Non-Committing Attribute and Identity-Based Encryption: Constructions and Applications*

Rishab Goyal[◇], Fuyuki Kitagawa[†], Venkata Koppula[⋆], Ryo Nishimaki[†], Mahesh Sreekumar Rajasree[†⁕], Takashi Yamakawa[†]

[◇]UW-Madison
rishab@cs.wisc.edu
[⋆]IIT Delhi
kvenkata@cse.iitd.ac.in
[⁕]University of Warsaw, Poland
srmahesh1994@gmail.com
[†]NTT Social Informatics Laboratories, Tokyo, Japan
{fuyuki.kitagawa,ryo.nishimaki,takashi.yamakawa}@ntt.com

Abstract

A receiver non-committing encryption (RNCE) scheme [Canetti *et al.*, STOC 1996; Canetti *et al.*, TCC 2005] allows one to sample a public key pk and (dummy) ciphertext ct without knowing the message m. Later, when the message is known, one can sample a secret key sk that looks like the secret key corresponding to pk, and decryption of ct produces m. In this work, we study receiver non-committing attribute-based encryption (RNC-ABE) and identity-based encryption (RNC-IBE). We give constructions based on standard assumptions on bilinear groups (prior works [Hiroka *et al.*, ASIACRYPT 2021] require indistinguishability obfuscation).

Our RNC-ABE and RNC-IBE constructions have important implications for incompressible attribute- and identity-based encryption. These notions were recently introduced by Goyal *et al.*, ITCS 2025. However, there were no constructions for the strongest security definitions in Goyal *et al.*, ITCS 2025. Our RNC-ABE and RNC-IBE schemes also lead to the first incompressible ABE and IBE schemes with optimal ciphertext size, which was another open question in Goyal *et al.*, ITCS 2025.

We also give constructions for relaxed RNC-IBE (where the identity space is polynomial in the security parameter, but the public key is compact) that are based on DDH or LWE. This leads to a relaxed incompressible IBE scheme with strong security under the same assumptions.

Keywords: non-committing, attribute-based encryption, identity-based encryption, incompressible encryption.

1 Introduction

Non-committing encryption. Non-committing public-key encryption (NCE), introduced by Canetti, Feige, Goldreich and Naor [CFGN96], is a crucial cryptographic tool in the design of adaptively secure multiparty computation protocols [CFGN96, CLOS02]. An NCE scheme consists of the following algorithms: $\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}$ and $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$. The syntax for $\mathsf{Setup}, \mathsf{Enc}$ and Dec mirrors that of (standard) public-key encryption (PKE) schemes. Additionally, the simulator Sim enables the sampling of a ciphertext without knowledge of the underlying message.

*A preliminary version of this paper appeared at PKC 2025 [GKK+25].
†This work was done while the author was a postdoctoral fellow at IIT Delhi, India.

Once the message is revealed, the simulator can generate the necessary randomness to reconcile the public key and the ciphertext.

More formally, security is captured using the real and ideal world framework. In the real world, the adversary first receives the public key. It then sends a message m, and receives an encryption of m, together with the randomness r_{enc} used for encryption, and the randomness r_{setup} used for sampling the public key. In the ideal world, the simulator Sim_1 produces the public key pk and ciphertext ct^* (together with internal state st which is passed to Sim_2). After the adversary receives the public key, it sends the message m^*. The second-stage simulator Sim_2 receives the message m^* (and the internal state st), samples randomness r_{setup} that can explain pk, and randomness r_{enc} to explain ct^*. The adversary receives $\mathsf{ct}^*, r_{\mathsf{enc}}$ and r_{setup}, and must distinguish between the real world and the ideal world. Today, we have several constructions of NCE, from a wide range of assumptions [CDSMW09, HORR15, CPR17, YKT19, BBD+20], as well as a good understanding of the barriers [Nie02].

