as being associated with wire $w \in \{1, \dots, \ell + q\}$. It produces the "header" component $\mathcal{K} = g_{n+d-1}^{\alpha^{(u)}\theta\gamma - r_{\ell+q}}$.
3. It forms key components for every wire w. The structure of the key component depends upon the category of w, i.e., whether w is an Input wire, OR gate, or AND gate. We describe below how it generates the key components in each case.
   - *Input wire*: If $w \in \{1, \dots, \ell\}$ then it corresponds to the w-th input. It chooses random $z_w \in \mathbb{Z}_p$ and computes the key component
   $$\mathcal{K}_w = \big(K_{w,1} = g_n^{r_w} e(A_w, g_{n-m})^{z_w} = g_n^{r_w} g_n^{a_w z_w},\ K_{w,2} = g_n^{-z_w}\big).$$
   - *OR gate*: Suppose that wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{OR}$, and $t = \mathsf{depth}(w)$. It picks random $\mu_w, \nu_w \in \mathbb{Z}_p$ and creates the key component
   $$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_{n+t-1}^{r_w - \mu_w r_{\mathbb{A}(w)}}, K_{w,4} = g_{n+t-1}^{r_w - \nu_w r_{\mathbb{B}(w)}}\big).$$
   - *AND gate*: Let wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{AND}$, and $t = \mathsf{depth}(w)$. It selects random $\mu_w, \nu_w \in \mathbb{Z}_p$ and forms the key component
   $$\mathcal{K}_w = \big(K_{w,1} = g_1^{\mu_w}, K_{w,2} = g_1^{\nu_w}, K_{w,3} = g_{n+t-1}^{r_w - \mu_w r_{\mathbb{A}(w)} - \nu_w r_{\mathbb{B}(w)}}\big).$$

4. It provides the decryption key $\mathsf{SK}_{f,\mathsf{ID}} = \big(f, \mathsf{ID}, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\ldots,\ell+q\}}\big)$ to the user and publishes the updated user list UL.

RABE.Encrypt($\mathsf{PP}, \mathsf{UL}, x, \mathsf{RL}, M$): On input the public parameters PP, the current user list UL, a descriptor input string $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$, a revoked user identity list RL, and a message $M \in \mathbb{G}_\kappa$, the encrypter proceeds as follows:

1. It defines the revoked user key index set $\mathsf{RI} \subseteq \mathcal{N}$ corresponding to RL using UL, i.e., if $\mathsf{ID} \in \mathsf{RL}$ and $(\mathsf{ID}, j) \in \mathsf{UL}$ it puts j in RI, and sets $\mathsf{SI} = \mathcal{N} \backslash \mathsf{RI}$.
2. It computes ϑ_{2^n-1-j} for all $j \in \mathsf{SI}$ utilizing the ξ_ι values included in PP and multilinear map as follows, where we define $\vartheta_\varpi = g_{n-1}^{\alpha^{(\varpi)}}$ for positive integer ϖ. Observe that any $j \in \mathsf{SI} \subseteq \mathcal{N}$ can be expressed as a bit string of length n with $\mathsf{HW}(j) = m$. Hence, j can be written as $j = \sum_{\iota\in J} 2^\iota$ where $J \subseteq \{0,\ldots,n-1\}$ of size m. Now $2^n - 1 = \sum_{\iota=0}^{n-1} 2^\iota$. Thus, $2^n - 1 - j = \sum_{\iota\in\overline{J}} 2^\iota$ where $\overline{J} = \{0,\ldots,n-1\}\backslash J = \{\iota_1,\ldots,\iota_{n-m}\}$. It computes ϑ_{2^n-1-j} as
$$\vartheta_{2^n-1-j} = e(\xi_{\iota_1},\ldots,\xi_{\iota_{n-m}}, g_{m-1}) = g_{n-1}^{\alpha^{(2^n-1-j)}}.$$
3. It picks random $s \in \mathbb{Z}_p$ and computes
$$\begin{aligned} C_M &= \Omega^s M = g_\kappa^{\alpha^{(2^n-1)}\theta s} M,\ C = g_m^s, \\ C'_i &= A_i^s = g_m^{a_i s} \text{ for } i \in \mathcal{S}_x = \{i | i \in \{1,\ldots,\ell\} \ \wedge\ x_i = 1\}, \\ C'' &= \big(Y \prod_{j\in\mathsf{SI}} \vartheta_{2^n-1-j}\big)^s = \big(g_{n-1}^\gamma \prod_{j\in\mathsf{SI}} g_{n-1}^{\alpha^{(2^n-1-j)}}\big)^s. \end{aligned}$$
4. It outputs the ciphertext $\mathsf{CT}_{x,\mathsf{RL}} = (x, \mathsf{RL}, C_M, C, \{C'_i\}_{i\in\mathcal{S}_x}, C'')$.

