Registered ABE via Predicate Encodings, Revisited

Junichi Tomida

NTT Research tomida.junichi@gmail.com

Abstract. Predicate encodings, proposed by Wee (TCC'14), are a simple framework that abstracts a variety of attribute-based encryption (ABE) schemes. Zhu *et al.* showed how to apply predicate encodings to registered ABE (Reg-ABE) using the same definition as in vanilla ABE (Asiacrypt'23). We provide a new definition of predicate encodings that is more suitable for registered ABE and allows us to obtain more efficient Reg-ABE schemes than those by Zhu *et al.* for predicates captured by predicate encodings. Notably, we obtain

- the first Reg-ABE scheme for (non-)zero inner product with $O(\log L)$-size ciphertexts,
- the first Reg-ABE scheme for conjunctions/disjunctions with $O(\log L)$-size ciphertexts,
- the first registered broadcast encryption scheme with $O(\log L)$-size ciphertexts,

under the standard k-Lin assumption, where L denotes the maximum number of users. Prior to our work, Reg-ABE schemes for (non-)zero inner product, conjunctions, and disjunctions require $O(\ell \log L)$-size ciphertexts where ℓ is the vector length or the policy size, and registered broadcast encryption schemes rely on the generic group model.

Keywords: attribute-based encryption, registered attribute-based encryption, predicate encodings

Table of Contents

1 Introduction . . . 3
1.1 Our Results . . . 3
1.2 Technical Overview . . . 5
1.3 Open Problems . . . 9
2 Preliminaries . . . 10
2.1 Definitions . . . 10
3 sReg-ABE from Predicate Encodings . . . 12
3.1 Construction . . . 12
3.2 Security . . . 14
4 Concrete Predicate Encodings for Reg-ABE . . . 20
4.1 Identity-based Encryption . . . 20
4.2 Zero Inner Product . . . 20
4.3 Non-zero Inner Product . . . 21
4.4 Conjunctions . . . 21
4.5 Disjunctions . . . 21
4.6 Broadcast Encryption . . . 22
4.7 Boolean Span Programs . . . 22
4.8 Arithmetic Span Programs . . . 23
4.9 Summary of Encoding Sizes . . . 23
5 Instantiations of sReg-ABE . . . 23
5.1 sReg-ABE Scheme for Zero Inner Product (ZIPE) . . . 24
5.2 sReg-ABE Scheme for Non-zero Inner Product (NIPE) . . . 25
References . . . 25
A Registered ABE . . . 28

1 Introduction

Registered ABE. Registered attribute-based encryption (Reg-ABE) [HLWW23] is an emerging extension of attribute-based encryption (ABE) [GPSW06], designed to tackle the notorious key-escrow problem. In traditional ABE systems, a trusted authority needs to maintain a long-term master secret key msk to generate secret decryption keys as long as the system is in operation. However, the compromise of msk grants adversaries the ability to decrypt all ciphertexts within the system, thereby rendering the authority a single point of failure.

Reg-ABE presents a novel paradigm that introduces "key curator" instead of a trusted authority. Each user within a Reg-ABE system generates a pair of public and secret keys and registers the public key, along with its associated attribute y, with the key curator. The key curator then aggregates these pairs into a compact master public key (mpk) in a *verifiable and transparent manner*. A user in a Reg-ABE system encrypts a message with respect to a policy x using mpk to generate a ciphertext (ct). Decryption of the ciphertext is possible only for users possessing the attribute y satisfying the policy x with their secret key. More generally, decryption is allowed if and only if $\mathsf{P}(x, y) = 1$ for some predicate P. The decryption algorithms in Reg-ABE also require a helper secret key (hsk), which is a compact auxiliary information generated by the key curator. The point is that the key curator does not possess any secret, and its corruption does not affect the security of encrypted data.

The study of Reg-ABE began with registration-based encryption (RBE) [GHMR18], which can be conceptualized as Reg-ABE for the equality predicate $\mathsf{P}(x, y) = 1 \Leftrightarrow x = y$, to address the key escrow problem of identity-based encryption (IBE) [BF01]. The first RBE scheme [GHMR18] uses indistinguishability obfuscation (iO) [GGH+13][1], and lately RBE schemes relying on standard assumptions were proposed [GHM+19, GV20, CES21]. However, these schemes heavily rely on non-black-box use of cryptographic primitives, making them impractical. The non-black-box approach is also used for constructing Reg-ABE and more generalized registered functional encryption [FWW23, FFM+23, DPY23].

Reg-ABE via Predicate Encodings. Motivated by the inefficiency of non-black-box constructions, Reg-ABE (including RBE) schemes with only black-box use of cryptography have also been extensively studied, e.g., [HLWW23, DKL+23, GKMR23, FFM+23, FKdP23, ZZGQ23, GLWW24, AT24, LWW25, GHK+25, PS25, ZZZ+25]. Most of these constructions exhibit a drastic improvement of efficiency over the non-black box Reg-ABE schemes. Among these studies of black-box Reg-ABE, Zhu, Zhang, Gong, and Qian (ZZGQ) presented a framework to construct Reg-ABE schemes from predicate encodings [ZZGQ23].

Predicate encodings, introduced by Wee [Wee14], are information-theoretic primitives that extract the core structures of various ABE schemes based on pairings, such as IBE [BF01], inner-product encryption [KSW08], and ABE for Boolean/arithmetic span programs [GPSW06, IW14]. The crux of predicate encodings is that they drastically simplify the security analysis of ABE, that is, an encoding with simple information-theoretic security can be transformed into a fully secure ABE scheme via the compilers by [Wee14, CGW15]. Zhu *et al.* extended the compilers to the Reg-ABE setting in prime-order groups, which yielded new Reg-ABE schemes for predicates such as inner product encryption and arithmetic branching programs. Importantly, they used the same definition and constructions of predicate encodings as in vanilla ABE [Wee14, CGW15], while they observed that the Reg-ABE scheme for Boolean span programs (BSP) by Hohenberger, Lu, Waters, and Wu (HLWW) [HLWW23] is structurally more optimized than the Reg-ABE scheme for BSP via their predicate encoding framework. For this reason, they suggested the optimization potential of their framework but left it as a future work, which remains unsolved.

1.1 Our Results

To solve the aforementioned problem, we present a new definition of predicate encodings that is more suitable in the Reg-ABE setting and new constructions of predicate encodings following the new

[1] They also constructed a "weakly-efficient" RBE scheme from standard assumptions.

Table 1. Comparison between sReg-ABE schemes by the ZZGQ framework [ZZGQ23] and ours.

Predicate	$\lvert\mathsf{mpk}\rvert$		$\lvert\mathsf{hsk}\rvert$		$\lvert\mathsf{ct}\rvert$	
	ZZGQ	Ours	ZZGQ	Ours	ZZGQ	Ours
IBE	9	7	14	11	7	5
ZIPE	$3+2(\ell+2)$	$3+2(\ell+1)$	$5+3(\ell+2)$	$5+3(\ell+1)$	$3+2(\ell+1)$	5
NIPE	$3+2(\ell+3)$	$3+2\ell$	$5+3(\ell+3)$	$5+3\ell$	$3+2(\ell+2)$	5
BSP	$3+2(d+2\ell+1)$	$3+2(d+\ell)$	$5+3(d+2\ell+1)$	$5+3(d+\ell)$	$3+2(2\ell+1)$	$3+2\ell$
ASP	$3+2(d+3\ell+1)$	$3+2(d+2\ell)$	$5+3(d+3\ell+1)$	$5+3(d+2\ell)$	$3+2(3\ell+1)$	$3+4\ell$

Note: IBE, ZIPE, NIPE, BSP, ASP stand for identity-based encryption, zero inner product encryption, non-zero inner product encryption, Boolean span programs, and arithmetic span programs, respectively. The above figures refer to the number of group elements when we instantiate the scheme from the SXDH assumption (1-Lin). The columns of $\lvert\mathsf{mpk}\rvert$ and $\lvert\mathsf{ct}\rvert$ shows the number of G_1 elements while those for $\lvert\mathsf{hsk}\rvert$ show the number of G_2 elements. $\lvert\mathsf{mpk}\rvert$ and $\lvert\mathsf{ct}\rvert$ includes one G_T element in addition to the above figures. The parameter ℓ refers to the vector length in NIPE and ZIPE while it refers to the attribute size in BSP and ASP. The parameter d refers to the width of span programs.

Table 2. Comparison among pairing-based slotted Reg-ABE schemes for Boolean functions.

Reference	$\lvert\mathsf{crs}\rvert$	$\lvert\mathsf{mpk}\rvert$ and $\lvert\mathsf{hsk}\rvert$	$\lvert\mathsf{ct}\rvert$	Policy	Assumption	Prime order	Adaptive
[HLWW23]	$O(L^2\cdot\ell)$	$O(\ell)$	$O(\lvert f\rvert)$	MSP	SD	×	✓
[ZZGQ23]	$O(L^2\cdot\ell)$	$O(\ell)$	$O(\lvert f\rvert)$	MSP	k-Lin	✓	✓
[GLWW24]	$L^{1+o(1)}$	$O(\ell)$	$O(\lvert f\rvert)$	MSP	q-type	✓	×
[AT24]	$O(L^2)$	$\mathsf{poly}(\ell^*)$	$\mathsf{poly}(\ell^*)\cdot O(\lvert f\rvert)$	NSP	k-Lin	✓	✓
[GHK+25]	$O(L)$	$O(\ell)$	$O(\lvert f\rvert)$	DNF	GGM	✓	✓
[RT26]	$L^{1+o(1)}$	$O(\ell^*)$	$O(\lvert f\rvert)$	NSP	GGM + ROM	✓	✓
[SWW26]	$O(L)$	$O(\ell)$	$O(\lvert f\rvert)$	MSP	q-type	✓	×
Ours	$O(L^2)$	$O(\ell)$	$O(1)$	Conjunctions Disjunctions	k-Lin	✓	✓

Note: MSP, NSP, DNF, SD, GGM, and ROM stand for monotone span programs, non-monotone span programs, disjunctive normal form, subgroup decision, generic group model, and random oracle model, respectively. The element sizes refer to the number of group elements and hide $O(\lambda)$ factors. L denotes the maximum number of users in the system, ℓ denotes the size of attribute universe, and ℓ^* is the number of active attributes (i.e., attributes used in the system), and $\lvert f\rvert$ denotes the policy size. The column "Adaptive" indicates whether the scheme is adaptively secure. Regarding MSP, we do not improve the asymptotic efficiency over ZZGQ [ZZGQ23].

definition. We also provide a compiler from new predicate encodings to Reg-ABE schemes in the prime-order setting under the k-Lin assumption. Our new definition is as simple as the original one yet allows us to naturally capture the structure of the HLWW Reg-ABE scheme for BSP in contrast to the ZZGQ framework. Moreover, our new encodings and compiler provide more efficient Reg-ABE schemes than those by the ZZGQ framework in various predicates, which we demonstrate in Table 1. Note that we compare the efficiency of *slotted* Reg-ABE schemes in the table, which are a typical building block of full-fledged Reg-ABE schemes. Hohenberger *et al.* provided a generic compiler, called the powers-of-two construction, from slotted Reg-ABE to full-fledged Reg-ABE that incurs $O(\log L)$ multiplicative overhead in the sizes of $\mathsf{mpk}, \mathsf{hsk}, \mathsf{ct}$ [HLWW23], where L is the maximum number of users in the resulting scheme.

Notably, our framework provides

- the first slotted Reg-ABE scheme for (non-)zero inner product with $O(1)$-size ciphertexts,[2]
- the first slotted Reg-ABE scheme for conjunctions/disjunctions with $O(1)$-size ciphertexts, and

[2] Here and in what follows, the sizes refer to the number of group elements, i.e., they hide $\mathsf{poly}(\lambda)$ factors.

– the first slotted registered broadcast encryption (slotted Reg-BE) scheme with $O(1)$-size ciphertexts

under the standard k-Lin assumption. By the powers-of-two compiler, we obtain

– the first Reg-ABE scheme for (non-)zero inner product with $O(\log L)$-size ciphertexts,
– the first Reg-ABE scheme for conjunctions/disjunctions with $O(\log L)$-size ciphertexts, and
– the first registered broadcast encryption (Reg-BE) scheme with $O(\log L)$-size ciphertexts

under the same assumption. Prior to our work, Reg-ABE schemes for (non-)zero inner product and conjunctions/disjunctions require $O(\ell \log L)$-size ciphertexts [ZZGQ23, FFM+23] where ℓ is the vector length or the policy size, and the only known registered broadcast encryption scheme relies on the generic group model [GHK+25]. Note that our slotted Reg-ABE schemes for conjunctions/disjunctions are the first pairing-based ones for Boolean functions with constant-size ciphertexts as described in Table 2. As with all known pairing-based Reg-ABE schemes, Reg-ABE schemes based on our compiler support a priori bounded number of users.

(Non-)zero inner product is a fundamental predicate because it implies many other widely studied predicates, such as polynomial evaluation, conjunctions, disjunctions, broadcast encryption, etc. [KSW08]. In fact, our Reg-ABE schemes for conjunctions/disjunctions and Reg-BE scheme are direct implications from our Reg-ABE scheme for (non-)zero inner product.

Broadcast Encryption in Decentralized Model. Broadcast encryption [FN94] allows an encryptor to choose a set S of eligible users, and the ciphertext for S is decryptable by only the users with in S. Similar to ABE, Broadcast encryption is a centralized primitive that requires a trusted authority distributing user secret keys. Distributed broadcast encryption (DBE) [WQZD10, BZ14] is a variant of broadcast encryption that eliminates the requirement of the authority. Briefly, in DBE, each user generates its public and secret keys and posts the public key to a public bulletin board. An encryptor uses a set of public keys of eligible recipients. If we allow the ciphertext size to be linear in the broadcast size $|S|$, a DBE scheme can be trivially constructed from public key encryption. Thus, the size is required to be sublinear in $|S|$, while there is no requirement for public key compression, unlike in Reg-ABE. A DBE scheme with $O(1)$-size ciphertexts can be constructed from pairings [KMW23].

On the other hand, we can define Reg-BE as Reg-ABE for the broadcast predicate, i.e., $\mathsf{P}(S, i) = 1 \Leftrightarrow i \in S$, which is also a variant of broadcast encryption with decentralized key generation. Reg-BE requires the size of mpk to be polylogarithmic to the number L of users in addition to the size of ct being sublinear in the broadcast size $|S|$. It is straightforward to construct a Reg-BE scheme from a Reg-ABE scheme for zero inner product or Boolean span programs, but either construction suffered from at least $O(|S| \log L)$-size ciphertexts prior to our work [HLWW23, ZZGQ23, FFM+23]. Note that a DBE scheme with $O(\log L)$-size ciphertexts can generically be obtained from a Reg-BE scheme with even $O(|S| \log L)$-size ciphertexts [FWW23], and thus constructing Reg-BE schemes would be more challenging than constructing DBE schemes.

Garg, Hajiabadi, Kolonelos, Kothapalli, and Policharla (GHKKP) provided the first Reg-BE scheme with $O(\log L)$-size ciphertexts whose security relies on the generic group model [GHK+25].[3] The techniques behind our Reg-BE scheme are quite different from theirs. Specifically, GHKKP obtained their Reg-BE scheme as an implication from their registered threshold encryption with constant-size ciphertexts, which in turn is built on a special purpose witness encryption.

1.2 Technical Overview

sReg-ABE and Predicate Encodings. Since a full-fledged Reg-ABE scheme is obtained from a slotted Reg-ABE (sReg-ABE) scheme via the powers-of-two approach as mentioned above, we focus on

[3] Precisely, their scheme supports constant-size ciphertexts but does not support update efficiency, which is a standard requirement of registration-based cryptography. The update efficiency would be achievable via the powers-of-two compiler, which incurs the $O(\log L)$ multiplicative overhead.

sReg-ABE in what follows. First, we briefly recall sReg-ABE and predicate encodings. The sReg-ABE scheme is started by generating and publishing common reference string crs. Each user generates a pair of its public and secret keys $(\mathsf{pk}, \mathsf{sk})$ from crs. In sReg-ABE for a predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, the number L of users who can join the system is fixed in advance, and all users join the system all at once by registering their pk and key attribute $y \in \mathcal{Y}$. Then, the system generates a compact master public key mpk and helper secret keys hsk_i for user $i \in [L]$ in a deterministic manner. An encryptor takes mpk, a ciphertext attribute $x \in \mathcal{X}$, and a message M to generate ciphertext ct_x. Finally, ct_x can be decrypted with sk_i and hsk_i for user i if and only if $\mathsf{P}(x, y_i) = 1$.