Prior works have also explored weaker notions of non-committing encryption. Receiver NCE (RNCE) [1] is one such relaxation that has garnered significant attention. Here, the simulator Sim_2 only needs to output the secret key corresponding to pk (but does not need to produce the randomness r_{enc} and r_{setup}). Receiver NCE has been used for applications such as designing secure multiparty computation protocols [CFGN96, CLOS02], adaptively secure attribute-based encryption for Turing machines [GSW21], selective-opening-secure schemes [HPW15], and more. In this work, we will focus on RNCE, with the aim of going beyond public-key encryption.

Receiver Non-committing Identity-Based Encryption (and beyond). Identity-based encryption (IBE) [Sha85] is a powerful generalization of public-key encryption, where users can encrypt messages for any identity using a master public key. The master public key, together with a corresponding master secret key, is sampled by the master authority using a Setup algorithm. The master secret key can be used to issue secret keys for every identity using a KeyGen algorithm. Using a secret key for identity id, one can decrypt a ciphertext for identity id. Intuitively, security says that even if an adversary has polynomially many secret keys corresponding to identities of its choice, if it does not have a secret key for id^*, then the adversary cannot decrypt an encryption for id^*.

Attribute-based encryption (ABE) [SW05, GPSW06], in the key-policy setting, is a further generalization of IBE where secret keys are associated with predicates and users can encrypt a message for an attribute using a master public key. Using a secret key for a predicate f, one can decrypt any ciphertext associated with attribute x as long as $f(x) = 1$. Similar to the IBE setting, security says that even if an adversary has polynomially many secret keys corresponding to predicates of its choice, as long as all queried predicates f satisfy $f(x^*) = 0$, the adversary cannot decrypt an encryption for x^*.

In the non-committing setting, Hiroka, Morimae, Nishimaki and Yamakawa [HMNY21] introduced non-committing ABE in the context of attribute-based quantum encryption with certified deletion; this can be viewed as a stronger version of non-committing IBE.

In a receiver non-committing identity-based encryption (RNC-IBE) scheme, in addition to $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we have a simulator Sim that can produce the master public key, secret keys, and challenge ciphertext without knowing the challenge message. Later, when the message is revealed, the simulator can sample a master secret key that is consistent with the master public key, secret keys, and the challenge ciphertext.

More formally, in the real world, the adversary receives the master public key mpk. Then, it can send polynomially many identities and receives the secret keys corresponding to these identities. The adversary then sends the challenge identity id^*, together with the challenge message m^*, and receives $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$ together with the master secret key msk. In the ideal world, the adversary interacts with a simulator. The simulator first sends the master public key mpk. Then, the adversary receives secret keys for identities of its choice. Finally, in the challenge phase, after the adversary sends the challenge identity id^* together with the challenge message m^*, the simulator first generates a challenge ciphertext ct^* that depends only on id^*, and then, using m^*, produces a master secret key msk consistent with the master public key, the secret keys already issued, and the challenge ciphertext. The adversary finally receives ct^* and msk, and must distinguish between the real and ideal worlds.

While we have several constructions for IBE [BF01, Coc01, HL02, GS02, CHK03, BB04a, BB04b, Wat05, BBG05, BGH07, GPV08, SW08, AB09, GH09, ABB10a, ABB10b, LW10, CHKP12, CGW15, DG17b, DG17a, BLSV18, WC23, HKK+24, GKR25] (and even ABE [SW05, GPSW06, BSW07, Wat12, Att14, Att16, GWW19, GW20, GSW21]), our

[1] Also referred to as weak NCE [GSW21].

understanding with respect to RNC-IBE and RNC-ABE is very limited. Hiroka *et al.* [HMNY21] gave a construction for RNC-ABE using indistinguishability obfuscation. This brings us to the first central question of our work.

Q1. Can we construct RNC-ABE and RNC-IBE from the same assumptions that give us (regular) ABE and IBE?

Besides being a natural question in itself, this would resolve interesting open questions in the landscape of incompressible cryptography, which we discuss next.