Remark 1. We would like to mention that the number of ciphertext components could be made constant (precisely only 4), as in RABE-I, rather than scaling with the size of $\mathcal{S}_x$ using a $(\ell + n + d + m - 2)$-leveled multilinear map. However, since in current approximate multilinear map candidates the multilinearity is expensive, we opt for a construction that requires lower multilinearity level.

RABE.Decrypt($\mathsf{PP}, \mathsf{UL}, \mathsf{CT}_{x,\mathsf{RL}}, \mathsf{SK}_{f,\mathsf{ID}}$): A user intakes the public parameters PP, current user list UL, a ciphertext $\mathsf{CT}_{x,\mathsf{RL}} = (x, \mathsf{RL}, C_M, C, \{C'_i\}_{i\in\mathcal{S}_x}, C'')$ encrypted for descriptor input string $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$ along with a revoked user identity list RL, and its decryption key $\mathsf{SK}_{f,\mathsf{ID}} = \big(f, \mathsf{ID}, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\ldots,\ell+q\}}\big)$ for its decryption policy circuit $f = (\ell, q, d, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ as well as its user identity ID, where $u \in \mathcal{N}$ is the index assigned to ID (say). It outputs $\perp$, if $[f(x) = 0] \vee [\mathsf{ID} \in \mathsf{RL}]$; otherwise, (if $[f(x) = 1] \wedge [\mathsf{ID} \notin \mathsf{RL}]$) proceeds as follows:

1. First, as a "header" computation, it computes
$$\widehat{E} = e(\mathcal{K}, C) = e\big(g_{n+d-1}^{\alpha^{(u)}\theta\gamma - r_{\ell+q}}, g_m^s\big) = g_\kappa^{(\alpha^{(u)}\theta\gamma - r_{\ell+q})s}.$$
2. Next, it performs the bottom-up evaluation of the circuit. For every wire w with corresponding $\mathsf{depth}(w) = t$, if $f_w(x) = 0$, nothing is computed for that wire, otherwise (if $f_w(x) = 1$), it attempts to compute $E_w = g_{n+t+m-1}^{r_w s}$

as described below. The user proceeds iteratively starting with computing E_1 and moves forward in order to finally compute $E_{\ell+q}$. Note that computing these values *in order* ensures that the computation on a wire w with $\mathsf{depth}(w) = t-1$ that evaluates to 1 will be defined before the computation on a wire w with $\mathsf{depth}(w) = t$. The computation procedure depends on whether the wire is an Input wire, OR gate, or AND gate.