A predicate encoding for $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is a set of matrices depending on ciphertext and key attributes $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Specifically, a (ω, n_c, n_k)-predicate encoding for P uniquely specifies matrices

$$\mathbf{a}_y \in \mathbb{Z}_p^{n_k}, \quad \mathbf{K}_y \in \mathbb{Z}_p^{\omega \times n_k}, \quad \mathbf{C}_x \in \mathbb{Z}_p^{\omega \times n_c}, \quad \mathbf{d}_{x,y} \in \mathbb{Z}_p^{n_k+n_c}, \quad \mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{a}_y & \mathbf{0} \\ \mathbf{K}_y & \mathbf{C}_x \end{pmatrix} \tag{1}$$

for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, which satisfies *decoding* correctness, *i.e.,* $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = (1, \mathbf{0})^\top$ if $\mathsf{P}(x, y) = 1$ and security, *i.e.,* the columns of $\mathbf{M}_{x,y}$ do not span $(1, \mathbf{0})^\top$ if $\mathsf{P}(x, y) = 0$ (the above formulation of predicate encodings follows [ZZGQ23]). For instance, a predicate encoding for identity-based encryption (IBE), *i.e.,* $\mathsf{P}(x, y) = 1 \Leftrightarrow x = y$, are given as $\mathbf{a}_y = 1, \mathbf{K}_y = (y, 1)^\top, \mathbf{C}_x = (x, 1)^\top, \mathbf{d}_{x,y} = (1, -1)$.

sReg-ABE from Predicate Encodings. Next, we recall the simplified ZZGQ sReg-ABE scheme from predicate encodings [ZZGQ23]. Let $e : G_1 \times G_2 \to G_T$ be bilinear groups, and $[\cdot]_i$ denotes element-wise exponentiation to $g_i \in G_i$. Their scheme is described as follows:

$$\begin{aligned}
\mathsf{crs} &= ([\alpha]_T,\ \{[w_{j,0}, \mathbf{w}_j]_1\}_{j\in[L]},\ \{[r_i, r_i w_{j,0}, r_i \mathbf{w}_j, r_i w_{i,0} + \alpha]_2\}_{i,j\in[L], i\neq j}) \\
\mathsf{pk}_i &= ([v_i]_1, \{[v_i r_j]_2\}_{j\neq i}), \quad \mathsf{sk}_i = v_i \\
\mathsf{mpk} &= ([\textstyle\sum_{j\in[L]}((w_{j,0} + v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j}), \sum_{j\in[L]} \mathbf{w}_j]_1,\ [\alpha]_T) \\
\mathsf{hsk}_i &= [r_i, \underbrace{r_i \textstyle\sum_{j\neq i}((w_{j,0} + v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j})}_{\mathbf{h}_{i,1}}, \underbrace{r_i \textstyle\sum_{j\neq i} \mathbf{w}_j}_{\mathbf{h}_{i,2}}, \underbrace{r_i w_{i,0} + \alpha}_{h_{i,3}}]_2 \\
\mathsf{ct}_x &= ([s, \underbrace{s \textstyle\sum_{j\in[L]}((w_{j,0} + v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j})}_{\mathbf{c}_1}, \underbrace{s \textstyle\sum_{j\in[L]} \mathbf{w}_j\mathbf{C}_x}_{\mathbf{c}_2}]_1, [s\alpha]_T M)
\end{aligned} \tag{2}$$

where $\alpha, w_{i,0}, r_i, v_i, s \leftarrow \mathbb{Z}_p, \mathbf{w}_j \leftarrow \mathbb{Z}_p^\omega$, and the system aggregates $(\mathsf{pk}_1, y_1), \ldots, (\mathsf{pk}_L, y_L)$ into mpk. In decryption for user i with attribute y_i, we compute

$$\begin{aligned}
&[(r_i\mathbf{c}_1 - s\mathbf{h}_{i,1} \,||\, r_i\mathbf{c}_2 - s\mathbf{h}_{i,2}\mathbf{C}_x)\mathbf{d}_{x,y_i}^\top - sh_{i,3} - sr_i v_i]_T \cdot [s\alpha]_T M \\
=&[sr_i((w_{i,0} + v_i)\mathbf{a}_{y_i} + \mathbf{w}_i\mathbf{K}_{y_i} \,||\, \mathbf{w}_i\mathbf{C}_x)\mathbf{d}_{x,y_i}^\top - sr_i(w_{i,0} + v_i) - s\alpha]_T \cdot [s\alpha]_T M \\
=&\underbrace{\left[sr_i(w_{i,0} + v_i \,||\, \mathbf{w}_i)\begin{pmatrix} \mathbf{a}_{y_i} & \mathbf{0} \\ \mathbf{K}_{y_i} & \mathbf{C}_x \end{pmatrix}\mathbf{d}_{x,y_i}^\top - sr_i(w_{i,0} + v_i) - s\alpha\right]_T}_{X} \cdot [s\alpha]_T M = M
\end{aligned} \tag{3}$$

where the third equality holds only if $\mathsf{P}(x, y_i) = 1$ from the correctness of predicate encodings. Let $\mathsf{mpk}_j = [(w_{j,0} + v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j}, \mathbf{w}_j]_1$. Then, by stretching the notation a bit, we can view $\mathsf{mpk} = \sum_j \mathsf{mpk}_j$. In essence, ct_x represents a ciphertext relative to $\sum_j \mathsf{mpk}_j$, and during decryption for user i, hsk_i enables us to strip ct_x of its association with $\sum_{j\neq i} \mathsf{mpk}_j$, retaining only the relationship with mpk_i. This adjustment ensures that the correctness of the ZZGQ sReg-ABE scheme rely primarily on predicate encodings, akin to vanilla ABE. The intuition for security is that the only way to efficiently compute the masking term $[s\alpha]_T$ from ct and crs (recall that mpk and hsk_i are publicly computable from crs) is to compute the expression X in Eq. (3) for some $i \in [L]$ with the knowledge of $\mathsf{sk}_i = v_i$. If $\mathsf{P}(x, y_i) = 0$, however, X includes monomials in $sr_i\mathbf{w}_i, sr_i w_{i,0}$ for all $\mathbf{d}_{x,y_i} \in \mathbb{Z}_p^{n_c+n_k}$, which prevents the adversary from learning $[s\alpha]_T$.

Predicate Encodings for Reg-ABE. The modification from the original definition to the new one for Reg-ABE is quite simple: just changing $\mathbf{M}_{x,y}$ in Eq. (1) to

$$\mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix} \mathbf{C}_x$$

Accordingly, the sizes of matrices are changed as follows:

$$\mathbf{a}_y \in \mathbb{Z}_p^{n_k}, \quad \mathbf{K}_y \in \mathbb{Z}_p^{\omega \times n_k}, \quad \mathbf{C}_x \in \mathbb{Z}_p^{n_k \times n_c}, \quad \mathbf{d}_{x,y} \in \mathbb{Z}_p^{n_c} \tag{4}$$

The correctness and the security of predicate encodings of Reg-ABE is the same as in the original one, i.e, $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = (1, \mathbf{0})^\top$ if $\mathsf{P}(x,y) = 1$ and the columns of $\mathbf{M}_{x,y}$ do not span $(1,\mathbf{0})^\top$ if $\mathsf{P}(x,y) = 0$. Our simplified compiler from a predicate encoding under the new definition to a sReg-ABE scheme is given as follows:

$$\begin{aligned}
\mathsf{crs} &= ([\alpha]_T,\ \{[w_{j,0}, \mathbf{w}_j]_1\}_{j\in[L]},\ \{[r_i, r_i w_{j,0}, r_i \mathbf{w}_j, r_i w_{i,0} + \alpha]_2\}_{i,j\in[L], i\neq j}) \\
\mathsf{pk}_i &= ([v_i]_1, \{[v_i r_j]_2\}_{j\neq i}), \quad \mathsf{sk}_i = v_i \\
\mathsf{mpk} &= ([\textstyle\sum_{j\in[L]}((w_{j,0}+v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j})]_1,\ [\alpha]_T) \\
\mathsf{hsk}_i &= [r_i, \underbrace{r_i \textstyle\sum_{j\neq i}((w_{j,0}+v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j})}_{\mathbf{h}_{i,1}}, \underbrace{r_i w_{i,0} + \alpha}_{h_{i,3}}]_2 \\
\mathsf{ct}_x &= ([s, \underbrace{s \textstyle\sum_{j\in[L]}((w_{j,0}+v_j)\mathbf{a}_{y_j} + \mathbf{w}_j\mathbf{K}_{y_j})\mathbf{C}_x}_{\mathbf{c}_1}]_1, [s\alpha]_T M)
\end{aligned} \tag{5}$$

Intuitively, our new definition captures the fact that an encryption algorithm can compute $\begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix} \mathbf{C}_x$ in the sReg-ABE setting rather than giving out $\begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix}$ and $\mathbf{C}_x$ separately as in the ZZGQ scheme (Eq. (2)). The differences from the ZZGQ scheme are that $\sum_{j\in[L]} \mathbf{w}_j$ is not in mpk, $\mathbf{h}_{i,2}$ is not in hsk_i, $\mathbf{c}_2$ is not in ct_x, and $\mathbf{c}_1$ is multiplied by $\mathbf{C}_x$.

Decryption is also similar to the ZZGQ scheme: for some $i \in [L]$ we compute

$$\begin{aligned}
&[(r_i\mathbf{c}_1 - s\mathbf{h}_{i,1}\mathbf{C}_x)\mathbf{d}_{x,y_i}^\top - sh_{i,3} - sr_i v_i]_T \cdot [s\alpha]_T M \\
&= \left[sr_i(w_{i,0} + v_i \,||\, \mathbf{w}_i) \begin{pmatrix} \mathbf{a}_{y_i} \\ \mathbf{K}_{y_i} \end{pmatrix} \mathbf{C}_x \mathbf{d}_{x,y_i}^\top - sr_i(w_{i,0}+v_i) - s\alpha\right]_T \cdot [s\alpha]_T M = M
\end{aligned} \tag{6}$$

The correctness and the security of our scheme follow from those of predicate encodings for Reg-ABE similarly to the ZZGQ scheme.

To see that our compiler is at least as good as the ZZGQ compiler, we show that for any predicate encoding Π, there exists a predicate encoding Π' for Reg-ABE such that the ZZGQ scheme (Eq. (2)) from Π is exactly the same as our scheme (Eq. (5)) from Π'. Given a (ω, n_c, n_k)-predicate encoding $\Pi = (\mathbf{a}_y, \mathbf{K}_y, \mathbf{C}_x, \mathbf{d}_{x,y})$, we define Π' as

$$\begin{aligned}
&\mathbf{a}'_y = (\mathbf{a}_y || \mathbf{0}) \in \mathbb{Z}_p^{n_k+\omega}, \quad \mathbf{K}'_y = (\mathbf{K}_y || \mathbf{I}_\omega) \in \mathbb{Z}_p^{\omega \times (n_k+\omega)} \\
&\mathbf{C}'_x = \begin{pmatrix} \mathbf{I}_{n_k} & \\ & \mathbf{C}_x \end{pmatrix} \in \mathbb{Z}_p^{(n_k+\omega)\times(n_k+n_c)}, \quad \mathbf{d}'_{x,y} = \mathbf{d}_{x,y}
\end{aligned} \tag{7}$$

Observe that the resulting schemes are identical. Since we have

$$\mathbf{M}'_{x,y} = \begin{pmatrix} \mathbf{a}'_y \\ \mathbf{K}'_y \end{pmatrix} \mathbf{C}'_x = \begin{pmatrix} \mathbf{a}_y & \mathbf{0} \\ \mathbf{K}_y & \mathbf{C}_x \end{pmatrix} = \mathbf{M}_{x,y}$$

the correctness and security of Π' directly follows from those of Π. This also means that predicate encodings for Reg-ABE is as general as the original predicate encodings in the sense that any predicate captured by predicate encodings is also captured by predicate encodings for Reg-ABE.

New Encodings: ZIPE. Our definition allows us to construct more efficient novel encodings than the trivial ones converted from the original encodings as described in the previous paragraph. These include encodings for IBE, (non-)zero inner-product encryption, and arithmetic span programs. Especially, the most interesting encodings are those for (non-)zero inner product encryption since they yield the first sReg-ABE scheme with constant-size ciphertexts. Therefore, we briefly introduce our encoding for zero inner product encryption (ZIPE) in this overview.

The predicate $\mathsf{P}^{\mathrm{ZIPE}} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ for ZIPE is defined as $\mathsf{P}^{\mathrm{ZIPE}}(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle = 0$ where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^\ell$. The original encoding for ZIPE is described as follows:

$$\mathbf{a}_y = (1), \quad \mathbf{K}_y = \begin{pmatrix} 0 \\ \mathbf{y}^\top \end{pmatrix}, \quad \mathbf{C}_x = \begin{pmatrix} \mathbf{x} \\ \mathbf{I}_\ell \end{pmatrix}, \quad \mathbf{d}_{x,y} = (1 || -\mathbf{y})$$

By the naive transformation in Eq. (7), we can obtain a predicate encoding for Reg-ABE as

$$\mathbf{a}'_y = (1||\mathbf{0}), \quad \mathbf{K}'_y = \begin{pmatrix} 0 & 1 \\ \mathbf{y}^\top & \\ & \mathbf{I}_{\ell-1} \end{pmatrix}, \quad \mathbf{C}'_x = \begin{pmatrix} 1 & \\ \mathbf{x} & \\ & \mathbf{I}_\ell \end{pmatrix}, \quad \mathbf{d}'_{x,y} = (1 || -\mathbf{y})$$

On the other hand, our new encoding for $\mathsf{P}^{\mathrm{ZIPE}}$ is described as follows:

$$\mathbf{a}'_y = (1||\mathbf{0}), \quad \mathbf{K}'_y = (0, \mathbf{y}), \quad \mathbf{C}'_x = \begin{pmatrix} 1 \\ \mathbf{x}^\top \end{pmatrix}, \quad \mathbf{d}'_{x,y} = (1)$$

It is not hard to see that

$$\mathbf{M}'_{x,y} = \begin{pmatrix} \mathbf{a}'_y \\ \mathbf{K}'_y \end{pmatrix} \mathbf{C}'_x = (1, \langle \mathbf{x}, \mathbf{y} \rangle)^\top$$

Therefore, our new encoding is correct and secure. Since the ciphertext size of our scheme in Eq. (5) is $O(n_c)$, where n_c is the width of $\mathbf{C}_x$, we obtain the sReg-ABE scheme for ZIPE with constant-size ciphertexts from the above encoding.

The predicates for conjunctions $\mathsf{P}^{\mathrm{AND}}(X, Y) = 1 \Leftrightarrow X \subseteq Y$ and broadcast encryption $\mathsf{P}^{\mathrm{BE}}(S, i) = 1 \Leftrightarrow i \in S$ can be embedded in $\mathsf{P}^{\mathrm{ZIPE}}$ with standard techniques. Therefore, our sReg-ABE scheme for ZIPE immediately implies the sReg-ABE schemes for conjunctions and broadcast encryption with constant-size ciphertexts (see Sections 4.4 and 4.6 for details). Similarly, the predicate for disjunctions $\mathsf{P}^{\mathrm{OR}}(X, Y) = 1 \Leftrightarrow X \cap Y \neq \emptyset$ can be embedded in $\mathsf{P}^{\mathrm{NIPE}}$. Therefore, our sReg-ABE scheme for NIPE immediately implies the sReg-ABE scheme for disjunctions with constant-size ciphertexts (see Section 4.5 for details).

sReg-ABE for Boolean Span Programs by HLWW. Predicate encodings for Reg-ABE naturally capture the optimized structure of the HLWW sReg-ABE scheme for BSP [HLWW23]. First, we briefly recall the predicate $\mathsf{P}^{\mathrm{BSP}} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ for (read-once) Boolean span programs. In $\mathsf{P}^{\mathrm{BSP}}$, $\mathcal{X}$ consists of span programs $\mathbf{X} \in \mathbb{Z}_p^{d \times \ell}$, and $\mathcal{Y}$ consists of bitstrings $\mathbf{y} \in \{0,1\}^\ell$. Then, we define $\mathsf{P}^{\mathrm{BSP}}(\mathbf{X}, \mathbf{y}) = 1$ if and only if $(1, \mathbf{0}) \in \mathsf{span}(\mathbf{x}_i)_{i:y_i=1}$ where $\mathbf{x}_i$ is the i-th row of $\mathbf{X}$.

In a typical vanilla ciphertext-policy ABE scheme for BSP, e.g., [LOS+10], we have terms like

$$\mathsf{sk} : r, \{y_i r w_i\}_{i \in [\ell]}, \quad \mathsf{ct} : s, \{s(\mathbf{x}_i \mathbf{v}^\top + w_i)\}_{i \in [\ell]}$$

where y_i is the i-th element of $\mathbf{y}$, $\mathbf{v} = (v_1, \tilde{\mathbf{v}})$, v_1 is some secret value, and $\tilde{\mathbf{v}}$ is share generating randomness. In the context of sReg-ABE for the single user case ($L = 1$), this encoding is translated as

$$\mathsf{mpk} : \mathbf{v}, \{y_i w_i\}_{i \in [\ell]} \quad \mathsf{ct} : s, \{s(\mathbf{x}_i \mathbf{v}^\top + w_i), s y_i w_i\}_{i \in [\ell]}$$

which is basically the encoding used in the ZZGQ framework [ZZGQ23]. On the other hand, the HLWW scheme uses the following optimized encoding:[4]

$$\mathsf{mpk} : \mathbf{v}, \{(1-y_i)w_i\}_{i\in[\ell]} \quad \mathsf{ct} : s, \{s(\mathbf{x}_i\mathbf{v}^\top + (1-y_i)w_i)\}_{i\in[\ell]}$$

As discussed in [ZZGQ23, Section 1.4], the latter encoding cannot be captured by the original predicate encoding framework. Intuitively, this is because ct in the HLWW scheme includes the term that depends on both the key attribute $\mathbf{y}$ and the ciphertext attribute $\mathbf{X}$, while the syntax of the original predicate encodings does not allow terms depending on both $\mathbf{y}$ and $\mathbf{X}$. In contrast, our framework can capture it as

$$\mathsf{mpk} : (\mathbf{v}, \mathbf{w}) \underbrace{\begin{pmatrix} \mathbf{I}_d & \\ & \mathsf{diag}(\bar{\mathbf{y}}) \end{pmatrix}}_{\begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix}}, \quad \mathsf{ct} : s, s(\mathbf{v}, \mathbf{w}) \underbrace{\begin{pmatrix} \mathbf{I}_d & \\ & \mathsf{diag}(\bar{\mathbf{y}}) \end{pmatrix}}_{\begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix}} \underbrace{\begin{pmatrix} \mathbf{X}^\top \\ \mathbf{I}_\ell \end{pmatrix}}_{\mathbf{C}_x}$$

where $\mathsf{diag}(\bar{\mathbf{y}}) = \begin{pmatrix} 1-y_1 & & \\ & \ddots & \\ & & 1-y_\ell \end{pmatrix}$.

Prime-order Scheme. Our full-fledged compiler (given in Section 3.1) is obtained by the standard dual system technique in prime-order groups [CGW15, ZZGQ23]. Specifically, we use the following substitution to the simplified scheme in Eq. (5), which basically yields our compiler:

$$\begin{aligned}
&\alpha \mapsto \mathbf{h}^\top, \quad w_{j,0} \mapsto \mathbf{A}\mathbf{W}_{j,0}, \quad \mathbf{w}_j \mapsto \mathbf{A}\mathbf{W}_j, \quad r_i \mapsto \mathbf{B}\mathbf{r}_i^\top \\
&r_i w_{j,0} \mapsto \mathbf{W}_{j,0}\mathbf{B}\mathbf{r}_i, \quad r_i\mathbf{w}_j \mapsto \mathbf{W}_{j,0}(\mathbf{I}_\omega \otimes \mathbf{B}\mathbf{r}_i^\top), \quad v_i \mapsto \mathbf{A}\mathbf{V}_i, \quad v_i r_j \mapsto \mathbf{V}_i\mathbf{B}\mathbf{r}_j \\
&s \mapsto \mathbf{s}\mathbf{A}, \quad s v_i \mapsto \mathbf{s}\mathbf{A}\mathbf{V}_i, \quad s w_{j,0} \mapsto \mathbf{s}\mathbf{A}\mathbf{W}_{j,0}, \quad s\mathbf{w}_j \mapsto \mathbf{s}\mathbf{A}\mathbf{W}_j
\end{aligned}$$

where $\mathbf{h} \leftarrow \mathbb{Z}_p^{2k+1}$, $\mathbf{W}_{j,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}$, $\mathbf{W}_j \leftarrow \mathbb{Z}_p^{(2k+1)\times\omega(k+1)}$, $\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$, $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, $\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}$, $\mathbf{s} \leftarrow \mathbb{Z}_p^k$, $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(2k+1)}$. The security proof of our full-fledged scheme mostly follows that of the ZZGQ scheme, and we refrain from describing the details in the overview.