Incompressible (Attribute-based and Identity-Based) Encryption. The concept of incompressible public key encryption was introduced by Guan, Wichs and Zhandry [GWZ22] to address scenarios where the adversary eventually receives the entire secret decryption key, but has limited long-term storage, and as a result, cannot store the entire ciphertext. For S-incompressible security, we require that no adversary should distinguish in the following game with non-negligible advantage: the adversary, after receiving the public key pk, sends two challenge messages m_0, m_1, and receives the challenge ciphertext ct^*. It must then compress the ciphertext into a short state st of size at most S bits. After it computes the compressed state st, it receives the secret key sk, and must guess whether m_0 was encrypted or m_1.

Guan *et al.* gave two constructions of incompressible PKE: one based on general PKE (with ciphertext size being $(S + |m|) \cdot \mathrm{poly}(\lambda)$), and another based on indistinguishability obfuscation (with ciphertext size $\max(S, |m|) + \mathrm{poly}(\lambda)$).

Later works [GWZ23, GKRV25] observed that any RNCE scheme can be used to build an incompressible PKE scheme with ciphertext size $S + |m| + \mathrm{poly}(\lambda)$ as follows. Consider an RNCE scheme and an incompressible SKE scheme with a secret key size of $\mathrm{poly}(\lambda)$ and a ciphertext size $(|m| + S + \mathrm{poly}(\lambda))$ (Dziembowski [Dzi06] gave a construction of such an incompressible SKE scheme, based on one-way functions). In the incompressible PKE scheme, the public and secret keys are identical to those of the RNCE scheme. To encrypt a message m, first generate a fresh secret key $\mathsf{inc.sk}$ for the incompressible SKE scheme. Using this key, encrypt the message to produce an incompressible ciphertext $\mathsf{inc.ct}$. Next, encrypt $\mathsf{inc.sk}$ using the RNCE to obtain $\mathsf{nce.ct}$, and the final ciphertext becomes $\mathsf{ct} := (\mathsf{nce.ct}, \mathsf{inc.ct})$. Since $|\mathsf{inc.sk}| = \mathrm{poly}(\lambda)$, we have $|\mathsf{nce.ct}| = \mathrm{poly}(\lambda)$. Therefore, the total ciphertext size is $|\mathsf{ct}| = |m| + S + \mathrm{poly}(\lambda)$, and hence the scheme is ciphertext-rate preserving.

To argue security, we begin by switching the RNCE to simulation mode, which allows us to freely program the secret key so that $\mathsf{nce.ct}$ can decrypt to any desired value. At this point, the scheme's security relies on the incompressibility of the SKE scheme.

In this work, we focus on incompressible attribute- and identity-based encryption schemes. These primitives, introduced in a recent work by Goyal, Koppula, Rajasree and Verma [GKRV25], area natural generalization of incompressible PKE to the ABE and IBE settings. Here again, the syntax is the same as that of (regular) ABE and IBE. For incompressibility security, however, note that there can be multiple flavors of security. Goyal *et al.* defined two notions of security for incompressible ABE and IBE:

- (regular) incompressible security: in this case, the adversary first receives the master public key. Then it can send polynomially many identities (in the case of ABE, predicates), and receives secret keys for these identities (predicates). During the challenge phase, the adversary sends a challenge identity id^* (attribute x^*) together with challenge messages m_0, m_1, and receives the challenge ciphertext, followed by post-challenge secret key queries (similar to the pre-challenge secret key queries). Finally, the adversary must compress the ciphertext into a short state st of size at most S bits. After this, it receives the secret key corresponding to id^* (a distinguishing predicate f^*, that is, $f^*(x^*) = 1$), and must guess whether m_0 was encrypted or m_1.
- strong incompressible security: this game is similar to that of regular incompressible security. However, instead of receiving the secret key for id^* (predicate f^*) at the end, the adversary receives the entire master secret key.

Goyal *et al.* gave constructions for (regular) incompressible ABE and IBE; however, there were no constructions achieving strong incompressibility. Our first observation is that the connection between incompressible PKE and RNCE also extends to ABE and IBE, and as a result, if we construct an RNC-ABE or RNC-IBE scheme, then that also resolves the following question (left open in [GKRV25]).