- *Input wire*: If $w \in \{1, \ldots, \ell\}$ then it corresponds to the w-the input and $t = \mathsf{depth}(w) = 1$. Suppose that $x_w = f_w(x) = 1$. The user extracts $K_{w,1}, K_{w,2}$ from $\mathcal{K}_w$ included in its decryption key $\mathsf{SK}_{f,\mathsf{ID}}$ and computes
$$E_w = e(K_{w,1}, C)e(K_{w,2}, C'_w) = g_{n+m}^{r_w s} = g_{n+1+m-1}^{r_w s}.$$
- *OR gate*: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $t = \mathsf{depth}(w)$. Let $f_w(x) = 1$. Then the user checks whether $f_{\mathbb{A}(w)}(x) = 1$. If so, then it extracts $K_{w,1}, K_{w,3}$ from $\mathcal{K}_w$ contained in $\mathsf{SK}_{f,\mathsf{ID}}$ and computes
$$E_w = e(E_{\mathbb{A}(w)}, K_{w,1})e(K_{w,3}, C) = g_{n+t+m-1}^{r_w s}$$
Alternatively, if $f_{\mathbb{A}(w)}(x) = 0$, then it must hold that $f_{\mathbb{B}(w)}(x) = 1$ as $f_w(x) = 1$. In this case, it extracts $K_{w,2}, K_{w,4}$ from $\mathcal{K}_w$ in $\mathsf{SK}_{f,\mathsf{ID}}$ and computes
$$E_w = e(E_{\mathbb{B}(w)}, K_{w,2})e(K_{w,4}, C) = g_{n+t+m-1}^{r_w s}$$
- *AND gate*: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $t = \mathsf{depth}(w)$. Suppose that $f_w(x) = 1$. Then $f_{\mathbb{A}(w)}(x) = f_{\mathbb{B}(w)}(x) = 1$. The user extracts $K_{w,1}, K_{w,2}, K_{w,3}$ from $\mathcal{K}_w$ included in $\mathsf{SK}_{f,\mathsf{ID}}$ and computes
$$E_w = e(E_{\mathbb{A}(w)}, K_{w,1})e(E_{\mathbb{B}(w)}, K_{w,2})e(K_{w,3}, C) = g_{n+t+m-1}^{r_w s}.$$
Note that both $E_{\mathbb{A}(w)}$ and $E_{\mathbb{B}(w)}$ are already computed at this stage in the course of the bottom-up evaluation of the circuit as $\mathsf{depth}(\mathbb{A}(w)) = \mathsf{depth}(\mathbb{B}(w)) = t-1$.

At the end, the user computes $E_{\ell+q} = g_{\kappa}^{r_{\ell+q} s}$, as $f(x) = f_{\ell+q}(x) = 1$.

3. It determines the revoked user key index set $\mathsf{RI} \subseteq \mathcal{N}$ corresponding to RL using UL and obtains $\mathsf{SI} = \mathcal{N} \backslash \mathsf{RI}$. Note that since $\mathsf{ID} \notin \mathsf{RL}$, $u \in \mathsf{SI}$.
4. It computes $\vartheta'_u = g_m^{\alpha^{(u)}}$ and $\vartheta_{2^n-1-j+u} = g_{n-1}^{\alpha^{(2^n-1-j+u)}}$ for all $j \in \mathsf{SI}\backslash\{u\}$ using the ξ_ι values included in PP and multilinear map as follows:
   (a) (Computing ϑ'_u) Note that u can be expressed as a bit string of length n with $\mathsf{HW}(u) = m$ as $u \in \mathsf{SI} \subseteq \mathcal{N}$. Let $u = \sum_{\iota \in U} 2^\iota$ where $U = \{\iota'_1, \ldots, \iota'_m\} \subseteq \{0, \ldots, n-1\}$. It computes $\vartheta'_u = e(\xi_{\iota'_1}, \ldots, \xi_{\iota'_m}) = g_m^{\alpha^{(u)}}$.

   (b) (Computing $\vartheta_{2^n-1-j+u}$ for $j \in \mathsf{SI}\backslash\{u\}$) Let $2^n - 1 - j = \sum_{\iota \in \overline{J}} 2^\iota$ where $\overline{J} = \{\iota_1, \ldots, \iota_{n-m}\} \subseteq \{0, \ldots, n-1\}$ as earlier. Now U and $\overline{J}$ are disjoined only if $\overline{J} \cup U = \{0, \ldots, n-1\}$, i.e., $2^n - 1 - j + u = \sum_{\iota \in \overline{J}} 2^\iota + \sum_{\iota \in U} 2^\iota = \sum_{\iota=0}^{n-1} 2^\iota = 2^n - 1$, i.e., $j = u$. Since $j \neq u$, there must exist at least one $\widehat{\iota} \in \{0, \ldots, n-1\}$ such that $\widehat{\iota} \in \overline{J} \cap U$. Without loss of generality, let $\widehat{\iota} = \iota_{n-m} = \iota'_m$. Then $2^n - 1 - j + u = \sum_{\iota \in \overline{J}\backslash\{\iota_{n-m}\}} 2^\iota + \sum_{\iota \in U\backslash\{\iota'_m\}} 2^\iota +$