1.3 Open Problems

We left several open problems.

- We provide the first slotted Reg-ABE schemes for conjunctions/disjunctions with constant size ciphertexts from pairings. A natural question is whether we can obtain slotted Reg-ABE schemes for broader classes of Boolean functions such as DNF, formulas, and span programs with the ciphertext size being constant. For vanilla ABE, Agrawal and Chase presented an ABE scheme for span programs with constant size ciphertexts [AC16].
- Our idea to define predicate encodings that are more suitable to the Reg-ABE setting seems applicable to more general pair encodings [Att14]. Simplifying the pair encoding framework in the Reg-ABE setting by [AT24] with this idea is an interesting future direction.
- As we saw in the overview, the original predicate encodings can be transformed in to those for Reg-ABE. However, the resulting encodings are not optimized. Developing encoding transformations that optimize predicate encodings for Reg-ABE is also an interesting problem. Such a transformation is studied by [ABS17] for original predicate encodings.

[4] Precisely, in the HLWW scheme, the encryption algorithm samples $\tilde{\mathbf{v}}$. The predicate encoding framework sometimes substitutes such $\tilde{\mathbf{v}}$ with $s\tilde{\mathbf{v}}'$, where $\tilde{\mathbf{v}}'$ is a part of the master public key to simplify the predicate encoding syntax [CGW15].

2 Preliminaries

Notations. For $n, m \in \mathbb{N}$, $[m]$ and $[n, m]$ denotes $\{1, \ldots, m\}$ and $\{n, \ldots, m\}$, respectively. $\mathbf{O}$ and $\mathbf{0}$ denotes a zero matrix and a zero vector (with sizes corresponding to the context). For set S, $s \leftarrow S$ means that s is uniformly chosen from S. For two families of distributions $A = \{A_\lambda\}, B = \{B_\lambda\}$, $A \approx_c B$ and $A \approx_s B$ mean A and B are computationally and statistically indistinguishable, respectively. For a matrix $\mathbf{A} = (a_{j,\ell})_{j,\ell}$ over $\mathbb{Z}_p$, $[\mathbf{A}]_i$ denotes a matrix over G_i whose (j, ℓ)-th entry is $g_i^{a_{j,\ell}}$. For matrices $\mathbf{A}, \mathbf{B}$ over $\mathbb{Z}_p$ such that $\mathbf{AB}$ is defined, $e([\mathbf{A}]_1, [\mathbf{B}]_2)$ denotes the pairing operation that computes $[\mathbf{AB}]_T$.

2.1 Definitions

Definition 2.1 (Bilinear Groups). Let $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of bilinear groups. Bilinear groups $\mathbb{G}_\lambda = (p, G_1, G_2, G_T, g_1, g_2, e)$ are specified by a prime p, cyclic groups G_1, G_2, G_T of order p, generators g_1 and g_2 of G_1 and G_2 respectively, and a bilinear map $e : G_1 \times G_2 \to G_T$, which has two properties.

- (Bilinearity): $\forall h_1 \in G_1, h_2 \in G_2, a, b \in \mathbb{Z}_p, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- (Non-degeneracy): For g_1 and g_2, $g_T = e(g_1, g_2)$ is a generator of G_T.

In what follows, we omit the index λ from $\mathbb{G}_\lambda$ and abuse notation by denoting a family of bilinear groups $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ also by $\mathbb{G}$ if it is clear in the context.

Definition 2.2 (Matrix Decisional Diffie-Hellman (MDDH) Assumption [EHK+13]). Let $\{\mathbb{G}\}$ be a family of bilinear groups. We consider the following distribution, for any $n > k$ and $m \in \mathbb{N}$: $\mathbf{M} \leftarrow \mathbb{Z}_p^{k \times n}, \mathbf{R} \leftarrow \mathbb{Z}_p^{m \times k}, \mathbf{Z}_0 = \mathbf{RM}, \mathbf{Z}_1 \leftarrow \mathbb{Z}_p^{m \times n}, P_{i,\beta} = (\mathbb{G}, [\mathbf{M}]_i, [\mathbf{Z}_\beta]_i)$. We say that the MDDH_k assumption holds with respect to $\{\mathbb{G}\}$ if $P_{i,0} \approx_c P_{i,1}$ for $i \in \{1, 2\}$. We often omit k and just say the MDDH assumption. The k-Lin assumption implies the MDDH_k assumption.

Definition 2.3 (QA-NIZK [JR13]). Quasi-adaptive non-interactive zero knowledge argument (QA-NIZK) for linear space over bilinear groups $\mathbb{G}$ consists of the four algorithms.

$\mathsf{LGen}(1^\lambda, [\mathbf{A}]_1)$**:** It takes a security parameter 1^λ and $[\mathbf{A}]_1 \in G_1^{n \times m}$ as input and outputs a common reference string crs and a trapdoor td.

$\mathsf{LProve}(\mathsf{crs}, [\mathbf{M}]_1, \mathbf{V})$**:** It takes crs, $[\mathbf{M}]_1 \in G_1^{n \times \ell}$ with a witness $\mathbf{V} \in \mathbb{Z}_p^{m \times \ell}$ as input and outputs a proof π.

$\mathsf{LVerify}(\mathsf{crs}, [\mathbf{M}]_1, \pi)$**:** It takes crs, $[\mathbf{M}]_1, \pi$ as input and outputs $\beta \in \{0, 1\}$.

$\mathsf{LSim}(\mathsf{crs}, \mathsf{td}, [\mathbf{M}]_1)$**:** It takes $\mathsf{crs}, \mathsf{td}, [\mathbf{M}]_1$ and outputs a simulated proof $\tilde{\pi}$.

Perfect Completeness. For all $n, m, \ell, \lambda, \mathbf{A}, \mathbf{V}, \mathbf{M}$ such that $\mathbf{M} = \mathbf{AV}$,

$$\Pr\left[\mathsf{LVerify}(\mathsf{crs}, [\mathbf{M}]_1, \pi) = 1 \ : \ \begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}]_1) \\ \pi \leftarrow \mathsf{LProve}(\mathsf{crs}, [\mathbf{M}]_1, \mathbf{V})\end{array}\right] = 1$$

Perfect Zero-knowledge. For all $n, m, \ell, \lambda, \mathbf{A}, \mathbf{V}, \mathbf{M}$ such that $\mathbf{M} = \mathbf{AV}$, $\mathsf{LProve}(\mathsf{crs}, [\mathbf{M}]_1, \mathbf{V})$ and $\mathsf{LSim}(\mathsf{crs}, \mathsf{td}, [\mathbf{M}]_1)$ are identically distributed where $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}]_1)$.

Stronger Unbounded Simulation Soundness. For all PPT adversaries $\mathcal{A}$ and $n, m \in \mathbb{N}$, the following advantage of $\mathcal{A}$ is negligible in λ:

$$\Pr\left[\begin{array}{ll} ([\mathbf{M}^*]_1, \pi^*) \notin \mathcal{L} & \mathbf{A} \leftarrow \mathbb{Z}_p^{n \times m} \\ \wedge \nexists \mathbf{V}, \mathbf{M}^* = \mathbf{AV} & : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}]_1) \\ \wedge \mathsf{LVerify}(\mathsf{crs}, [\mathbf{M}^*]_1, \pi^*) = 1 & ([\mathbf{M}^*]_1, \pi^*) \leftarrow \mathcal{A}^{\mathsf{LSim}(\mathsf{crs}, \mathsf{td}, \cdot)}(1^\lambda, \mathsf{crs}, \mathbf{A}) \end{array}\right]$$

where $\mathcal{L}$ is a list of pairs of $\mathcal{A}$'s query to LSim and the corresponding response.

A QA-NIZK scheme that satisfies the above properties from MDDH is given in [KW15] (see also [ZZGQ23, Appendix B]).

We now describe the definition of *bounded-user* slotted registered ABE, which we refer to as sReg-ABE throughout the paper. We follow the notation in [ZZGQ23] (which, in turn, follows [HLWW23]). For compactness, we follow the requirement of large-universe schemes in [HLWW23]. The definition for full-fledged registered attribute-based encryption (RegABE) is deferred to Appendix A. It is shown in [HLWW23] how to generically convert (bounded-user or unbounded-user) slotted registered ABE to registered ABE while preserving the (un)bounded-user property and compactness. The size expansions of $|\mathsf{mpk}|$, $|\mathsf{hsk}_i|$, $|\mathsf{ct}|$ by the conversion are logarithmic in the maximum number of registered users.

Definition 2.4 (Bounded-User Slotted Registered ABE [HLWW23]). Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y}$ be a predicate. Let $\mathcal{M}$ be a message space. A bounded-user slotted registered attribute-based encryption (sReg-ABE) scheme for P consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^L)$**:** It takes a security parameter 1^λ, the number of slots 1^L and outputs a common reference string crs.[5]

$\mathsf{Gen}(\mathsf{crs}, i)$**:** It takes crs and an index $i \in [L]$ as input, and outputs a public key pk_i and a secret key sk_i.

$\mathsf{Ver}(\mathsf{crs}, i, \mathsf{pk}_i)$**:** It takes $\mathsf{crs}, i, \mathsf{pk}_i$ as input, and outputs 1 if pk_i is valid, and 0 otherwise.

$\mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_i, y_i\}_{i\in[L]})$**:** It takes crs, a set of pairs pk_i, and $y_i \in \mathcal{Y}$ for $i \in [L]$ as input, and outputs a master public key mpk and a set of helper keys hsk_i for $i \in [L]$. This algorithm is deterministic.

$\mathsf{Enc}(\mathsf{mpk}, x, M)$**:** It takes mpk, $x \in \mathcal{X}$, and a message $M \in \mathcal{M}$ as inputs, and outputs a ciphertext ct_x.

$\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x, x, y_i)$**:** It takes $\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x, x, y_i$ as input, and outputs a message M or a symbol $\perp$.

Throughout the paper, we only consider the bounded-user setting and often omit the modifier 'bounded-user'.

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

$$\Pr[\mathsf{Ver}(\mathsf{crs}, i, \mathsf{pk}_i) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^L); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, i)] = 1$$

Correctness. For all $\lambda, L \in \mathbb{N}, i \in [L]$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$, $\{\mathsf{pk}_j\}_{j\in[L]\setminus\{i\}}$ such that $\mathsf{Ver}(\mathsf{crs}, j, \mathsf{pk}_j) = 1$, $x \in \mathcal{X}$, $y_1, \ldots, y_L \in \mathcal{Y}$ such that $\mathsf{P}(x, y_i) = 1$, and $M \in \mathcal{M}$, we have

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x) = M \;:\; \begin{array}{l}(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, i)\\ (\mathsf{mpk}, \{\mathsf{hsk}_j\}_{j\in[L]}) \leftarrow \mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_j, y_j\}_j)\\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, M)\end{array}\right] = 1$$

Compactness. For all $\lambda, L \in \mathbb{N}, i \in [L]$, the sizes of mpk and hsk_i obtained from $\mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_i, y_i\}_{i\in[L]})$ are $\mathsf{poly}(\lambda, \log|\mathcal{Y}|, \log L)$.

Security. For all stateful admissible adversaries $\mathcal{A}$, the following advantage $\mathsf{Adv}^{\mathsf{sRegABE}}_{\mathcal{A}}(\lambda)$ is negligible in λ:

$$\Pr\left[\beta = \beta' \;:\; \begin{array}{l}L \leftarrow \mathcal{A}(1^\lambda);\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^L)\\ \{\mathsf{pk}^*_i, y_i\}_{i\in[L]}, x, M_0, M_1 \leftarrow \mathcal{A}^{\mathsf{OGen}(\cdot),\mathsf{OCor}(\cdot)}(\mathsf{crs})\\ (\mathsf{mpk}, \{\mathsf{hsk}_i\}_{i\in[L]}) \leftarrow \mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}^*_i, y_i\}_{i\in[L]})\\ \beta \leftarrow \{0,1\};\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, M_\beta);\ \beta' \leftarrow \mathcal{A}(\mathsf{ct}_x)\end{array}\right] - \frac{1}{2}$$

where $\mathcal{D}_i$ and $\mathcal{C}$ are dictionaries that are initially empty, $\mathsf{OGen}(i)$ runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}, i)$, set $\mathcal{D}_i = \mathcal{D}_i \cup \{(\mathsf{pk}, \mathsf{sk})\}$ and returns pk, and $\mathsf{OCor}(i, \mathsf{pk})$ returns sk if $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{D}_i$ (returns $\perp$ otherwise) and set $\mathcal{C} = \mathcal{C} \cup \{(i, \mathsf{pk})\}$. $\mathcal{A}$ is admissible if its queries satisfy

$$(\mathsf{pk}^*_i, *) \notin \mathcal{D}_i \Rightarrow \mathsf{Ver}(\mathsf{crs}, i, \mathsf{pk}) = 1$$
$$(i, \mathsf{pk}^*_i) \in \mathcal{C} \vee (\mathsf{pk}^*_i, *) \notin \mathcal{D}_i \Rightarrow \mathsf{P}(x, y_i) = 0$$

[5] In *unbounded-user* schemes, Setup takes L in binary instead of unary and is efficient even if we set $L = O(2^\lambda)$; see [HLWW23].

Definition 2.5 (Predicate Encodings [Wee14]). The original (ω, n_c, n_k)-predicate encoding for $\mathsf{P}: \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is the matrix system defined as follows. For every $x \in \mathcal{X}, y \in \mathcal{Y}$, there exist

$$\mathbf{a}_y \in \mathbb{Z}_p^{n_k},\ \ \mathbf{K}_y \in \mathbb{Z}_p^{\omega \times n_k},\ \ \mathbf{C}_x \in \mathbb{Z}_p^{\omega \times n_c},\ \ \mathbf{d}_{x,y} \in \mathbb{Z}_p^{n_k+n_c},\ \ \mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{a}_y & \mathbf{0} \\ \mathbf{K}_y & \mathbf{C}_x \end{pmatrix}$$

which satisfy the following two properties:

Correctness: For all (x, y) such that $\mathsf{P}(x, y) = 1$, we have $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = (1, \mathbf{0})^\top$.
Security: For all (x, y) such that $\mathsf{P}(x, y) = 0$ and $\alpha \in \mathbb{Z}_p$, the following distributions are statistically close over $\mathbf{w} \leftarrow \mathbb{Z}_p^\omega$:

$$\{\alpha, \mathbf{M}_{x,y}, (\alpha, \mathbf{w})\mathbf{M}_{x,y}\} \quad \text{and} \quad \{\alpha, \mathbf{M}_{x,y}, (0, \mathbf{w})\mathbf{M}_{x,y}\}$$

It is not hard to see that the security definition is equivalent to the following statement: the columns of $\mathbf{M}_{x,y}$ do not span $(1, \mathbf{0})^\top$.

We present our new definition of predicate encodings that is more suitable to construct Reg-ABE schemes.

Definition 2.6 (Predicate Encodings for Reg-ABE). A (ω, n_c, n_k)-predicate encoding *for Reg-ABE* with respect to $\mathsf{P}: \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is defined as follows. For every $x \in \mathcal{X}, y \in \mathcal{Y}$, there exist

$$\mathbf{a}_y \in \mathbb{Z}_p^{n_k},\ \ \mathbf{K}_y \in \mathbb{Z}_p^{\omega \times n_k},\ \ \mathbf{C}_x \in \mathbb{Z}_p^{n_k \times n_c},\ \ \mathbf{d}_{x,y} \in \mathbb{Z}_p^{n_c},\ \ \mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{a}_y \\ \mathbf{K}_y \end{pmatrix} \mathbf{C}_x$$

which satisfy correctness and security in Definition 2.5 and compactness:[6]

Compactness: We have $n_k = O(\log |\mathcal{Y}|)$.

The difference of matrices from the original definition in Definition 2.5 is the sizes of $\mathbf{C}_x$, $\mathbf{d}_{x,y}$ and the definition of $\mathbf{M}_{x,y}$.

3 sReg-ABE from Predicate Encodings

In this section, we show how to construct sReg-ABE scheme from predicate encodings in Definition 2.6. We provide concrete constructions of predicate encodings in Section 4 and some instantiations of sReg-ABE schemes for specific predicates in Section 5.

3.1 Construction

Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of bilinear groups where $\mathbb{G}_\lambda = (p, G_1, G_2, G_T, g_1, g_2, e)$. Let $\Pi = (\mathsf{LGen}, \mathsf{LProve}, \mathsf{LVerify}, \mathsf{LSim})$ be a QA-NIZK scheme. Then, we can construct a sReg-ABE scheme for predicate P from a (ω, n_c, n_k)-predicate encoding for P as follows.

$\mathsf{Setup}(1^\lambda, 1^L)$**:** It outputs crs as follows.

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (2k+1)},\ \mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1) \times k},\ \mathbf{h} \leftarrow \mathbb{Z}_p^{2k+1}$$

$$\left\{ \begin{array}{l} \mathbf{W}_i = (\mathbf{W}_{i,1} || \cdots || \mathbf{W}_{i,\omega}) \leftarrow \mathbb{Z}_p^{(2k+1) \times \omega(k+1)},\ \mathbf{W}_{i,0} \leftarrow \mathbb{Z}_p^{(2k+1) \times (k+1)} \\ \mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+1)},\ \mathbf{r}_i \leftarrow \mathbb{Z}_p^k \\ (\mathsf{crs}_i, \mathsf{td}_i) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}_i]_1) \ \text{ where } \mathbf{A}_i = \begin{pmatrix} \mathbf{A} \\ \mathbf{R}_i \end{pmatrix} \end{array} \right\}_{i \in [L]}$$

$$\mathsf{crs} = \left(\begin{array}{l} [\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{A}\mathbf{W}_{i,0}, \mathbf{A}\mathbf{W}_i]_1\}_{i \in [L]} \\ \{[\mathbf{B}\mathbf{r}_i^\top, \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top]_2\}_{i \in [L]}, \{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top, \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{B}\mathbf{r}_j^\top)]_2\}_{\substack{i,j \in [L] \\ i \neq j}} \end{array} \right)$$

[6] Zhu *et al.* defined the compactness of Reg-ABE depending on predicate encodings, i.e., they allow $|\mathsf{mpk}|$ and $|\mathsf{hsk}_i|$ to be $\mathsf{poly}(\omega, n_k)$ instead of $\mathsf{poly}(\log |\mathcal{Y}|)$ [ZZGQ23]. If we follow this definition of compactness, we do not need the compactness for predicate encodings. However, we choose to follow the original definition of compactness by [HLWW23] and thus need the compactness for predicate encodings so that the resulting sReg-ABE scheme achieve compactness as per Definition 2.4.