Q2. Can we construct strongly incompressible ABE and IBE?

Ciphertext rate of incompressible encryption schemes. An important parameter in the design of incompressible encryption schemes is the size of the ciphertext, as a function of the message size and the adversary's long-term storage bound S. The optimal ciphertext size (ignoring dependence on λ) is $\max(S, |m|)$. Guan *et al.* [GWZ22] showed how to construct incompressible PKE schemes with optimal ciphertext size, using indistinguishability obfuscation. Later, Branco, Döttling and Dujmovic [BDD22] showed how to construct incompressible PKE schemes with optimal ciphertext size using standard assumptions (that is, without using obfuscation). A natural question is whether we can achieve incompressible ABE and IBE schemes with optimal ciphertext size. This was also left as an open question by Goyal *et al.* [GKRV25].

Q3. Can we construct incompressible ABE and IBE schemes with optimal ciphertext size?

1.1 Our results

In this work, we introduce new constructions for RNC-ABE for predicate classes with predicate encodings and RNC-IBE under standard assumptions. We also obtain the corresponding receiver non-committing key encapsulation mechanisms. Additionally, we give a construction from iO and one-way functions. These constructions, in turn, give us the first strongly incompressible ABE schemes for predicate classes with predicate encodings and IBE schemes. In particular, our SXDH-based construction yields the first such schemes with optimal ciphertext size.

Main Results: The first construction is based on the SXDH assumption, utilizing the dual system technique of Waters [Wat09]. Notably, this construction achieves a robust security notion where the adversary obtains the entire randomness used by the setup algorithm (see Theorem B.1 for further details). Also, the size of the ciphertext is independent of the size of the session key.

Theorem 1.1. *Assuming the hardness of the* SXDH *problem, there exists an adaptively secure RNC-AB-KEM for predicate classes with predicate encodings and RNC-IB-KEM. Additionally, the adversary is allowed to learn the entire setup randomness, and the size of the ciphertext is independent of the size of the session key.*

Note that Theorem 1.1 addresses Q1 from above. Additionally, it resolves questions Q2 and Q3 simultaneously! More generally, by combining the RNC-AB-KEM for a predicate class with predicate encodings or RNC-IB-KEM with the incompressible secret key encryption scheme in [BDD22] (based on the LWE or DCR assumptions), we obtain a strongly incompressible ABE scheme for the same predicate class and IBE scheme with ciphertext size $\max(S, |m|) + \mathrm{poly}(\lambda)$. For our Theorem 1.1 instantiation, the resulting incompressible ABE and IBE scheme achieves the strongest possible security, where even the randomness used during setup can be revealed to the adversary. The size of the incompressible SKE secret key grows with $|S|$, but since the ciphertext size of the RNC-AB-KEM or RNC-IB-KEM is independent of the session-key size, this does not affect the final ciphertext size.

Theorem 1.2. *Assuming the existence of an adaptively secure RNC-IB-KEM whose ciphertext size is independent of the size of the session key, there exists an adaptively secure strongly incompressible IBE scheme with optimal ciphertext size (*$\max(S, |m|) + \mathrm{poly}(\lambda)$*). The analogous statement holds for any adaptively secure RNC-AB-KEM for a predicate class with predicate encodings, yielding an adaptively secure strongly incompressible ABE scheme for the same predicate class with the same ciphertext bound. Moreover, if the underlying non-committing KEM enjoys the stronger guarantee where the setup randomness may be revealed in the response phase, then the resulting incompressible scheme is super-strong. In particular, assuming the hardness of* SXDH *and* LWE *(or* DCR*), we obtain such super-strong schemes.*

RNC-IBE and Incompressible IBE Constructions for a Polynomially Bounded Identity Space: The second RNC-IBE construction supports polynomially many identities with a compact master public key, in the sense that the master public key does not explicitly list one public key for each identity. Unlike the first construction, it can be instantiated under either DDH or LWE. This construction introduces an additional feature where a non-committing ciphertext can be generated together with the master public key, i.e., it does not require knowledge of the target identity id^*. This can be viewed as a stronger simulation property than standard RNC-IBE, although we do not formalize it as a separate notion. This is the first construction to offer such a capability, which may be of independent interest. Similar to the first construction, this scheme remains secure when the randomness used by the setup algorithm is provided to the adversary.