$2 \cdot 2^{\widehat{\iota}} = \sum_{\iota \in \overline{J} \backslash \{\widehat{\iota}\}} 2^{\iota} + \sum_{\iota \in U \backslash \{\widehat{\iota}\}} 2^{\iota} + 2^{\widehat{\iota}+1}$ where $[\overline{J} \backslash \{\widehat{\iota}\}] \cap [U \backslash \{\widehat{\iota}\}]$ may not be empty. It computes $\vartheta_{2^n-1-j+u}$ as

$$\vartheta_{2^n-1-j+u} = e(\xi_{\iota_1}, \ldots, \xi_{\iota_{n-m-1}}, \xi_{\iota'_1}, \ldots, \xi_{\iota'_{m-1}}, \xi_{\widehat{\iota}+1}) = g_{n-1}^{\alpha^{(2^n-1-j+u)}}.$$

Note that $\xi_{\widehat{\iota}+1}$ is extractable from PP since $\widehat{\iota} \in \{0, \ldots, n-1\}$.

5. Finally, utilizing the fact that $u \in \mathsf{SI}$, the user retrieves the message by the following computation:

$$C_M \widehat{E} E_{\ell+q} e\Big(\prod_{j \in \mathsf{SI} \backslash \{u\}} \vartheta_{2^n-1-j+u}, Z, C\Big) e\big(\vartheta'_u, Z, C''\big)^{-1} = M.$$

■ **Security:**

Theorem 2. $\mathsf{RABE\text{-}II}$ *is secure in the selective revocation list model against* CPA *as per the security model of Sect. 2.1 if the* (n, d, m)-cMDHE *assumption holds for the underlying multilinear group generator* $\mathcal{G}^{\mathsf{MLM}}$ *described in Sect. 2.2, such that* d *denotes the allowed depth of the decryption circuits, and* n, m *are two integers for which* $N_{\max} \leq \binom{n}{m}$*, where* $N_{\max}$ *is the maximum number of users supported by the system.*

The proof of Theorem 2 closely resembles that of Theorem 1 and is omitted here due to page restriction.

5 Efficiency

Both our RABE schemes permit general Boolean circuits of arbitrary polynomial size and unbounded fan-out with bounded depth and input length. This is the most expressive form of decryption policies accomplished for ABE till date [4,10]. We utilize the power of the multilinear map framework. All previous RABE constructions in the standard model [1–3,11,14], could support at most polynomial size monotone Boolean formulae because of the inherent limitation [10] of the traditional bilinear map setting underlying those schemes.

Another drawback of the previous standard model RABE schemes supporting direct revocation mode [1,2,14] is that they essentially utilize the tree-based revocation mechanism of Naor et al. [12]. As a result, the number of group elements comprising the revocation controlling segments of the ciphertexts and decryption keys are $O(\widehat{r} \log \frac{N_{\max}}{\widehat{r}})$ and $O(\log N_{\max})$ respectively, where $N_{\max}$ and $\widehat{r}$ denote respectively the maximum number of users supported by the system and the number of revoked users. Moreover, the number of group elements in the ABE realizing portion of the ciphertexts scales with the size of the attribute set or the complexity of the decryption policy associated to it. Our first RABE construction, $\mathsf{RABE\text{-}I}$, which is designed by carefully integrating the revocation strategy introduced in [5] with an improved variant of [10], features only 3 group elements in the ciphertexts. Furthermore, the number of decryption key components is $\ell + 4q + 1$ in the worst case, ℓ and q being respectively the input length and number of gates in the policy circuits. This is the same in all currently available multilinear map-based vanilla ABE constructions for general circuits [4,10].