$\mathsf{Gen}(\mathsf{crs}, i)$**:** It outputs pk_i and sk_i as follows:

$$\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)},\ \mathbf{M}_i = \begin{pmatrix} \mathbf{AV}_i \\ \mathbf{R}_i\mathbf{V}_i \end{pmatrix},\ \pi_i \leftarrow \mathsf{LProve}(\mathsf{crs}_i, [\mathbf{M}_i]_1, \mathbf{V}_i)$$
$$\mathsf{pk}_i = ([\underbrace{\mathbf{AV}_i}_{\mathbf{T}_i}, \underbrace{\mathbf{R}_i\mathbf{V}_i}_{\mathbf{Q}_i}]_1, \{[\underbrace{\mathbf{V}_i\mathbf{Br}_j^\top}_{\mathbf{p}_{i,j}^\top}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i),\ \ \mathsf{sk}_i = \mathbf{V}_i$$

$\mathsf{Ver}(\mathsf{crs}, i, \mathsf{pk}_i)$**:** It parses $\mathsf{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{p}_{i,j}^\top]_2\}_{j\in[L]\setminus\{i\}}, \pi_i)$. It outputs 1 if $\mathsf{LVerify}(\mathsf{crs}_i, [\mathbf{M}_i]_1, \pi_i) = 1$ where $\mathbf{M}_i = \begin{pmatrix} \mathbf{T}_i \\ \mathbf{Q}_i \end{pmatrix}$ and $\forall i \neq j : [\mathbf{Ap}_{i,j}^\top]_T = [\mathbf{T}_i\mathbf{Br}_j^\top]_T$, and outputs 0 otherwise.

$\mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_i, y_i\}_{i\in[L]})$**:** It parses $\mathsf{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{p}_{i,j}^\top]_2\}_{j\in[L]\setminus\{i\}}, \pi_i)$ and computes $\mathbf{a}_{y_i}$ and $\mathbf{K}_{y_i}$ following Definition 2.6. It outputs mpk and $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$ as follows:

$$\mathsf{mpk} = \left([\mathbf{Ah}^\top]_T, \left[\mathbf{A}, \underbrace{\sum_{i\in[L]} \Big((\mathbf{AW}_{i,0} + \mathbf{T}_i)(\mathbf{a}_{y_i} \otimes \mathbf{I}_{k+1}) + \mathbf{AW}_i(\mathbf{K}_{y_i} \otimes \mathbf{I}_{k+1}) \Big)}_{\mathbf{P}} \right]_1 \right)$$
$$\mathsf{hsk}_i = \left[\underbrace{\mathbf{Br}_i^\top}_{\mathbf{h}_1^\top}, \underbrace{\mathbf{W}_{i,0}\mathbf{Br}_i^\top + \mathbf{h}^\top}_{\mathbf{h}_2^\top}, \underbrace{\sum_{j\in[L]\setminus\{i\}} \Big((\mathbf{W}_{j,0}\mathbf{Br}_i^\top + \mathbf{p}_{j,i}^\top)\mathbf{a}_{y_i} + \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)\mathbf{K}_{y_i} \Big)}_{\mathbf{H}_3} \right]_2$$

$\mathsf{Enc}(\mathsf{mpk}, x, M)$**:** It takes mpk, $x \in \mathcal{X}$, and $M \in G_T$ as inputs, and outputs ct_x by computing as follows. It parses mpk as above. It computes $\mathbf{C}_x$ following Definition 2.6 and

$$\mathbf{s} \leftarrow \mathbb{Z}_p^k,\ \mathsf{ct}_x = ([\underbrace{\mathbf{sA}}_{\mathbf{c}_1}, \underbrace{\mathbf{sP}(\mathbf{C}_x \otimes \mathbf{I}_{k+1})}_{\mathbf{c}_2}]_1,\ \underbrace{[\mathbf{sAh}^\top]_T M}_{C})$$

$\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x, x, y_i)$**:** It parses $\mathsf{sk}_i = \mathbf{V}_i$, $\mathsf{hsk}_i = ([\mathbf{h}_1^\top, \mathbf{h}_2^\top, \mathbf{H}_3]_2)$, $\mathsf{ct}_x = ([\mathbf{c}_1, \mathbf{c}_2]_1, C)$ and computes $\mathbf{C}_x$ and $\mathbf{d}_{x,y_i}$ following Definition 2.6. It outputs $M' = C \cdot [(\mathbf{z}_1 - \mathbf{z}_2)\mathbf{d}_{x,y_i}^\top - z_3]_T$ where

$$[\mathbf{z}_1]_T = e([\mathbf{c}_2]_1, [\mathbf{I}_{n_c} \otimes \mathbf{h}_1^\top]_2),\ [\mathbf{z}_2]_T = e([\mathbf{c}_1]_1, [\mathbf{H}_3\mathbf{C}_x]_2)$$
$$[z_3]_T = e([\mathbf{c}_1]_1, [\mathbf{h}_2^\top]_2) \cdot e([\mathbf{c}_1]_1, [\mathbf{V}_i\mathbf{h}_1^\top]_2)$$

Completeness. It is obvious that the completeness of the above scheme holds from the perfect completeness of the QA-NIZK scheme.

Correctness. In decryption, we have

$$\begin{aligned}
\mathbf{z}_1 =& \mathbf{sP}(\mathbf{C}_x \otimes \mathbf{Br}_j^\top) \\
=& \mathbf{s} \sum_{j\in[L]} \Big((\mathbf{AW}_{j,0} + \mathbf{T}_j)(\mathbf{a}_{y_j}\mathbf{C}_x \otimes \mathbf{Br}_j^\top) + \mathbf{AW}_j(\mathbf{K}_{y_j}\mathbf{C}_x \otimes \mathbf{Br}_j^\top) \Big) \\
=& \sum_{j\in[L]} \Big(\mathbf{sAW}_{j,0}\mathbf{Br}_j^\top \mathbf{a}_{y_j}\mathbf{C}_x + \mathbf{sAW}_j(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)\mathbf{K}_{y_j}\mathbf{C}_x \Big) \\
=& \sum_{j\in[L]} (\mathbf{sA}(\mathbf{W}_{j,0} + \mathbf{V}_j)\mathbf{Br}_j^\top || \mathbf{sAW}_j(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)) \begin{pmatrix} \mathbf{a}_{y_j} \\ \mathbf{K}_{y_j} \end{pmatrix} \mathbf{C}_x
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{z}_2 =& \mathbf{sA} \sum_{j\in[L]\setminus\{i\}} \Big((\mathbf{W}_{j,0}\mathbf{Br}_j^\top + \mathbf{p}_{j,i}^\top)\mathbf{a}_{y_j} + \mathbf{W}_j(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)\mathbf{K}_{y_j} \Big) \mathbf{C}_x \\
=& \sum_{j\in[L]\setminus\{i\}} (\mathbf{sA}(\mathbf{W}_{j,0} + \mathbf{V}_j)\mathbf{Br}_j^\top || \mathbf{sAW}_j(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)) \begin{pmatrix} \mathbf{a}_{y_j} \\ \mathbf{K}_{y_j} \end{pmatrix} \mathbf{C}_x
\end{aligned}$$

Therefore, we have

$$(\mathbf{z}_1 - \mathbf{z}_2)\mathbf{d}_{x,y_i}^\top = \mathbf{s}\mathbf{A}(\mathbf{W}_{i,0} + \mathbf{V}_i)\mathbf{B}\mathbf{r}_i^\top$$
$$(\mathbf{z}_1 - \mathbf{z}_2)\mathbf{d}_{x,y_i}^\top - z_3 = -\mathbf{s}\mathbf{A}\mathbf{h}^\top$$

which follows from the correctness of predicate encoding and the equality $z_3 = \mathbf{s}\mathbf{A}(\mathbf{V}_i\mathbf{B}\mathbf{r}_i^\top + \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top)$.

Compactness. It follows from the sizes described below and the compactness of the predicate encoding, i.e., $n_k = O(\log |\mathcal{Y}|)$.

Efficiency. Regarding the number of group elements, we have

$$|\mathsf{crs}| = O(\omega L^2), \quad |\mathsf{pk}_i| = O(L), \quad |\mathsf{sk}_i| = O(1)$$
$$|\mathsf{mpk}| = O(n_k), \quad |\mathsf{hsk}_i| = O(n_k), \quad |\mathsf{ct}| = O(n_c)$$

Especially,

$$|\mathsf{mpk}| = (k(2k+1) + k(k+1)n_k)|G_1| + k|G_T|$$
$$|\mathsf{hsk}_i| = (3k + 2 + (2k+1)n_k)|G_2|$$
$$|\mathsf{ct}| = (2k + 1 + (k+1)n_c)|G_1| + |G_T|$$

where ω, n_c, n_k are for the encoding of Definition 2.6.

Efficiency of ZZGQ scheme [ZZGQ23, Section 3.1]. For comparison, we provide space efficiency of the ZZGQ scheme. Regarding the number of group elements, we have

$$|\mathsf{crs}| = O(\omega L^2), \quad |\mathsf{pk}_i| = O(L), \quad |\mathsf{sk}_i| = O(1)$$
$$|\mathsf{mpk}| = O(n_k + \omega), \quad |\mathsf{hsk}_i| = O(n_k + \omega), \quad |\mathsf{ct}| = O(n_k + n_c)$$

Especially,

$$|\mathsf{mpk}| = (k(2k+1) + k(k+1)(n_k + \omega))|G_1| + k|G_T|$$
$$|\mathsf{hsk}_i| = (3k + 2 + (2k+1)(n_k + \omega))|G_2|$$
$$|\mathsf{ct}| = (2k + 1 + (k+1)(n_k + n_c))|G_1| + |G_T|$$

where ω, n_c, n_k are for the encoding of Definition 2.5.

3.2 Security

The security of our slotted registered ABE in Section 3.1 is given by the following theorem.

Theorem 3.1. *If Π satisfies perfect zero-knowledge and stronger unbounded simulation soundness for linear space, the MDDH assumption holds, and the predicate encoding is secure, then our sReg-ABE scheme satisfies the security defined in Definition 2.4.*

Proof. The proof of Theorem 3.1 is almost the same as that for the ZZGQ scheme, namely, the proof of [ZZGQ23, Theorem 1] except for the steps we use the security of predicate encodings for Reg-ABE, i.e., $\mathsf{H}_{5,v}$ in the hybrids below. Following the ZZGQ, we consider the following hybrids:

G_0**:** This is the original game. In this game, the adversary is given the following elements from the challenger:

$$\mathsf{crs} = \left(\begin{array}{l} [\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, \boxed{[\mathbf{R}_i]}_{\mathsf{H}_2,\mathsf{H}_4}, \mathbf{A}\mathbf{W}_{i,0}, \mathbf{A}\mathbf{W}_i]_1\}_{i\in[L]} \\ \{\underbrace{[\mathbf{B}\mathbf{r}_i^\top}_{\mathbf{f}_i^\top}, \underbrace{\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top]_2}_{\boxed{\mathbf{g}_i^\top}_{\mathsf{H}_{5,i}}}\}_{i\in[L]}, \{\underbrace{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top}_{\mathbf{n}_{i,j}^\top}, \underbrace{\mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{B}\mathbf{r}_j^\top)]_2}_{\mathbf{N}_{i,j}}\}_{\substack{i,j\in[L] \\ i\neq j}} \end{array} \right)$$

where each element is computed as described in Setup in Section 3.1,

$$\mathsf{pk}_i^\ell = \Big([\underbrace{\mathbf{A}\mathbf{V}_i^\ell}_{\mathbf{T}_i^\ell}, \underbrace{\mathbf{R}_i\mathbf{V}_i^\ell}_{\mathbf{Q}_i^\ell}]_1, \{[\underbrace{\mathbf{V}_i^\ell\mathbf{B}\mathbf{r}_j^\top}_{\mathbf{p}_{i,j}^{\ell\top}}]_2\}_{j\in[L]\setminus\{i\}}, \boxed{\pi_i^\ell}_{\mathsf{H}_1}\Big), \quad \mathsf{sk}_i^\ell = \mathbf{V}_i^\ell$$

from the ℓ-th query $\mathsf{OGen}(i)$ and $\mathsf{OCor}(i, \mathsf{pk}_i^\ell)$, respectively, where each element is computed as described in Gen in Section 3.1, and the challenge ciphertext ct_x:

$$\mathsf{ct}_x = \left([\underbrace{\mathbf{s}\mathbf{A}}_{\boxed{\mathbf{c}_1}_{\mathsf{H}_4}}, \underbrace{\mathbf{s}\mathbf{P}(\mathbf{C}_x\otimes\mathbf{I}_{k+1})}_{\boxed{\mathbf{c}_2}_{\mathsf{H}_3,\mathsf{H}_4}}]_1, \underbrace{[\mathbf{s}\mathbf{A}\mathbf{h}^\top]_T M_\beta}_{\boxed{C}_{\mathsf{H}_4,\mathsf{H}_6}}\right)$$

with respect to the attributes $x, \{y_i\}_{i\in[L]}$ and the challenge public keys $\mathsf{pk}_i^* = ([\mathbf{T}_i^*, \mathbf{Q}_i^*]_1, \{[\mathbf{p}_{i,j}^{*\top}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i^*)$, where $\beta \leftarrow \{0,1\}$ and the other elements are computed as described in Enc in Section 3.1. The boxed elements $\boxed{\mathsf{element}}_{\mathsf{H}_j}$ will be changed in hybrid H_j.

H_1: It is the same as G_0 except that the reply from $\mathsf{OGen}(i)$ is changed. Specifically, π_i^ℓ for all (i,ℓ) is generated as $\pi_i^\ell \leftarrow \mathsf{LSim}(\mathsf{crs}_i, \mathsf{td}_i, [\mathbf{M}_i^\ell]_1)$ instead of $\mathsf{LProve}(\mathsf{crs}_i, [\mathbf{M}_i^\ell]_1, \mathbf{V}_i^\ell)$ where $\mathbf{M}_i^\ell = \left(\begin{smallmatrix}\mathbf{T}_i^\ell\\ \mathbf{Q}_i^\ell\end{smallmatrix}\right)$.

H_2: It is the same as H_1 except that $[\mathbf{R}_i]_1$ in crs is defined as $\mathbf{R}_i = \widetilde{\mathbf{R}}_i \left(\begin{smallmatrix}\mathbf{s}\mathbf{A}\\ \mathbf{I}_{2k+1}\end{smallmatrix}\right)$ where $\widetilde{\mathbf{R}}_i \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+2)}, \mathbf{s} \leftarrow \mathbb{Z}_p^k$, instead of $\mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+1)}$. Note that $\mathbf{s}$ is also used for generating the challenge ciphertext ct_x.

H_3: It is the same as H_2 except that we change the way $[\mathbf{c}_2]_1$ in ct_x is generated. In H_3, $\mathbf{c}_2$ is computed as

$$\sum_{i\in[L]}\Big((\mathbf{s}\mathbf{A}\mathbf{W}_{i,0} + \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\mathbf{Q}_i^*)(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1}) + \mathbf{s}\mathbf{A}\mathbf{W}_i(\mathbf{K}_{y_i}\otimes\mathbf{I}_{k+1})\Big)(\mathbf{C}_x\otimes\mathbf{I}_{k+1})$$

where $\mathbf{e}_1 = (1, \mathbf{0})$.

H_4: It is the same as H_2 except that we replace the term $\mathbf{s}\mathbf{A}$ with $\mathbf{c} \leftarrow \mathbb{Z}_p^{2k+1}$. Specifically, we defined $\mathbf{R}_i = \widetilde{\mathbf{R}}_i \left(\begin{smallmatrix}\mathbf{c}\\ \mathbf{I}_{2k+1}\end{smallmatrix}\right)$ in crs and $\mathbf{c}_1 = \mathbf{c}$, $C = [\mathbf{c}\mathbf{h}^\top]_T M_\beta$,

$$\mathbf{c}_2 = \sum_{i\in[L]}\Big((\mathbf{c}\mathbf{W}_{i,0} + \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\mathbf{Q}_i^*)(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1}) + \mathbf{c}\mathbf{W}_i(\mathbf{K}_{y_i}\otimes\mathbf{I}_{k+1})\Big)(\mathbf{C}_x\otimes\mathbf{I}_{k+1})$$

in ct_x.

$\mathsf{H}_{5,v}(v\in[L])$: It is the same as H_4 except that $\mathbf{g}_i$ in crs is defined as $\mathbf{g}_i = \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top + \boxed{\alpha_i\mathbf{a}^{\perp\top}}$ for $i \in [v]$ instead of $\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top$ where $\alpha_i \leftarrow \mathbb{Z}_p, \mathbf{a}^\perp \leftarrow \mathbb{Z}_p^{2k+1}$ conditioned on $\mathbf{a}^\perp\mathbf{A} = \mathbf{0}$.

H_6: It is the same as $\mathsf{H}_{5,L}$ except that C in ct_x is generated as $C \leftarrow G_T$ instead of $[\mathbf{c}\mathbf{h}^\top]_T M_\beta$.

Observe that the adversary does not obtain the information of β in H_6. Theorem 3.1 follows from Lemmata 3.1 to 3.6. □

Lemma 3.1. *If Π satisfies perfect zero-knowledge, then $\mathsf{G}_0 = \mathsf{H}_1$.*

Proof. Lemma 3.1 is obvious from perfect zero-knowledge of Π since $\mathbf{M}_i^\ell = \mathbf{A}_i\mathbf{V}_i^\ell$ for all (i,ℓ) in pk_i^ℓ from the ℓ-query to $\mathsf{OGen}(i)$. □

Lemma 3.2. $\mathsf{H}_1 \approx_s \mathsf{H}_2$.