Theorem 1.3. *Assuming the hardness of $\mathcal{X}$ where $\mathcal{X} \in \{\mathsf{DDH}, \mathsf{LWE}\}$, there exists an adaptively secure RNC-IBE that supports polynomially many identities. Moreover, it remains secure even if the setup randomness is revealed to the adversary.*

By combining these results with an incompressible secret key encryption scheme in a hybrid encryption framework, we obtain the first super-strong incompressible IBE schemes under either DDH or LWE that support polynomially many identities. The resulting incompressible IBE scheme also inherits this compact master public key property. Finally, it remains secure even if the adversary receives the randomness used during the setup.

Theorem 1.4. *Assuming the hardness of $\mathcal{X}$ where $\mathcal{X} \in \{\mathsf{DDH}, \mathsf{LWE}\}$, there exists an adaptively secure super-strong incompressible IBE scheme that supports polynomially many identities.*

1.1.1 Constructions from Indistinguishability Obfuscation:

We additionally present an RNC-IBE scheme using indistinguishability obfuscation (iO) and one-way functions. It is important to highlight that while the NC-ABE scheme introduced by Hiroka *et al.* [HMNY21] also utilizes iO, our approach differs substantially. Their construction requires the use of a non-interactive zero-knowledge (NIZK) proof system and achieves *adaptive* security, whereas ours depends only on simpler primitives, such as selectively secure IBE, but achieves only *selective* security.

Theorem 1.5. *Assuming the existence of iO and one-way functions, there exists a selectively secure RNC-IBE scheme.*

In this construction, the ciphertext rate is poor because the size of the ciphertext depends on both the length of the identity and $|m| \cdot \mathrm{poly}(\lambda)$. However, by employing Dziembowski's incompressible SKE scheme (with a secret key size of $\mathrm{poly}(\lambda)$), we can still obtain a rate-$\frac{1}{2}$ selectively secure strongly incompressible IBE scheme (see Theorem 3.8).

Theorem 1.6. *Assuming the existence of iO and one-way functions, there exists a rate-$\frac{1}{2}$ selectively secure strongly incompressible IBE scheme.*

1.2 Related Works

In the field of incompressible encryption, Dziembowski [Dzi06] introduced the first constructions for incompressible symmetric key encryption (SKE). He presented an information-theoretic scheme with a rate of $\frac{1}{3}$, as well as a construction achieving a rate of $\frac{1}{2}$ based on one-way functions. More than a decade later, Guan *et al.* [GWZ22] introduced two incompressible public key encryption (PKE) schemes – the first, although based on standard PKE, had poor compression efficiency, while the second employed indistinguishability obfuscators [GGH+13a, SW14, GGH+16] to realize a rate-1 scheme. Branco *et al.* [BDD22] followed up by designing a rate-1 incompressible PKE scheme that offers chosen ciphertext attack (CCA) security, combining a rate-1 incompressible SKE with programmable hash proof systems. More recently, Guan *et al.* [GWZ23] advanced the notion by developing multi-user incompressible encryption. Here, the adversary is given multiple ciphertexts encrypted with different secret keys.

Goyal *et al.* [GKRV25] extended the concept to functional and attribute-based encryption (ABE), introducing a variety of incompressible security notions for functional encryption, attribute-based encryption, and identity-based encryption. Their work also provided constructions for incompressible functional encryption that achieve an optimal trade-off between ciphertext size and secret key size.

In another direction, Bhushan *et al.* [BGK+25] explored incompressible encryption in the context of leakage resilience. They presented a range of leakage-resilient incompressible encryption schemes tailored to various leakage functions. Their work also examined the challenges of constructing rate-1 schemes with short ciphertexts or schemes that can withstand significant leakage.