This the added revocation functionality is attained without any extra overhead on the decryption keys.

One problem in RABE-I is that the number of PP elements is linear to $N_{\max}$ and, hence, the construction can accommodate only a small number of users. In our second RABE scheme, RABE-II, we attempt to reduce it by applying a more advanced revocation technique [6] so that we can support potentially large number of users. For RABE-II we consider $\kappa = n + d + m - 1$ such that $N_{\max} \leq \binom{n}{m}$ and the number of PP components becomes linear to n. As discussed in [6], a judicious choice of n and m would require $n \approx \log N_{\max}$. Therefore, the number of PP components reduces approximately to $\log N_{\max}$ in RABE-II. This is comparable to the best PP size attained by previous RABE constructions with direct revocation mode secure in the standard model [1,2,14]. Also, in RABE-II we need to provide only one component in PP in place of two in case of RABE-I corresponding to each input of the decryption policy circuits. However, observe that in this scheme also we could maintain the property that the number of ciphertext and decryption key components meant for revocation do not grow with $N_{\max}$. To the best of our knowledge, no previous RABE scheme with direct revocation could achieve such parameters.

Regarding computational complexity, note that the (worst case) number of multilinear operations involved in the setup, key generation, encryption, and decryption algorithms are respectively $2\ell+2N_{\max}+2$, $2\ell+4q+2$, 4, and $\ell+3q+4$ for RABE-I while $\ell + 2n + 5$, $4\ell + 4q + 3$, $\ell + 3$, and $2\ell + 3q + 3$ for RABE-II. Thus, we can see that RABE-II involves slightly more computation in the key generation, encryption and decryption procedures compared to RABE-I.

Remark 2. Note that a recent work [13] has applied the same revocation technique as ours [5,6] in the context of *identity-based encryption* (IBE). We emphasize that although IBE and ABE are related concepts, the richer functionality offered by the latter, especially when the access structures are highly expressive such as general polynomial-size circuits, poses significantly more challenges in enforcing revocation and necessitates more elegant techniques which we have developed in this work.

6 Conclusion

In this work, employing multilinear map [7–9], we have adopted a new technique [5,6] for enforcing direct user revocation in the context of ABE. Following that method, we have developed two selectively secure RABE schemes, both of which support decryption policies representable as general polynomial-size circuits as well as features very short ciphertexts without imposing any extra overhead in the decryption keys for the added revocation functionality. In our first construction, the size of the public parameters is linear to the maximum number of users supported by the system, while we have shrunk it to logarithmic in our second construction.

To the best of our knowledge, our work is the first in the literature which attained these features. Both our RABE constructions are proven secure in the selective revocation list model under reasonable assumptions.

References

1. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 278–300. Springer, Heidelberg (2009)
2. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 248–265. Springer, Heidelberg (2009)
3. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 417–426. ACM (2008)
4. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014)
5. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
6. Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 206–223. Springer, Heidelberg (2014)
7. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
8. Coron, J.S., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers (2015)
9. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
10. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)
11. Liang, X., Lu, R., Lin, X., Shen, X.S.: Ciphertext policy attribute based encryption with efficient revocation. Technical report, University of Waterloo (2010)
12. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
13. Park, S., Lee, K., Lee, D.H.: New constructions of revocable identity-based encryption from multilinear maps. IACR Cryptology ePrint Archive 2013, 880 (2013)
14. Qian, J., Dong, X.: Fully secure revocable attribute-based encryption. J. Shanghai Jiaotong Univ. (Sci.) **16**, 490–496 (2011)
15. Shi, Y., Zheng, Q., Liu, J., Han, Z.: Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. Inf. Sci. **295**, 221–231 (2015)
16. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 261–270. ACM (2010)