Proof. Since $\left(\begin{smallmatrix}\mathbf{s}\mathbf{A}\\ \mathbf{I}_{2k+1}\end{smallmatrix}\right)$ is full-rank, it suffices to prove that $\mathbf{R} \approx_s \widetilde{\mathbf{R}}\mathbf{M}$ where $\mathbf{R} \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+1)}$, $\widetilde{\mathbf{R}} \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+2)}$, and $\mathbf{M}$ is a full-rank matrix in $\mathbb{Z}_p^{(2k+2)\times(2k+1)}$ independent of $\mathbf{R}, \widetilde{\mathbf{R}}$. As long as $\mathbf{R}$ and $\widetilde{\mathbf{R}}$ are full-rank, which occurs with overwhelming probability, $\mathbf{R}$ and $\widetilde{\mathbf{R}}\mathbf{M}$ are uniformly distributed in a set of full-rank matrices over $\mathbb{Z}_p^{(2k+2)\times(2k+1)}$. Hence, these distributions are statistically close. □

Lemma 3.3. *If Π satisfies unbounded simulation soundness,* $\mathsf{H}_2 \approx_c \mathsf{H}_3$.

Proof. If for all challenge public keys $\{\mathsf{pk}_i^*\}_{i\in[L]}$ that $\mathcal{A}$ outputs, there exists $\mathbf{V}_i^* \in \mathbb{Z}_p^{(2k+1)\times(k+1)}$ such that $\begin{pmatrix}\mathbf{T}_i^*\\ \mathbf{Q}_i^*\end{pmatrix} = \mathbf{A}_i\mathbf{V}_i^*$ where $\mathbf{A}_i = \left(\widetilde{\mathbf{R}}_i\begin{pmatrix}\mathbf{A}\\ \mathbf{sA}\\ \mathbf{I}_{2k+1}\end{pmatrix}\right)$, then we have

$$\mathbf{sT}_i^* = \mathbf{sAV}_i^* = \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\widetilde{\mathbf{R}}_i\begin{pmatrix}\mathbf{sA}\\ \mathbf{I}_{2k+1}\end{pmatrix}\mathbf{V}_i^* = \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\mathbf{Q}_i^*$$

Thus, as long as the above condition holds, the views of $\mathcal{A}$ in H_2 and H_3 are identical. Furthermore, the condition always holds if pk^* is obtained from $\mathsf{OGen}(i)$. Hence, H_2 and H_3 are identical unless the following Bad event happens: let Bad be the event such that there exist $\mathsf{pk}_i^* = ([\mathbf{T}_i^*, \mathbf{Q}_i^*]_1, \{[\mathbf{p}_{i,j}^{*\top}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i^*)$ such that for all $\mathbf{V}_i^* \in \mathbb{Z}_p^{(2k+1)\times(k+1)}$, $\mathbf{M}_i^* = \begin{pmatrix}\mathbf{T}_i^*\\ \mathbf{Q}_i^*\end{pmatrix} \neq \mathbf{A}_i\mathbf{V}_i^*$ and $\mathsf{LVerify}(\mathsf{crs}_i, [\mathbf{M}_i^*]_1, \pi_i^*) = 1$. Hence, to prove the lemma it suffices to show $\Pr[\mathsf{Bad}]$ is negligible.

We show that if $\mathcal{A}$ makes Bad happen with non-negligible probability, we can construct $\mathcal{B}$ that breaks strong unbounded simulation soundness of Π as follows.

1. $\mathcal{B}$ randomly chooses $i^* \leftarrow [L]$ as a guess of the slot for which Bad occurs.
2. $\mathcal{B}$ is given an input for the unbounded simulation soundness game $(1^\lambda, \mathsf{crs}_{i^*}, \mathbf{A}_{i^*})$, samples $\mathbf{s} \leftarrow \mathbb{Z}_p^k$, and parses $\mathbf{A}_{i^*} = \begin{pmatrix}\mathbf{A}\\ \mathbf{R}_{i^*}\end{pmatrix} = \left(\widetilde{\mathbf{R}}_{i^*}\begin{pmatrix}\mathbf{A}\\ \mathbf{sA}\\ \mathbf{I}_{2k+1}\end{pmatrix}\right)$, i.e, randomly samples $\widetilde{\mathbf{R}}_{i^*} \in \mathbb{Z}_p^{(2k+2)\times(2k+2)}$ satisfying the equality.
3. $\mathcal{B}$ samples $\widetilde{\mathbf{R}}_i \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+2)}$ and sets $\mathbf{R}_i = \widetilde{\mathbf{R}}_i\begin{pmatrix}\mathbf{sA}\\ \mathbf{I}_{2k+1}\end{pmatrix}$ for $i \in [L]\setminus\{i^*\}$, samples all variable other than $\mathbf{A}, \mathsf{crs}_{i^*}, \{\mathbf{R}_i\}_{i\in[L]}$ in the same manner as $\mathsf{Setup}(1^\lambda, 1^L, \kappa)$, and gives crs to $\mathcal{A}$ as follows:

$$\mathsf{crs} = \begin{pmatrix}[\mathbf{A}]_1, [\mathbf{Ah}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{AW}_{i,0}, \mathbf{AW}_i]_1\}_{i\in[L]}\\ \{[\mathbf{Br}_i^\top, \mathbf{W}_{i,0}\mathbf{Br}_i^\top + \mathbf{h}^\top]_2\}_{i\in[L]}, \{[\mathbf{W}_{i,0}\mathbf{Br}_{i^*}^\top, \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{Br}_j^\top)]_2\}_{\substack{i,j\in[L]\\ i\neq j}}\end{pmatrix}$$

4. When $\mathcal{A}$ queries $\mathsf{OGen}(i)$, $\mathcal{B}$ computes

$$\mathbf{V}_i \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)},\ \mathbf{M}_i = \begin{pmatrix}\mathbf{AV}_i\\ \mathbf{R}_i\mathbf{V}_i\end{pmatrix},\ \pi_i \leftarrow \mathsf{LSim}(\mathsf{crs}_i, \mathsf{td}_i, [\mathbf{M}_i]_1)$$
$$\mathsf{pk}_i = ([\mathbf{AV}_i, \mathbf{R}_i\mathbf{V}_i]_1, \{[\mathbf{V}_i\mathbf{Br}_j^\top]_2\}_{j\in[L]\setminus\{i\}}, \pi_i),\ \ \mathsf{sk}_i = \mathbf{V}_i$$

   where $\mathcal{B}$ uses the simulation oracle in the unbounded simulation soundness game of Π when generating π_{i^*}. Then, $\mathcal{B}$ stores a pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ in dictionary $\mathcal{D}_i$ and gives pk_i to $\mathcal{A}$.
5. When $\mathcal{A}$ queries $\mathsf{OCor}(i, \mathsf{pk})$, $\mathcal{B}$ replies sk if and only if $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{D}_i$.
6. When $\mathcal{A}$ outputs a set of challenge public keys $\{\mathsf{pk}_i^*\}_i$ where $\mathsf{pk}_{i^*}^* = ([\mathbf{T}_{i^*}^*, \mathbf{Q}_{i^*}^*]_1, \{[\mathbf{p}_{i^*,j}^{*\top}]_2\}_{j\in[L]\setminus\{i^*\}}, \pi_{i^*}^*)$, $\mathcal{B}$ outputs $\left(\left[\begin{pmatrix}\mathbf{T}_{i^*}^*\\ \mathbf{Q}_{i^*}^*\end{pmatrix}\right]_1, \pi_{i^*}^*\right)$ and halts.

If Bad occurs, then $\begin{pmatrix}\mathbf{T}_{i^*}^*\\ \mathbf{Q}_{i^*}^*\end{pmatrix}$ is not in the space spanned by the columns of $\mathbf{A}_{i^*}$ with probability at least $1/L$ since i^* is uniformly chosen from $[L]$. Hence, $\mathcal{B}$ breaks strong unbounded simulation soundness of Π with $\Pr[\mathsf{Bad}]/L$, which is non-negligible if $\Pr[\mathsf{Bad}]$ is non-negligible. □

Lemma 3.4. *If the MDDH assumption holds in* $\mathbb{G}$, $\mathsf{H}_3 \approx_c \mathsf{H}_4$.

Proof. Thanks to the MDDH assumption, we have $([\mathbf{A}]_1, [\mathbf{sA}]_1) \approx_c ([\mathbf{A}]_1, [\mathbf{c}]_1)$ where $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(2k+1)}$, $\mathbf{s} \leftarrow \mathbb{Z}_p^k$, $\mathbf{c} \leftarrow \mathbb{Z}_p^{sk+1}$. Observe that all elements that $\mathcal{A}$ obtains in H_3 and H_4 are affine in $\mathbf{A}, \mathbf{sA}, \mathbf{c}$. Hence, we can easily construct a reduction $\mathcal{B}$ from the MDDH problem to distinguishing H_3 and H_4. □

Lemma 3.5. $\mathsf{H}_{5,L} \approx_s \mathsf{H}_6$.

Proof. Recall that the terms that involve $\mathbf{h}$ in $\mathsf{H}_{5,L}$ are $[\mathbf{A}\mathbf{h}^\top]_T$ and $\{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top + \alpha_i \mathbf{a}^{\perp\top}]_2\}_{i\in[L]}$ in crs and $[\mathbf{c}\mathbf{h}^\top]_T M_\beta$ in ct_x. Let $\mathbf{h}^\top = \mathbf{h}'^\top + \alpha' \mathbf{a}^{\perp\top}$ where $\mathbf{h}' \leftarrow \mathbb{Z}_p^{2k+1}, \alpha' \leftarrow \mathbb{Z}_p$. This does not change the distribution of $\mathbf{h}$. Then, the above terms can be written as $[\mathbf{A}\mathbf{h}'^\top]_T$, $\{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}'^\top + (\alpha_i + \alpha')\mathbf{a}^{\perp\top}]_2\}$, $[\mathbf{c}(\mathbf{h}^\top + \alpha' \mathbf{a}^{\perp\top})]_T M_\beta$. Since $\alpha_i + \alpha'$ is randomly distributed in $\mathbb{Z}_p$, $\alpha' \mathbf{c}\mathbf{a}^{\perp\top}$ is randomly distributed in $\mathbb{Z}_p$ unless $\mathbf{c}\mathbf{a}^{\perp\top} = 0$, which occurs with negligible probability. The latter distribution corresponds to H_6. Thus, both hybrids are identical with overwhelming probability. □

Lemma 3.6. *Let $\mathsf{H}_{5,0} = \mathsf{H}_4$. If the predicate encoding is secure as per Definition 2.6 and the MDDH assumption holds in $\mathbb{G}$, then $\mathsf{H}_{5,v-1} \approx_c \mathsf{H}_{5,v}$ for $v \in [L]$.*

Proof. We define intermediate hybrids $\widehat{\mathsf{H}}^1_{5,v-1}, \widehat{\mathsf{H}}^2_{5,v-1}, \widehat{\mathsf{H}}^3_{5,v-1}$ between $\mathsf{H}_{5,v-1}$ and $\mathsf{H}_{5,v}$ and show that $\mathsf{H}_{5,v-1} \approx_c \widehat{\mathsf{H}}^1_{5,v-1} \approx_c \widehat{\mathsf{H}}^2_{5,v-1} \approx_c \widehat{\mathsf{H}}^3_{5,v-1} \approx_c \mathsf{H}_{5,v}$. These hybrids are defined as follows.

$\widehat{\mathsf{H}}^1_{5,v-1}$: It is the same as $\mathsf{H}_{5,v-1}$ except that $\mathbf{B}\mathbf{r}_v^\top$ is replaced with $\mathbf{d}^\top$ where $\mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$. Recall that crs in $\mathsf{H}_{5,v-1}$ is described as

$$\left([\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{A}\mathbf{W}_{i,0}, \mathbf{A}\mathbf{W}_i]_1\}_{i\in[L]} \right.$$
$$\left. \{[\underbrace{\mathbf{B}\mathbf{r}_i^\top}_{\mathbf{f}_i^\top}, \underbrace{\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top + \alpha_i \mathbf{a}^{\perp\top}}_{\mathbf{g}_i^\top}]_2\}_{i\in[L]}, \{[\underbrace{\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top}_{\mathbf{n}_{i,j}^\top}, \underbrace{\mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{B}\mathbf{r}_j^\top)}_{\mathbf{N}_{i,j}}]_2\}_{\substack{i,j\in[L] \\ i\neq j}} \right)$$

where $\alpha_i \leftarrow \mathbb{Z}_p$ if $i \leq v-1$ and otherwise $\alpha_i = 0$, and pk_i^ℓ obtained from the ℓ-th query to $\mathsf{OGen}(i)$ in $\mathsf{H}_{5,v-1}$ is described as

$$\left([\underbrace{\mathbf{A}\mathbf{V}_i^\ell}_{\mathbf{T}_i^\ell}, \underbrace{\mathbf{R}_i\mathbf{V}_i^\ell}_{\mathbf{Q}_i^\ell}]_1, \{[\underbrace{\mathbf{V}_i^\ell \mathbf{B}\mathbf{r}_j^\top}_{\mathbf{p}_{i,j}^{\ell\top}}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i^\ell\right)$$

In $\widehat{\mathsf{H}}^1_{5,v-1}$, the following terms are changed as

$$\mathbf{f}_v^\top = \mathbf{d}^\top, \;\; \mathbf{g}_v^\top = \mathbf{W}_{v,0}\mathbf{d}^\top + \mathbf{h}^\top, \;\; \mathbf{n}_{i,v}^\top = \mathbf{W}_{i,0}\mathbf{d}^\top, \;\; \mathbf{N}_{i,v} = \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{d}^\top)$$
$$\mathbf{p}_{i,v}^{\ell\top} = \mathbf{V}_i^\ell \mathbf{d}^\top$$

$\widehat{\mathsf{H}}^2_{5,v-1}$: It is the same as $\widehat{\mathsf{H}}^1_{5,v-1}$ except that $\mathbf{c}_2$ is computed as

$$\mathbf{c}_2 = \sum_{i\in[L]} \Big((\mathbf{c}\mathbf{W}_{i,0} + \mathbf{e}_1 \widetilde{\mathbf{R}}_i^{-1} \mathbf{Q}_i^*)(\mathbf{a}_{y_i} \otimes \mathbf{I}_{k+1}) + \mathbf{c}\mathbf{W}_i(\mathbf{K}_{y_i} \otimes \mathbf{I}_{k+1})\Big)(\mathbf{C}_x \otimes \mathbf{I}_{k+1})$$
$$+ \boxed{\mathbf{b}^\perp(\mathbf{a}_{y_i} \otimes \mathbf{I}_{k+1})(\mathbf{C}_x \otimes \mathbf{I}_{k+1})}$$

where $\mathbf{b}^\perp \leftarrow \mathbb{Z}_p^{k+1}$ conditioned on $\mathbf{b}^\perp \mathbf{B} = \mathbf{0}$.

$\widehat{\mathsf{H}}^3_{5,v-1}$: It is the same as $\widehat{\mathsf{H}}^1_{5,v-2}$ except that $\mathbf{g}_v^\top = \mathbf{W}_{v,0}\mathbf{d}^\top + \mathbf{h}^\top + \boxed{\alpha_v \mathbf{a}^{\perp\top}}$ where $\alpha_v \leftarrow \mathbb{Z}_p$.

Thanks to Lemmata 3.7 to 3.10, Lemma 3.6 holds. □

Lemma 3.7. *If the MDDH assumption holds in $\mathbb{G}$, then $\mathsf{H}_{5,v-1} \approx_c \widehat{\mathsf{H}}^1_{5,v-1}$ for $v \in [L]$.*

Proof. This lemma is straightforward from the MDDH assumption, which asserts that $[\mathbf{B}, \mathbf{z}_0]_2 \approx_c [\mathbf{B}, \mathbf{z}_1]_2$ where $\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k}, \mathbf{r}_v \leftarrow \mathbb{Z}_p^k, \mathbf{z}_0 = \mathbf{B}\mathbf{r}_v^\top, \mathbf{z}_1 = \mathbf{d} \leftarrow \mathbb{Z}_p$. Since all the terms that the adversary is given are affine in $\mathbf{B}$ or $\mathbf{z}_0$ (resp. $\mathbf{z}_1$) in $\mathsf{H}_{5,v-1}$ (resp. $\widehat{\mathsf{H}}^1_{5,v-1}$), they are simulatable given the MDDH instance. □

Lemma 3.8. *If the MDDH assumption holds in $\mathbb{G}$ and the predicate encoding is secure, then $\widehat{\mathsf{H}}^1_{5,v-1} \approx_c \widehat{\mathsf{H}}^2_{5,v-1}$ for $v \in [L]$.*

Proof. We consider two cases of the adversary's behavior, namely, the honest case and the dishonest case. The honest cases refer to one in which the adversary outputs the challenge public key pk^*_v for the v-th slot such that $(\mathsf{pk}^*_v, *) \in \mathcal{D}_v$ and $(v, \mathsf{pk}^*_v) \notin \mathcal{C}$. On the other hand, the dishonest case refers to one that is not the honest case, that is, $(\mathsf{pk}^*_v, *) \notin \mathcal{D}_v$ or $(v, \mathsf{pk}^*_v) \in \mathcal{C}$. Thus, it suffices to prove $\widehat{\mathsf{H}}^1_{5,v-1} \approx_c \widehat{\mathsf{H}}^2_{5,v-1}$ under both cases.

Honest Case. We prove that $\widehat{\mathsf{H}}^1_{5,v-1} \approx_c \widehat{\mathsf{H}}^2_{5,v-1}$ in the honest case if the MDDH assumption holds in $\mathbb{G}$. First, we show the following indistinguishability holds under the MDDH assumption, which we will use later in the proof:

$$\begin{aligned} &\{[\mathbf{R}, \mathbf{RV}]_1, \mathbf{A}, \mathbf{AV}, \mathbf{b}^\perp, \mathbf{c}, \mathbf{cV}\} \\ \approx_c &\{[\mathbf{R}, \mathbf{RV}]_1, \mathbf{A}, \mathbf{AV}, \mathbf{b}^\perp, \mathbf{c}, \mathbf{cV} + \mathbf{b}^\perp\} \end{aligned} \tag{8}$$

where $\mathbf{R} \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+1)}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}, \mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(2k+1)}, \mathbf{b}^\perp \leftarrow \mathbb{Z}_p^{k+1}, \mathbf{c} \leftarrow \mathbb{Z}_p^{2k+1}$. The proof of the indistinguishability is found in the proof of [AT24, Lemma D.8].