In addition to encryption, Guan *et al.* [GWZ22] also proposed incompressible signature schemes, which guarantee that an adversary cannot forge or reconstruct a signature from a compressed version. A related area is incompressible encodings [DGO19, GLW20, MW20], where it is computationally hard to reconstruct a codeword from a compressed version, even with access to the original message. Prior research [DGO19, GLW20, MW20] has shown positive results for incompressible encodings within the random oracle and common reference string (CRS) models.

The area of non-committing encryption (NCE) has also seen significant work focusing on building NCE schemes where the adversary gains access to the randomness used during the setup and encryption phases. These works developed schemes under various assumptions with the goal of achieving high ciphertext-rate [CFGN96, Bea97, DN00, CDSMW09, HOR15, HORR15, CPR17, YKT19, YKXT20, BBD+20].

Another direction explores optimizing parameters for weaker forms of NCE, where the adversary is restricted to gaining access to the randomness used in either the setup (receiver) or encryption (sender). Jarecki and Lysyanskaya [JL00] introduced a scheme that is non-committing only for the receiver, whereas Canetti, Halevi and Katz [CHK05] constructed a constant-rate NCE with erasures, where the adversary only receives the secret key and the ciphertext. Hiroka *et al.* [HMNY21] introduced non-committing attribute-based encryption (NC-ABE) using indistinguishable obfuscators, focusing on achieving ABE with certified deletion in quantum settings, where the adversary receives the master secret key along with the ciphertext. Bhushan *et al.* [BGK+25] introduced leakage-resilient non-commitment KEM schemes and used them to construct leakage-resilient incompressible PKE schemes.

A closely related primitive to non-committing encryption is deniable encryption (DE) [CDNO97]. In a (receiver) deniable encryption scheme, the encryption scheme includes an additional algorithm, Fake, which takes as input the secret key sk, a ciphertext ct (which is an encryption of message m'), and another message m. The algorithm then outputs a fake secret key sk' such that an adversary, when given either $(\mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m), m, m')$ or $(\mathsf{pk}, \mathsf{sk}', \mathsf{ct}, m, m')$, cannot distinguish between the two cases.

In other words, the Fake algorithm provides a mechanism to generate a fake secret key that decrypts a **properly generated** ciphertext to the required message. Observe that the key difference between NCE and DE is that in the non-committing setting, the simulator generates a fake ciphertext and produces a fake secret key, whereas in the deniable setting, the ciphertext is faithfully generated by the encryption algorithm, and only later is a fake secret key produced. Thus, DE can be viewed as a stronger primitive than NCE.

Similar to NCE, there have been numerous works focused on DE [CDNO97, OPW11, BNNO11, SW14, AFL16, DCIO16, AGM21, CGV22].

2 Technical Overview

RNC-ABE/IBE from Bilinear Groups

Dual system encryption is a versatile framework employed in the construction of numerous IBE [Wat09, LW10, CW13, HKS15, GGH20] and ABE [LW11, OT12, Att14, Wee14, CGW15, Att16, CGKW18, GWW19, FGC21] schemes. In our construction, we utilize a pairing group $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are all cyclic groups of prime order p, generated respectively by g_1, g_2, and $e(g_1, g_2)$, where e is a non-degenerate bilinear map, that is, for all $a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$. We use bracket notations, where for all exponents $a \in \mathbb{Z}_p$ and all groups $s \in \{1, 2, T\}$, we denote by $[a]_s$ the group element g_s^a. This notation extends to vectors and matrices as well.