We construct a distinguisher $\mathcal{B}$ between the LHS and RHS in Eq. (8) that internally uses a distinguisher $\mathcal{A}$ between $\widehat{\mathsf{H}}^1_{5,v-1}$ and $\widehat{\mathsf{H}}^2_{5,v-1}$ in the honest case as follows. Let Q be the maximum number of $\mathcal{A}$'s queries of the form $\mathsf{OGen}(i)$ for all $i \in [L]$.

1. $\mathcal{B}$ is given an instance $([\mathbf{R}_v, \mathbf{R}_v\mathbf{V}]_1, \mathbf{A}, \mathbf{AV}, \mathbf{b}^\perp, \mathbf{c}, \mathbf{z}_{\beta'})$ of Eq. (8) where $\mathbf{z}_{\beta'} = \mathbf{cV} + \beta'\mathbf{b}^\perp$ for $\beta' \in \{0,1\}$. $\mathcal{B}$ computes crs as follows and gives it to $\mathcal{A}$:

$$\mathsf{crs} = \begin{pmatrix} [\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{AW}_{i,0}, \mathbf{AW}_i]_1\}_{i\in[L]} \\ \{[\mathbf{f}_i^\top, \mathbf{g}_i^\top]_2\}_{i\in[L]}, \{[\mathbf{n}_{i,j}^\top, \mathbf{N}_{i,j}]_2\}_{\substack{i,j\in[L] \\ i\neq j}} \end{pmatrix}$$

where

$$\begin{aligned} &\mathbf{h} \leftarrow \mathbb{Z}_p^{2k+1},\ (\mathsf{crs}_i, \mathsf{td}_i) \leftarrow \mathsf{LGen}(1^\lambda),\ \mathbf{W}_{i,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)},\ \mathbf{W}_i \leftarrow \mathbb{Z}_p^{(2k+1)\times\omega(k+1)} \\ &\widetilde{\mathbf{R}}_{i\neq v} \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+2)},\ \mathbf{R}_{i\neq v} = \widetilde{\mathbf{R}}_i \begin{pmatrix} \mathbf{c} \\ \mathbf{I}_{2k+1} \end{pmatrix},\ \mathbf{r}_{i\neq v} \leftarrow \mathbb{Z}_p^k,\ \mathbf{d} \leftarrow \mathbb{Z}_p^{k+1} \\ &\alpha_{i<v} \leftarrow \mathbb{Z}_p,\ \alpha_{i>v} = 0,\ \mathbf{a}^\perp \leftarrow \mathbb{Z}_p^{2k+1} \text{ conditioned on } \mathbf{a}^\perp\mathbf{A}^\top = \mathbf{0} \\ &\mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \text{ conditioned on } \mathbf{b}^\perp\mathbf{B} = \mathbf{0} \\ &\mathbf{f}_i^\top = \begin{cases} \mathbf{B}\mathbf{r}_i^\top & (i \neq v) \\ \mathbf{d}^\top & (i = v) \end{cases}, \quad \mathbf{g}_i^\top = \begin{cases} \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top + \alpha_i\mathbf{a}^{\perp\top} & (i \neq v) \\ \mathbf{W}_{i,0}\mathbf{d}^\top + \mathbf{h}^\top & (i = v) \end{cases} \\ &\mathbf{n}_{i,j}^\top = \begin{cases} \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top & (j \neq v) \\ \mathbf{W}_{i,0}\mathbf{d}^\top & (j = v) \end{cases}, \quad \mathbf{N}_{i,j} = \begin{cases} \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{B}\mathbf{r}_j^\top) & (j \neq v) \\ \mathbf{W}_i(\mathbf{I}_\omega \otimes \mathbf{d}^\top) & (j = v) \end{cases} \end{aligned}$$

2. $\mathcal{B}$ randomly samples $q \leftarrow [Q]$ as a guess of the challenge public key for slot v. When $\mathcal{A}$ makes the ℓ-th query $\mathsf{OGen}(i)$, $\mathcal{B}$ gives pk_i^ℓ to $\mathcal{A}$ as follows:

$$\mathsf{pk}_i^\ell = ([\mathbf{T}_i^\ell, \mathbf{Q}_i^\ell]_1, \{[\mathbf{p}_{i,j}^{\ell\top}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i^\ell)$$

where $\mathbf{V}_i^\ell \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}$ and

$$\begin{aligned} &\mathbf{T}_i^\ell = \begin{cases} \mathbf{AV}_i^\ell & ((i,\ell) \neq (v,q)) \\ \mathbf{AV} & ((i,\ell) = (v,q)) \end{cases}, \quad \mathbf{Q}_i^\ell = \begin{cases} \mathbf{R}_i\mathbf{V}_i^\ell & ((i,\ell) \neq (v,q)) \\ \mathbf{R}_i\mathbf{V} & ((i,\ell) = (v,q)) \end{cases} \\ &\mathbf{p}_{i,j}^{\ell\top} = \begin{cases} \mathbf{V}_i^\ell\mathbf{B}\mathbf{r}_j^\top & (j \neq v, (i,\ell) \neq (v,q)) \\ \mathbf{V}\mathbf{B}\mathbf{r}_j^\top & ((i,\ell) = (v,q)) \\ \mathbf{V}_i^\ell\mathbf{d}^\top & (j = v) \end{cases}, \quad \pi_i^\ell \leftarrow \mathsf{LSim}\left(\mathsf{crs}_i, \mathsf{td}_i, \left[\begin{pmatrix} \mathbf{T}_i^\ell \\ \mathbf{Q}_i^\ell \end{pmatrix}\right]_1\right) \end{aligned}$$

Then, $\mathcal{B}$ sets $\mathcal{D}_i = (\mathsf{pk}_i^\ell, \mathbf{V}_i^\ell) \cup \mathcal{D}_i$ if $(i,\ell) \neq (v,q)$.

3. When $\mathcal{A}$ makes a query $\mathsf{OCor}(i, \mathsf{pk})$, $\mathcal{B}$ checks $i = v$ and $\mathsf{pk} = \mathsf{pk}_v^q$, and if so, $\mathcal{B}$ returns a random bit and halts. Otherwise, $\mathcal{B}$ gives sk to $\mathcal{A}$ if $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{D}_i$.
4. When $\mathcal{A}$ outputs the challenge $(\{\mathsf{pk}_i^*, y_i\}_{i\in[L]}, x, M_0, M_1)$, $\mathcal{B}$ checks $\mathsf{pk}_v^* = \mathsf{pk}_i^q$. If it is not the case, $\mathcal{B}$ returns a random bit and halts. Otherwise, it parses $\mathsf{pk}_i^* = ([\mathbf{T}_i^*, \mathbf{Q}_i^*]_1, \{[\mathbf{p}_{i,j}^{*\top}]_2\}_{j\in[L]\setminus\{i\}}, \pi_i^*)$ and gives $\mathsf{ct}_x = ([\mathbf{c}_1, \mathbf{c}_2]_1, C)$ to $\mathcal{A}$, where $\beta \leftarrow \{0,1\}$, $\mathbf{c}_1 = \mathbf{c}$, $C = [\mathbf{c}\mathbf{h}^\top]_T M_\beta$, and

$$\begin{aligned}\mathbf{c}_2 = &\sum_{i\in[L]\setminus\{v\}} (\mathbf{c}\mathbf{W}_{i,0} + \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\mathbf{Q}_i^*)(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1})(\mathbf{C}_x\otimes\mathbf{I}_{k+1})\\ &+ (\mathbf{c}\mathbf{W}_{v,0} + \mathbf{z}_{\beta'})(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1})(\mathbf{C}_x\otimes\mathbf{I}_{k+1})\\ &+ \sum_{i\in[L]} \mathbf{c}\mathbf{W}_i(\mathbf{K}_{y_i}\otimes\mathbf{I}_{k+1})(\mathbf{C}_x\otimes\mathbf{I}_{k+1})\end{aligned}$$

5. $\mathcal{B}$ outputs $\beta \stackrel{?}{=} \beta''$ when $\mathcal{A}$ output β''.

Observe that $\mathcal{A}$'s view corresponds to $\widehat{\mathsf{H}}^1_{5,v-1}$ if $\beta' = 0$ and $\widehat{\mathsf{H}}^2_{5,v-1}$ otherwise. This follows from the fact that $\mathsf{pk}_v^* = \mathsf{pk}_v^q$ implies $\mathbf{Q}_v^* = \mathbf{R}_v\mathbf{V}$, $\mathbf{R}_v = \widetilde{\mathbf{R}}_v \left(\begin{smallmatrix}\mathbf{c}\\ \mathbf{I}_{2k+1}\end{smallmatrix}\right)$, and thus $\mathbf{e}_1\widetilde{\mathbf{R}}_v^{-1}\mathbf{Q}_v^* = \mathbf{c}\mathbf{V}$. Finally, since we are considering the honest case, i.e., $\mathcal{A}$ outputs pk_v^* such that $(\mathsf{pk}_v^*, *) \in \mathcal{D}_v$ and $(v, \mathsf{pk}_v^*) \notin \mathcal{C}$, $\mathcal{B}$ does not halt in step 3 or 4 with the probability not less than $1/Q$. Thus, if $\mathcal{A}$ distinguishes $\widehat{\mathsf{H}}^1_{5,v-1}$ and $\widehat{\mathsf{H}}^2_{5,v-1}$ with non-negligible advantage, then $\mathcal{B}$ distinguishes the two cases of Eq. (8) with non-negligible advantage, which breaks the MDDH assumption.

Dishonest Case. We prove that $\widehat{\mathsf{H}}^1_{5,v-1} \approx_s \widehat{\mathsf{H}}^2_{5,v-1}$ in the dishonest case relying on the security of the predicate encoding. To see this, we consider the following programming:

$$\mathbf{W}_{v,0} = \mathbf{W}'_{v,0} + w_{v,0}\mathbf{a}^{\perp\top}\mathbf{b}^\perp, \quad \mathbf{W}_v = \mathbf{W}'_v + \mathbf{w}_v \otimes \mathbf{a}^{\perp\top}\mathbf{b}^\perp$$

where $\mathbf{W}'_{v,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}$, $\mathbf{W}'_v \leftarrow \mathbb{Z}_p^{(2k+1)\times\omega(k+1)}$, $w_{v,0} \leftarrow \mathbb{Z}_p$, $\mathbf{w}_v \leftarrow \mathbb{Z}_p^\omega$, and $\mathbf{b}^\perp \leftarrow \mathbb{Z}_p^{k+1}$ conditioned on $\mathbf{b}^\perp\mathbf{B} = \mathbf{0}$.. This programming affects $\mathbf{g}_v$ in crs as

$$\mathbf{g}_v^\top = \mathbf{W}'_{v,0}\mathbf{d}^\top + \mathbf{h}^\top + w_{v,0}\mathbf{a}^{\perp\top}\mathbf{b}^\perp\mathbf{d}^\top$$

and $\mathbf{c}_2$ in ct_x as

$$\begin{aligned}\mathbf{c}_2 = &\underbrace{\sum_{i\in[L]} \Big((\mathbf{c}\mathbf{W}'_{i,0} + \mathbf{e}_1\widetilde{\mathbf{R}}_i^{-1}\mathbf{Q}_i^*)(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1}) + \mathbf{c}\mathbf{W}'_i(\mathbf{K}_{y_i}\otimes\mathbf{I}_{k+1})\Big)}_{X}(\mathbf{C}_x\otimes\mathbf{I}_{k+1})\\ &+ (\mathbf{c}\mathbf{a}^{\perp\top}w_{v,0}\mathbf{a}_{y_v}\otimes\mathbf{b}^\perp + \mathbf{c}\mathbf{a}^{\perp\top}\mathbf{w}_v\mathbf{K}_{y_v}\otimes\mathbf{b}^\perp)(\mathbf{C}_x\otimes\mathbf{I}_{k+1})\\ =&X + \mathbf{c}\mathbf{a}^{\perp\top}(w_{v,0}||\mathbf{w}_v)\left(\begin{smallmatrix}\mathbf{a}_{y_v}\\ \mathbf{K}_{y_v}\end{smallmatrix}\right)\mathbf{C}_x\otimes\mathbf{b}^\perp\end{aligned}$$

Recall that the security of the predicate encoding asserts

$$\left(w_{v,0}, (w_{v,0}||\mathbf{w}_v)\left(\begin{smallmatrix}\mathbf{a}_{y_v}\\ \mathbf{K}_{y_v}\end{smallmatrix}\right)\mathbf{C}_x\right) \equiv \left(w_{v,0}, (0||\mathbf{w}_v)\left(\begin{smallmatrix}\mathbf{a}_{y_v}\\ \mathbf{K}_{y_v}\end{smallmatrix}\right)\mathbf{C}_x\right)$$

Therefore, $\mathbf{c}_2$ and $\mathbf{c}_2 + \mathbf{a}_{y_v}\mathbf{C}_x \otimes \underbrace{(-w_{v,0}\mathbf{c}\mathbf{a}^{\perp\top}\mathbf{b}^\perp)}_{\mathbf{b}'^\perp}$ are statistically close. Since we have $\mathbf{a}_{y_v}\mathbf{C}_x\otimes\mathbf{b}'^\perp = \mathbf{b}'^\perp(\mathbf{a}_{y_i}\otimes\mathbf{I}_{k+1})(\mathbf{C}_x\otimes\mathbf{I}_{k+1})$, the indistinguishability of the dishonest case holds. This concludes the proof. □

Lemma 3.9. $\widehat{\mathsf{H}}^2_{5,v-1} \approx_s \widehat{\mathsf{H}}^3_{5,v-1}$ *for* $v \in [L]$.

Proof. Let $\mathbf{W}_{v,0} = \mathbf{W}'_{v,0} + \alpha' \mathbf{a}^{\perp\top} \mathbf{b}^{\perp}$ where $\mathbf{W}'_{v,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}, \alpha' \leftarrow \mathbb{Z}_p$. Then, $\mathbf{W}_{v,0}$ and $\mathbf{W}'_{v,0}$ are equivalently distributed. The terms involving $\mathbf{W}_{v,0}$ are changed as follows:

$$\mathbf{A}\mathbf{W}_{v,0} = \mathbf{A}\mathbf{W}'_{v,0}, \quad \mathbf{g}_i^\top = \begin{cases} \mathbf{W}'_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top + \alpha_i \mathbf{a}^{\perp\top} & (i \neq v) \\ \mathbf{W}'_{i,0}\mathbf{d}^\top + \mathbf{h}^\top + \underbrace{(\alpha' \mathbf{b}^\perp \mathbf{d}^\top)}_{\alpha_i} \mathbf{a}^{\perp\top} & (i = v) \end{cases}$$

in crs, and $\mathbf{c}_2$ can be written as

$$\sum_{i\in[L]} \Big((\mathbf{c}\mathbf{W}_{i,0} + \mathbf{c}\mathbf{W}'_{v,0} + \mathbf{e}_1 \widetilde{\mathbf{R}}_i^{-1} \mathbf{Q}_i^*)(\mathbf{a}_{y_i} \otimes \mathbf{I}_{k+1}) + \mathbf{c}\mathbf{W}_i (\mathbf{K}_{y_i} \otimes \mathbf{I}_{k+1}) \Big) (\mathbf{C}_x \otimes \mathbf{I}_{k+1})$$
$$+ \underbrace{(1 + \alpha' \mathbf{c}\mathbf{a}^{\perp\top})\mathbf{b}^\perp}_{\mathbf{b}'^\perp} (\mathbf{a}_{y_i} \otimes \mathbf{I}_{k+1})(\mathbf{C}_x \otimes \mathbf{I}_{k+1})$$

in ct_x. It is not hard to see that α_v and $\mathbf{b}'^\perp$ are random elements in $\mathbb{Z}_p$ and $\mathbb{Z}_p^{k+1}$ conditioned on $\mathbf{b}'^\perp \mathbf{B} = \mathbf{0}$ unless $\mathbf{b}^\perp \mathbf{d}^\top = 0$ and $1 + \alpha' \mathbf{c}\mathbf{a}^{\perp\top} = 0$, respectively, which occur with negligible probability. This corresponds to the distribution in $\widehat{\mathsf{H}}^3_{5,v-1}$, and thus $\widehat{\mathsf{H}}^2_{5,v-1}$ and $\widehat{\mathsf{H}}^3_{5,v-1}$ are equivalent with overwhelming probability. □

Lemma 3.10. $\widehat{\mathsf{H}}^3_{5,v-1} \approx_c \mathsf{H}_{5,v}$ *for* $v \in [L]$.

This lemma can be proven similarly to $\mathsf{H}_{5,v-1} \approx_c \widehat{\mathsf{H}}^2_{5,v-1}$.

4 Concrete Predicate Encodings for Reg-ABE

In this section, we present concrete predicate encodings for Reg-ABE with respect to various predicates that can be captured by predicate encodings [Wee14, CGW15]. Our encodings can be plugged into the sReg-ABE scheme in Section 3.1, which yields more efficient sReg-ABE schemes than those by [ZZGQ23].

4.1 Identity-based Encryption

Reg-ABE for the IBE predicate is known as registreation-based encryption [GHMR18].

Definition 4.1 (Identity-based Encryption (IBE)). The predicate $\mathsf{P}^{\mathrm{IBE}}$ for IBE is defined as

$$\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p, \quad \mathsf{P}(x, y) = 1 \Leftrightarrow x = y$$

Encoding. A predicate encoding for $\mathsf{P}^{\mathrm{IBE}}$ is given as follows:

$$\mathbf{a}_y = (1, 0) \in \mathbb{Z}_p^{1\times 2}, \ \mathbf{K}_y = (y, 1) \in \mathbb{Z}_p^{1\times 2}, \ \mathbf{C}_x = \begin{pmatrix} 1 \\ -x \end{pmatrix} \in \mathbb{Z}_p^{2\times 1}, \ \mathbf{d}_{x,y} = (1)$$

We have $\omega = 1$, $n_c = 1$, $n_k = 2$.

Correctness: If $x = y$, we have $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = \left(\begin{smallmatrix} 1 \\ y-x \end{smallmatrix}\right) = (1, 0)^\top$.
Security: We have $\mathbf{M}_{x,y} = \left(\begin{smallmatrix} 1 \\ y-x \end{smallmatrix}\right)$, which never spans $(1, 0)^\top$ if $x \neq y$.
Compactness: We have $O(\log |\mathcal{Y}|) = O(\log p)$. Thus, $n_k = O(\log |\mathcal{Y}|)$.

4.2 Zero Inner Product

Definition 4.2 (Zero Inner Product Encryption (ZIPE)). The predicate $\mathsf{P}^{\mathrm{ZIPE}}$ for ZIPE is defined as

$$\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p^\ell, \quad \mathsf{P}(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle = 0$$

Encoding. A predicate encoding for $\mathsf{P}^{\mathrm{ZIPE}}$ is given as follows:

$$\mathbf{a}_y = (1, \mathbf{0}) \in \mathbb{Z}_p^{1\times(1+\ell)}, \;\; \mathbf{K}_y = (0, \mathbf{y}) \in \mathbb{Z}_p^{1\times(1+\ell)}, \;\; \mathbf{C}_x = \begin{pmatrix} 1 \\ \mathbf{x}^\top \end{pmatrix} \in \mathbb{Z}_p^{(1+\ell)\times 1}$$

$$\mathbf{d}_{x,y} = (1)$$

We have $\omega = 1$, $n_c = 1$, $n_k = \ell + 1$.

Correctness: If $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, we have $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = \left(\begin{smallmatrix} 1 \\ \langle \mathbf{x},\mathbf{y} \rangle \end{smallmatrix}\right) = (1, \mathbf{0})^\top$.
Security: We have $\mathbf{M}_{x,y} = \left(\begin{smallmatrix} 1 \\ \langle \mathbf{x},\mathbf{y} \rangle \end{smallmatrix}\right)$, which never spans $(1, \mathbf{0})^\top$ if $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$.
Compactness: We have $O(\log |\mathcal{Y}|) = O(\ell \log p)$. Thus, $n_k = O(\log |\mathcal{Y}|)$.

4.3 Non-zero Inner Product

Definition 4.3 (Non-zero Inner Product Encryption (NIPE)). The predicate $\mathsf{P}^{\mathrm{NIPE}}$ for NIPE is defined as

$$\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p^\ell, \quad \mathsf{P}(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle \neq 0$$

Encoding. A predicate encoding for $\mathsf{P}^{\mathrm{NIPE}}$ is given as follows:

$$\mathbf{a}_y = \mathbf{y} \in \mathbb{Z}_p^{1\times\ell}, \;\; \mathbf{K}_y = \perp, \;\; \mathbf{C}_x = \mathbf{x}^\top \in \mathbb{Z}_p^{\ell\times 1}, \;\; \mathbf{d}_{x,y} = \begin{pmatrix} 1 \\ \overline{\langle \mathbf{x}, \mathbf{y} \rangle} \end{pmatrix}$$

We have $\omega = 0$, $n_c = 1$, $n_k = \ell$.

Correctness: If $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$, we have $\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = (1)$.
Security: We have $\mathbf{M}_{x,y} = (\langle \mathbf{x}, \mathbf{y} \rangle)$, which never spans (1) if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.
Compactness: We have $O(\log |\mathcal{Y}|) = O(\ell \log p)$. Thus, $n_k = O(\log |\mathcal{Y}|)$.

4.4 Conjunctions

Definition 4.4 (Conjunctions). The predicate $\mathsf{P}^{\mathrm{AND}}$ for conjunctions is defined as

$$\mathcal{X} = \{0,1\}^\ell, \quad \mathcal{Y} = \{0,1\}^\ell, \quad \mathsf{P}(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \bigwedge_{i:x_i=1} (y_i = 1)$$

where $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$. Note that this is equivalent to the definition where $\mathcal{X} = \mathcal{Y} = 2^{[\ell]}$ and $\mathsf{P}^{\mathrm{AND}}(X, Y) = 1 \Leftrightarrow X \subseteq Y$.

$\mathsf{P}^{\mathrm{AND}}$ for policy size at most ℓ can be embedded in $\mathsf{P}^{\mathrm{ZIPE}}$ for vector length ℓ as follows. Given $\mathbf{y} \in \{0,1\}^\ell$, let $\bar{\mathbf{y}} = (1 - y_1, \ldots, 1 - y_\ell)$. Then, $\langle \mathbf{x}, \bar{\mathbf{y}} \rangle = 0$ if and only if $\bigwedge_{i:x_i=1} y_i = 1$. Thus, we can construct a predicate encoding for $\mathsf{P}^{\mathrm{AND}}$ from that for $\mathsf{P}^{\mathrm{ZIPE}}$.

4.5 Disjunctions

Definition 4.5 (Disjunctions). The predicate P^{OR} for disjunctions is defined as

$$\mathcal{X} = \{0,1\}^\ell, \quad \mathcal{Y} = \{0,1\}^\ell, \quad \mathsf{P}(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \bigvee_{i:x_i=1} (y_i = 1)$$

where $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$. Note that this is equivalent to the definition where $\mathcal{X} = \mathcal{Y} = 2^{[\ell]}$ and $\mathsf{P}^{\mathrm{OR}}(X, Y) = 1 \Leftrightarrow X \cap Y \neq \emptyset$.

P^{OR} for policy size at most ℓ can be embedded in $\mathsf{P}^{\mathrm{NIPE}}$ for vector length ℓ since $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$ if and only if $\bigvee_{i:x_i=1} y_i = 1$. Thus, we can construct a predicate encoding for P^{OR} from that for $\mathsf{P}^{\mathrm{NIPE}}$.

4.6 Broadcast Encryption

Definition 4.6 (Broadcast Encryption). The predicate P^{BE} for broadcast encryption is defined as

$$\mathcal{X} = \{S \subset \mathbb{Z}_p \backslash \{0\} \mid |S| \leq \ell\}, \quad \mathcal{Y} = \mathbb{Z}_p \backslash \{0\}, \quad \mathsf{P}(X, y) = 1 \Leftrightarrow y \in X$$

P^{BE} for broadcast size at most ℓ can be embedded in $\mathsf{P}^{\mathrm{ZIPE}}$ for vector length $\ell+1$ as follows. Given $X = \{x_1, \ldots, x_{\ell'}\} \in \mathcal{X}^{\mathrm{BE}}$ and $y \in \mathcal{Y}^{\mathrm{BE}}$ for P^{BE} such that $\ell' \leq \ell$, we define $\mathbf{x}$ and $\mathbf{y}$ as

$$\mathbf{x} = (c_0, \ldots, c_\ell), \quad \mathbf{y} = (1, y^1, \ldots, y^\ell)$$

where $c_0, \ldots, c_\ell$ satisfy $z^{\ell-\ell'} \prod_{i \in [\ell']} (z - x_i) = \sum_{i \in [0,\ell]} c_i z^i$ for a formal variable z. Then, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ if and only if $y \in X$. Thus, we can construct a predicate encoding for P^{BE} from that for $\mathsf{P}^{\mathrm{ZIPE}}$.

4.7 Boolean Span Programs

Definition 4.7 (Boolean Span Programs). Let $d \leq \ell$. The predicate $\mathsf{P}^{\mathrm{BSP}}$ for read-once Boolean span programs is defined as

$$\mathcal{X} = \mathbb{Z}_p^{\ell \times d}, \quad \mathcal{Y} = \{0,1\}^\ell, \quad \mathsf{P}(\mathbf{X}, \mathbf{y}) = 1 \Leftrightarrow \mathbf{X} \text{ accepts } \mathbf{y}$$

where $\mathbf{X}$ accepts $\mathbf{y} = (y_1, \ldots, y_\ell)$ if $\{\mathbf{x}_i\}_{i:y_i=1}$ span $(1, \mathbf{0})$, and $\mathbf{x}_i$ is the i-th row of $\mathbf{X}$.

Encoding. A predicate encoding for $\mathsf{P}^{\mathrm{BSP}}$ is given as follows:

$$\mathbf{a}_y = (1, \mathbf{0}) \in \mathbb{Z}_p^{1 \times (d+\ell)}, \quad \mathbf{K}_y = \begin{pmatrix} \mathbf{0}^\top & \mathbf{I}_{d-1} \\ \mathbf{0}^\top & \\ & \mathsf{diag}(\bar{\mathbf{y}}) \end{pmatrix} \in \mathbb{Z}_p^{(d-1+\ell) \times (d+\ell)}$$

$$\mathbf{C}_x = \begin{pmatrix} \mathbf{X}^\top \\ \mathbf{I}_\ell \end{pmatrix} \in \mathbb{Z}_p^{(d+\ell) \times \ell}, \quad \mathbf{d}_{x,y} = \mathbf{t} \in \mathbb{Z}_p^\ell$$

where $\mathsf{diag}(\bar{\mathbf{y}}) = \begin{pmatrix} 1-y_1 & & \\ & \ddots & \\ & & 1-y_\ell \end{pmatrix}$, $\mathbf{t} = (t_1, \ldots, t_\ell)$ is a vector such that $\mathsf{diag}(\bar{\mathbf{y}})\mathbf{t}^\top = \mathbf{0}^\top$ and $\mathbf{X}^\top \mathbf{t}^\top = (1, \mathbf{0})^\top$. It is not hard to see that such $\mathbf{t}$ exists if $\mathbf{X}$ accepts $\mathbf{y}$. We have $\omega = d-1+\ell$, $n_c = \ell$, $n_k = d+\ell$.

Correctness: If $\mathbf{X}$ accepts $\mathbf{y}$, we have

$$\mathbf{M}_{x,y} \mathbf{d}_{x,y}^\top = \begin{pmatrix} \mathbf{X}^\top \\ \mathsf{diag}(\bar{\mathbf{y}}) \end{pmatrix} \mathbf{t}^\top = (1, \mathbf{0})^\top$$

Security: We have $\mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{X}^\top \\ \mathsf{diag}(\bar{\mathbf{y}}) \end{pmatrix}$, the columns of which never span $(1, \mathbf{0})^\top$ if $\mathbf{X}$ does not accept $\mathbf{y}$.

This is because otherwise there exists $\mathbf{u} = (u_1, \ldots, u_\ell)$ such that $\mathbf{M}_{x,y} \mathbf{u}^\top = (1, \mathbf{0})^\top$. By definition, $u_i = 0$ if $y_i = 0$, which means that $\{\mathbf{x}_i^\top\}_{i:y_i=1}$ span $(1, \mathbf{0})^\top$. This contradicts with the condition that $\mathbf{X}$ does not accept $\mathbf{y}$.

Compactness: We have $O(\log |\mathcal{Y}|) = O(\ell)$. Thus, $n_k = O(\log |\mathcal{Y}|)$ since $d \leq \ell$.

Remark 4.1 (Relation to the HLWW scheme [HLWW23]). The above encoding essentially corresponds to the sReg-ABE scheme for Boolean span programs by [HLWW23], while their scheme is base on the composite-order groups. Our scheme for Boolean span programs can basically be captured as the prime-order variant of the composite-order scheme by [HLWW23]. See also the discussion in [ZZGQ23, Section 1.4].

4.8 Arithmetic Span Programs

Definition 4.8 (Arithmetic Span Programs). Let $d \leq \ell$. The predicate $\mathsf{P}^{\mathrm{ASP}}$ for read-once arithmetic span programs is defined as

$$\mathcal{X} = (\mathbb{Z}_p^{\ell \times d})^2, \quad \mathcal{Y} = \mathbb{Z}_p^{\ell}, \quad \mathsf{P}((\mathbf{X}, \mathbf{Z}), \mathbf{y}) = 1 \Leftrightarrow (\mathbf{X}, \mathbf{Z}) \text{ accepts } \mathbf{y}$$

where $(\mathbf{X}, \mathbf{Z})$ accepts $\mathbf{y} = (y_1, \ldots, y_\ell)$ if $\{\mathbf{v}_i\}_{i \in [\ell]}$ span $(1, \mathbf{0})$, $\mathbf{v}_i$ is the i-th row of $\mathbf{V} = \mathsf{diag}(\mathbf{y})\mathbf{X} + \mathbf{Z}$, and $\mathsf{diag}(\mathbf{y}) = \begin{pmatrix} y_1 & & \\ & \ddots & \\ & & y_\ell \end{pmatrix}$.

Encoding. A predicate encoding for $\mathsf{P}^{\mathrm{ASP}}$ is given as follows:

$$\mathbf{a}_y = (1, \mathbf{0}) \in \mathbb{Z}_p^{1 \times (d+2\ell)}, \quad \mathbf{K}_y = \begin{pmatrix} \mathbf{0}^\top & \mathbf{I}_{d-1} & \\ \mathbf{0}^\top & & \mathbf{I}_\ell \ -\mathsf{diag}(\mathbf{y}) \end{pmatrix} \in \mathbb{Z}_p^{(d-1+\ell) \times (d+2\ell)}$$

$$\mathbf{C}_x = \begin{pmatrix} \mathbf{X}^\top & \mathbf{Z}^\top \\ \mathbf{I}_\ell & \\ & \mathbf{I}_\ell \end{pmatrix} \in \mathbb{Z}_p^{(d+2\ell) \times 2\ell}, \quad \mathbf{d}_{x,y} = \mathbf{t}(\mathsf{diag}(\mathbf{y}) || \mathbf{I}_\ell) \in \mathbb{Z}_p^{\ell}$$

where $\mathbf{t} = (t_1, \ldots, t_\ell)$ is a vector such that $(\mathbf{X}^\top \mathsf{diag}(\mathbf{y}) + \mathbf{Z}^\top)\mathbf{t}^\top = (1, \mathbf{0})^\top$. We have $\omega = d - 1 + \ell$, $n_c = 2\ell$, $n_k = d + 2\ell$.

Correctness: If $(\mathbf{X}, \mathbf{Z})$ accepts $\mathbf{y}$, we have

$$\mathbf{M}_{x,y}\mathbf{d}_{x,y}^\top = \begin{pmatrix} \mathbf{X}^\top & \mathbf{Z}^\top \\ \mathbf{I}_\ell & -\mathsf{diag}(\mathbf{y}) \end{pmatrix} \begin{pmatrix} \mathsf{diag}(\mathbf{y}) \\ \mathbf{I}_\ell \end{pmatrix} \mathbf{t}^\top = \begin{pmatrix} \mathbf{X}^\top \mathsf{diag}(\mathbf{y}) + \mathbf{Z}^\top \\ \mathbf{O} \end{pmatrix} \mathbf{t}^\top = (1, \mathbf{0})^\top$$

Security: We have $\mathbf{M}_{x,y} = \begin{pmatrix} \mathbf{X}^\top & \mathbf{Z}^\top \\ \mathbf{I}_\ell & -\mathsf{diag}(\mathbf{y}) \end{pmatrix}$, the columns of which never span $(1, \mathbf{0})^\top$ if $(\mathbf{X}, \mathbf{Z})$ does not accept $\mathbf{y}$. This is because otherwise there exists $\mathbf{u} = (u_1, \ldots, u_{2\ell})$ such that $\mathbf{M}_{x,y}\mathbf{u}^\top = (1, \mathbf{0})^\top$. By definition, it must hold that $u_i = y_i u_{\ell+i}$, which means that the columns of $(\mathbf{X}^\top \mathsf{diag}(\mathbf{y}) + \mathbf{Z}^\top)$ span $(1, \mathbf{0})^\top$. This contradicts with the condition that $(\mathbf{X}, \mathbf{Z})$ does not accept $\mathbf{y}$.

Compactness: We have $O(\log |\mathcal{Y}|) = O(\ell \log p)$. Thus, $n_k = O(\log |\mathcal{Y}|)$ since $d \leq \ell$.

4.9 Summary of Encoding Sizes

In Table 3, we summarize the encoding sizes of each predicate for both of the original encodings and our new encodings. By plugging the parameters into the efficiency in Section 3.1, we obtain the concrete space efficiency of Reg-ABE schemes in Table 1. A caveat is that the original encodings are not compatible with our compiler and needs the ZZGQ compiler [ZZGQ23]. Since the asymptotic efficiency of the ZZGQ compiler and ours is different (see the last two paragraphs in Section 3.1), a comparison between the sizes of the original encodings and new ones is not informative. An apple-to-apple comparison of the resulting schemes is provided in Table 1.

5 Instantiations of sReg-ABE

In this section, we describe some concrete sReg-ABE schemes that are obtained by applying the encodings in Section 4 to the compiler in Section 3.1. Specifically, we provide sReg-ABE schemes for ZIPE and NIPE with constant-size ciphertexts. As in Section 3.1, let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of bilinear groups where $\mathbb{G}_\lambda = (p, G_1, G_2, G_T, g_1, g_2, e)$. Let $\Pi = (\mathsf{LGen}, \mathsf{LProve}, \mathsf{LVerify}, \mathsf{LSim})$ be a QA-NIZK scheme. Note that since Gen and Ver do not depend on predicate encodings (i.e., they are the same as in Section 3.1), we omit them here.

Table 3. Comparison of encoding sizes between the original one and ours.

Predicate	Original encoding [Wee14, CGW15]			Ours		
	ω	n_c	n_k	ω	n_c	n_k
IBE	2	1	1	1	1	2
ZIPE	$\ell+1$	ℓ	1	1	1	$\ell+1$
NIPE	$\ell+1$	ℓ	2	0	1	ℓ
BSP	$d+\ell$	ℓ	$\ell+1$	$d+\ell-1$	ℓ	$d+\ell$
ASP	$d+2\ell$	2ℓ	$\ell+1$	$d+\ell-1$	2ℓ	$d+2\ell$

Note: The parameter ℓ refers to the vector length in NIPE and ZIPE while it refers to the attribute size in BSP and ASP. The parameter d refers to the width of span programs.

5.1 sReg-ABE Scheme for Zero Inner Product (ZIPE)

$\mathsf{Setup}(1^\lambda, 1^L)$**:** It outputs crs as follows.

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(2k+1)},\ \mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k},\ \mathbf{h} \leftarrow \mathbb{Z}_p^{2k+1}$$
$$\left\{\begin{array}{l}\mathbf{W}_i \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)},\ \mathbf{W}_{i,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)}\\ \mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+1)},\ \mathbf{r}_i \leftarrow \mathbb{Z}_p^k\\ (\mathsf{crs}_i, \mathsf{td}_i) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}_i]_1) \text{ where } \mathbf{A}_i = \left(\begin{smallmatrix}\mathbf{A}\\ \mathbf{R}_i\end{smallmatrix}\right)\end{array}\right\}_{i\in[L]}$$
$$\mathsf{crs} = \left(\begin{array}{l}[\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{A}\mathbf{W}_{i,0}, \mathbf{A}\mathbf{W}_i]_1\}_{i\in[L]}\\ \{[\mathbf{B}\mathbf{r}_i^\top, \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top]_2\}_{i\in[L]}, \{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top, \mathbf{W}_i\mathbf{B}\mathbf{r}_j^\top]_2\}_{\substack{i,j\in[L]\\ i\neq j}}\end{array}\right)$$

$\mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_i, \mathbf{y}_i\}_{i\in[L]})$**:** It parses $\mathsf{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{p}_{i,j}^\top]_2\}_{j\in[L]\setminus\{i\}}, \pi_i)$. It outputs mpk and $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$ as follows:

$$\mathsf{mpk} = \left([\mathbf{A}\mathbf{h}^\top]_T, \left[\mathbf{A}, \underbrace{\sum_{i\in[L]}(\mathbf{A}\mathbf{W}_{i,0} + \mathbf{T}_i)}_{\mathbf{P}_1}, \underbrace{\sum_{i\in[L]}\mathbf{y}_i \otimes \mathbf{A}\mathbf{W}_i}_{\mathbf{P}_2}\right]_1\right)$$
$$\mathsf{hsk}_i = \left[\underbrace{\mathbf{B}\mathbf{r}_i^\top}_{\mathbf{h}_1^\top}, \underbrace{\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top}_{\mathbf{h}_2^\top}, \underbrace{\sum_{j\in[L]\setminus\{i\}}(\mathbf{W}_{j,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{p}_{j,i}^\top)}_{\mathbf{h}_3^\top}, \underbrace{\sum_{j\in[L]\setminus\{i\}}\mathbf{y}_i \otimes \mathbf{W}_i\mathbf{B}\mathbf{r}_j^\top}_{\mathbf{H}_4}\right]_2$$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, M)$**:** It takes mpk, $\mathbf{x}$, and $M \in G_T$ as inputs, and outputs ct_x by computing as follows. It parses mpk as above. It computes

$$\mathbf{s} \leftarrow \mathbb{Z}_p^k,\ \mathsf{ct}_x = ([\underbrace{\mathbf{s}\mathbf{A}}_{\mathbf{c}_1}, \underbrace{\mathbf{s}(\mathbf{P}_1 + \mathbf{P}_2(\mathbf{x}^\top \otimes \mathbf{I}_{k+1}))}_{\mathbf{c}_2}]_1,\ \underbrace{[\mathbf{s}\mathbf{A}\mathbf{h}^\top]_T M}_{C})$$

$\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x, \mathbf{x}, \mathbf{y}_i)$**:** It parses $\mathsf{sk}_i = \mathbf{V}_i$, $\mathsf{hsk}_i = ([\mathbf{h}_1^\top, \mathbf{h}_2^\top, \mathbf{h}_3^\top, \mathbf{H}_4]_2)$, $\mathsf{ct}_x = ([\mathbf{c}_1, \mathbf{c}_2]_1, C)$. It outputs $M' = C \cdot [z_1 - z_2 - z_3]_T$ where

$$[z_1]_T = e([\mathbf{c}_2]_1, [\mathbf{h}_1^\top]_2),\ [z_2]_T = e([\mathbf{c}_1]_1, [\mathbf{h}_3^\top + \mathbf{H}_4\mathbf{x}^\top]_2)$$
$$[z_3]_T = e([\mathbf{c}_1]_1, [\mathbf{h}_2^\top]_2) \cdot e([\mathbf{c}_1]_1, [\mathbf{V}_i\mathbf{h}_1^\top]_2)$$

Remark 5.1 (Slotted Reg-ABE Schemes for Conjunctions and Broadcast). By using the standard embedding in Sections 4.4 and 4.6, the above scheme can be used as a slotted Reg-ABE scheme for conjunctions or broadcast encryption with constant-size ciphertexts.

5.2 sReg-ABE Scheme for Non-zero Inner Product (NIPE)

$\mathsf{Setup}(1^\lambda, 1^L)$**:** It outputs crs as follows.

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(2k+1)},\ \mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1)\times k},\ \mathbf{h} \leftarrow \mathbb{Z}_p^{2k+1}$$
$$\left\{\begin{array}{l}\mathbf{W}_{i,0} \leftarrow \mathbb{Z}_p^{(2k+1)\times(k+1)},\ \mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2)\times(2k+1)},\ \mathbf{r}_i \leftarrow \mathbb{Z}_p^k \\ (\mathsf{crs}_i, \mathsf{td}_i) \leftarrow \mathsf{LGen}(1^\lambda, [\mathbf{A}_i]_1) \ \text{ where } \mathbf{A}_i = \left(\begin{smallmatrix}\mathbf{A}\\ \mathbf{R}_i\end{smallmatrix}\right)\end{array}\right\}_{i\in[L]}$$
$$\mathsf{crs} = \left(\begin{array}{l}[\mathbf{A}]_1, [\mathbf{A}\mathbf{h}^\top]_T, \{\mathsf{crs}_i, [\mathbf{R}_i, \mathbf{A}\mathbf{W}_{i,0}]_1\}_{i\in[L]} \\ \{[\mathbf{B}\mathbf{r}_i^\top, \mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top]_2\}_{i\in[L]}, \{[\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_j^\top]_2\}_{\substack{i,j\in[L]\\ i\neq j}}\end{array}\right)$$

$\mathsf{Agg}(\mathsf{crs}, \{\mathsf{pk}_i, \mathbf{y}_i\}_{i\in[L]})$**:** It parses $\mathsf{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{p}_{i,j}^\top]_2\}_{j\in[L]\setminus\{i\}}, \pi_i)$. It outputs mpk and $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$ as follows:

$$\mathsf{mpk} = \left([\mathbf{A}\mathbf{h}^\top]_T, \left[\mathbf{A}, \underbrace{\sum_{i\in[L]} \mathbf{y}_i \otimes (\mathbf{A}\mathbf{W}_{i,0} + \mathbf{T}_i)}_{\mathbf{P}_1}\right]_1\right)$$
$$\mathsf{hsk}_i = \left[\underbrace{\mathbf{B}\mathbf{r}_i^\top}_{\mathbf{h}_1^\top}, \underbrace{\mathbf{W}_{i,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{h}^\top}_{\mathbf{h}_2^\top}, \underbrace{\sum_{j\in[L]\setminus\{i\}} \mathbf{y}_i \otimes (\mathbf{W}_{j,0}\mathbf{B}\mathbf{r}_i^\top + \mathbf{p}_{j,i}^\top)}_{\mathbf{H}_3}\right]_2$$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, M)$**:** It takes mpk, $\mathbf{x}$, and $M \in G_T$ as inputs, and outputs ct_x by computing as follows. It parses mpk as above. It computes

$$\mathbf{s} \leftarrow \mathbb{Z}_p^k,\ \mathsf{ct}_x = ([\underbrace{\mathbf{s}\mathbf{A}}_{\mathbf{c}_1}, \underbrace{\mathbf{s}(\mathbf{P}_1(\mathbf{x}^\top \otimes \mathbf{I}_{k+1}))}_{\mathbf{c}_2}]_1,\ \underbrace{[\mathbf{s}\mathbf{A}\mathbf{h}^\top]_T M}_{C})$$

$\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x, \mathbf{x}, \mathbf{y}_i)$**:** It parses $\mathsf{sk}_i = \mathbf{V}_i$, $\mathsf{hsk}_i = ([\mathbf{h}_1^\top, \mathbf{h}_2^\top, \mathbf{H}_3]_2)$, $\mathsf{ct}_x = ([\mathbf{c}_1, \mathbf{c}_2]_1, C)$. It outputs $M' = C \cdot \left[\frac{z_1 - z_2}{\langle \mathbf{x}, \mathbf{y}_i\rangle} - z_3\right]_T$ where

$$[z_1]_T = e([\mathbf{c}_2]_1, [\mathbf{h}_1^\top]_2),\ [z_2]_T = e([\mathbf{c}_1]_1, [\mathbf{H}_3\mathbf{x}^\top]_2)$$
$$[z_3]_T = e([\mathbf{c}_1]_1, [\mathbf{h}_2^\top]_2) \cdot e([\mathbf{c}_1]_1, [\mathbf{V}_i\mathbf{h}_1^\top]_2)$$

Remark 5.2 (Slotted Reg-ABE Schemes for Disjunctions). By using the standard embedding in Section 4.5, the above scheme can be used as a slotted Reg-ABE scheme for disjunctions with constant-size ciphertexts.

References

ABS17. Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt. Generic transformations of predicate encodings: Constructions and applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 36–66, August 2017.

AC16. Shashank Agrawal and Melissa Chase. A study of pair encodings: Predicate encryption in prime order groups. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 259–288, January 2016.

AT24. Nuttapong Attrapadung and Junichi Tomida. A modular approach to registered ABE for unbounded predicates. In *CRYPTO 2024, Part III*, LNCS, pages 280–316, August 2024.

Att14. Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577, May 2014.

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

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499, August 2014.

CES21. Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing registration based encryption. In Maura B. Paterson, editor, *18th IMA International Conference on Cryptography and Coding*, volume 13129 of *LNCS*, pages 129–157, December 2021.

CGW15. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624, April 2015.

DKL+23. Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 417–446, April 2023.

DPY23. Pratish Datta, Tapas Pal, and Shota Yamada. Registered fe beyond predicates: (attribute-based) linear functions and more. Cryptology ePrint Archive, Paper 2023/457, 2023. https://eprint.iacr.org/2023/457.

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, August 2013.

FFM+23. Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. LNCS, pages 98–133, December 2023.

FKdP23. Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023*, volume 14442, pages 166–200. Springer, 2023.

FN94. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491, August 1994.

FWW23. Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 498–531, August 2023.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GHK+25. Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. In *CRYPTO 2025, Part III*, LNCS, pages 504–539, August 2025.

GHM+19. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 63–93, April 2019.

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

GKMR23. Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In *ACM CCS 2023*, pages 1065–1079. ACM Press, November 2023.

GLWW24. Rachit Garg, George Lu, Brent Waters, and David J. Wu. Reducing the CRS size in registered ABE systems. In *CRYPTO 2024, Part III*, LNCS, pages 143–177, August 2024.

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

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, August 2020.

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, April 2023.

IW14. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20, December 2013.

KMW23. Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. LNCS, pages 407–441, December 2023.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, April 2008.

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128, April 2015.

LOS+10. Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91, May / June 2010.

LWW25. George Lu, Brent Waters, and David J. Wu. Multi-authority registered attribute-based encryption. In *EUROCRYPT 2025, Part III*, LNCS, pages 3–33, June 2025.

PS25. Tapas Pal and Robert Schädlich. A general framework for registered functional encryption via user-specific pre-constraining. LNCS, pages 3–34, December 2025.

RT26. Yannis Rouselakis and Junichi Tomida. Towards practical registered ABE: more efficient, non-monotone, and cca-secure. *IACR Cryptol. ePrint Arch.*, 2026:414, 2026.

SWW26. Roy Stracovsky, Brent Waters, and David J. Wu. Pairing-based registered ABE for boolean formulas with a linear-size CRS. *IACR Cryptol. ePrint Arch.*, 2026:1062, 2026.

Wee14. Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637, February 2014.

WQZD10. Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption (poster presentation). In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 741–743. ACM Press, October 2010.

ZZGQ23. Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered ABE via predicate encodings. LNCS, pages 66–97, December 2023.

ZZZ+25. Ziqi Zhu, Jun Zhao, Kai Zhang, Junqing Gong, and Haifeng Qian. When KGC meets curator: New paradigm of registered ABE and FE. LNCS, pages 35–67, December 2025.

A Registered ABE

We follow the definition of bounded-user registered ABE in [HLWW23, ZZGQ23]. The definition below is taken mostly verbatim from [ZZGQ23, Appendix A].

Definition A.1 (Bounded-user Registered ABE [HLWW23, ZZGQ23]). Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y}$ be a predicate. Let $\mathcal{M}$ be a message space. A bounded-user registered ABE scheme for $\mathcal{P}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^L)$**:** It takes a security parameter 1^λ, the number of users in unary 1^L and outputs a common reference string crs.[7]

$\mathsf{Gen}(\mathsf{crs}, \mathsf{aux})$**:** It takes crs and a public state aux as input, and outputs a public key pk and a secret key sk.

$\mathsf{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, y)$**:** It takes $\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, y \in \mathcal{Y}$ as input, and outputs a master public key mpk and updated state aux.

$\mathsf{Enc}(\mathsf{mpk}, x, M)$**:** It takes mpk, $x \in \mathcal{X}$, and a message $M \in \mathcal{M}$ as inputs, and outputs a ciphertext ct_x.

$\mathsf{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk})$**:** It takes $(\mathsf{crs}, \mathsf{aux}, \mathsf{pk})$ as input and outputs a helper secret key hsk.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}_x)$**:** It takes $(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}_x)$ as input, and outputs a message M or a symbol $\perp$ or a special flag `getupd` to indicate the need of an updated helper key.

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

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

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

- $\mathsf{ORegNT}(\mathsf{pk}, y)$: run $(\mathsf{mpk}, \mathsf{aux}') \leftarrow \mathsf{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, y)$, update $\mathsf{aux} = \mathsf{aux}'$, append $(\mathsf{mpk}, \mathsf{aux})$ to $\mathcal{R}$ and return $(|\mathcal{R}|, \mathsf{mpk}, \mathsf{aux})$;
- $\mathsf{ORegT}(y^*)$: run $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{aux})$, $(\mathsf{mpk}, \mathsf{aux}') \leftarrow \mathsf{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*, y^*)$, update $\mathsf{aux} = \mathsf{aux}'$, compute $\mathsf{hsk}^* \leftarrow \mathsf{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$, append $(\mathsf{mpk}, y^*, \mathsf{aux})$ to $\mathcal{R}$, return $(t = |\mathcal{R}|, \mathsf{mpk}, \mathsf{aux}, \mathsf{pk}^*, \mathsf{sk}^*, \mathsf{hsk}^*)$;
- $\mathsf{OEnc}(i, x, M)$: let $\mathcal{R}[i] = (\mathsf{mpk}, *, *)$, run $\mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, M)$, append (x, M, ct_x) to $\mathcal{E}$ and return $(|\mathcal{E}|, \mathsf{ct})$;
- $\mathsf{ODec}(j)$: let $\mathcal{E}[j] = (x_j, M_j, \mathsf{ct}_{x_j})$, compute $M'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$, if $M'_j = \mathtt{getupd}$, run $\mathsf{hsk}^* \leftarrow \mathsf{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ and recompute $M'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. Set $b = 1$ when $M'_j \neq M_j$;

with the following restrictions:

- there exists one query to ORegT (we can consider $y^*, \mathsf{pk}^*, \mathsf{sk}^*, \mathsf{hsk}^*$ to be global);
- for query $(i, x, *)$ to OEnc, it holds that $i \geq t, \mathcal{R}[i] \neq \perp$ and $\mathsf{P}(x, y^*) = 1$;
- for query j to ODec, it holds that $\mathcal{E}[j] \neq \perp$.

Compactness and Efficiency. Let $\mathcal{R}$ be defined as before. Compactness refers to the property that

$$|\mathsf{mpk}_i| = \mathsf{poly}(\lambda, \log|\mathcal{Y}|, \log i), \quad |\mathsf{hsk}^*| = \mathsf{poly}(\lambda, \log|\mathcal{Y}|, \log|\mathcal{R}|)$$

where we let $\mathcal{R}[i] = (\mathsf{mpk}_i, y_i, *)$ for all $i \in [|\mathcal{R}|]$. Furthermore, update efficiency means that the number of invocations of Upd in ODec is at most $O(\log|\mathcal{R}|)$ and each invocation costs $\mathsf{poly}(\log|\mathcal{R}|)$ time (in RAM model).

Security. For all stateful admissible adversaries $\mathcal{A}$, the advantage

$$\Pr\left[\beta = \beta' \ : \ \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ x, M_0, M_1 \leftarrow \mathcal{A}^{\mathsf{ORegCK}(\cdot),\mathsf{ORegHK}(\cdot),\mathsf{OCor}(\cdot)}(\mathsf{crs}) \\ \beta \leftarrow \{0,1\};\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, M_\beta);\ \beta' \leftarrow \mathcal{A}(\mathsf{ct}_x) \end{array}\right] - \frac{1}{2}$$

is negligible in λ, where the oracles as follows with initial setting $\mathsf{aux}, \mathsf{mpk} = \perp, \mathcal{R} = \emptyset, \mathcal{C} = \emptyset$ and $\mathcal{D}$ being a dictionary with $\mathcal{D}[\mathsf{pk}] = \emptyset$ for all possible pk:

[7] When considering bounded-user schemes, it is mandatory to input 1^L in unary. This is already discussed and defined in [HLWW23, Def 4.4]. Note that the definition in [ZZGQ23], which also consider bounded-user schemes, does not contain 1^L; we correct it here.

- $\mathsf{ORegCK}(\mathsf{pk}, y)$: run $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathsf{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, y)$, update $\mathsf{mpk} = \mathsf{mpk}', \mathsf{aux} = \mathsf{aux}', \mathcal{D}[\mathsf{pk}] \cup \{y\}$, append pk to $\mathcal{C}$ and return $(\mathsf{mpk}, \mathsf{aux})$;
- $\mathsf{ORegHK}(y)$: run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{aux})$ and $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathsf{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, y)$, update $\mathsf{mpk} = \mathsf{mpk}', \mathsf{aux} = \mathsf{aux}', \mathcal{D}[\mathsf{pk}] \cup \{y\}$, append $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{R}$ and return $(|\mathcal{R}|, \mathsf{mpk}, \mathsf{aux}, \mathsf{pk})$;
- $\mathsf{OCor}(i)$: let $\mathcal{R}[i] = (\mathsf{pk}, \mathsf{sk})$, append pk to $\mathcal{C}$ and return sk;

with the following restrictions

- for query i to OCor, it holds that $\mathcal{R}[i] \neq \perp$;
- for all $y \in \bigcup_{\mathsf{pk} \in \mathcal{C}} \mathcal{D}[\mathsf{pk}]$, it holds that $\mathsf{P}(x, y) = 0$.

Conversion from Slotted Registered ABE to Registered ABE. Hohenberger *et al.* shows a generic conversion so called the powers of two construction, which converts slotted registered ABE to registered ABE and also shows that the conversion preserves the (un)bounded-user property and compactness [HLWW23]. By inspection of the conversion of [HLWW23], the size expansions of $|\mathsf{mpk}|$, $|\mathsf{hsk}_i|$, $|\mathsf{ct}|$ are at most logarithmic in the maximum number of registered users. Starting from our bounded-user slotted registered ABE in §3.1, we obtain bounded-user registered ABE for the same predicate.