We describe the core RNC-IB-KEM construction, which produces a session key in $\mathbb{G}_T$; by the hybrid-encryption transformation from Theorem 4.5, this yields an RNC-IBE scheme. For exposition, we focus on a single session-key component; the actual construction repeats this in parallel and hashes the resulting group elements to obtain a bit-string session key. By using predicate encoding, the same ideas lift to the ABE setting; we defer the details to Section 5. In a dual system encryption scheme, there are two categories of keys and ciphertexts: normal and semi-functional. Normal keys and ciphertexts operate within the real system, whereas semi-functional objects are progressively incorporated into the hybrid security proof. In our proof, the admissible secret-key queries are converted to semi-functional form one by one, while the challenge ciphertext is changed to a semi-functional ciphertext. We then rewrite each master-secret vector so that one coordinate is already fixed by the master public key and the orthogonal coordinate can be chosen later. This lets the simulator determine the public transcript early and complete the master secret key only in the final response phase so that the challenge ciphertext decapsulates to the desired session key.

The setup algorithm of our construction generates $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$ and sets the public parameters

$$\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$$

.

It then generates a secret vector $\mathbf{k} \leftarrow \mathbb{Z}_p^2$, setting the master public key as $\mathsf{mpk} := [\mathbf{a}^\top \mathbf{k}]_T$ and the master secret key as $\mathsf{msk} := \mathbf{k}$.

To generate a secret key for a specific identity $\mathsf{id} \in \mathbb{Z}_p$, the algorithm samples a random element $s \leftarrow \mathbb{Z}_p$ and outputs $\mathsf{sk}_{\mathsf{id}} := ([s\mathbf{b}]_2, [\mathbf{k} + s(\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top \mathbf{b}]_2)$. To generate a session key and its corresponding ciphertext for a target identity id^*, the algorithm generates a random element $r \leftarrow \mathbb{Z}_p$ and produces the ciphertext $\mathsf{ct} := ([r\mathbf{a}]_1, [r(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{a}]_1)$ and the session key $\mathsf{seskey} := [r\mathbf{a}^\top \mathbf{k}]_T$. For decapsulation, given the secret key $\mathsf{sk}_{\mathsf{id}} = ([\mathbf{d}]_2, [\mathbf{d}']_2)$ and a ciphertext $\mathsf{ct} = ([\mathbf{c}]_1, [\mathbf{c}']_1)$, the algorithm outputs $e([\mathbf{c}]_1^\top, [\mathbf{d}']_2) / e([\mathbf{c}']_1^\top, [\mathbf{d}]_2)$.

We will demonstrate that the experiment can be indistinguishably changed into a non-committing experiment where the session key seskey^* is set to $[x]_T$ for randomly chosen $x \leftarrow \mathbb{Z}_p$ and the master secret key is computed so that it in fact maps the challenge KEM ciphertext ct^* to $\mathsf{seskey}^* = [x]_T$. For simplicity, we assume that there is only one admissible key query across the entire game, for identity id, and that the challenge identity is id^*. In the actual proof, all admissible queries made before the final response are indexed globally and handled in that order. We begin with the following:

$$\mathsf{sk}_{\mathsf{id}} := ([s\mathbf{b}]_2, [\mathbf{k} + s(\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top \mathbf{b}]_2)$$

$$\mathsf{ct}^* := ([r^*\mathbf{a}]_1, [r^*(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{a}]_1) \text{ and } \mathsf{seskey}^* := [r^*\mathbf{a}^\top \mathbf{k}]_T$$

Using the DDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$, we can replace $r^*\mathbf{a}$ and $s\mathbf{b}$ with truly random vectors $\mathbf{u}, \mathbf{v} \leftarrow \mathbb{Z}_p^2$.

$$\mathsf{sk}_{\mathsf{id}} := ([\mathbf{v}]_2, [\mathbf{k} + (\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top \mathbf{v}]_2)$$

$$\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1) \text{ and } \mathsf{seskey}^* := [\mathbf{u}^\top \mathbf{k}]_T$$

Next, we change the sampling method by sampling $\mathbf{W}_1 := \hat{\mathbf{W}}_1 + w_1 \mathbf{W}_0$ and $\mathbf{W}_2 := \hat{\mathbf{W}}_2 + w_2 \mathbf{W}_0$ where $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$ and $\mathbf{W}_0 := \mathbf{b}^\perp (\mathbf{a}^\perp)^\top / (\mathbf{b}^\perp)^\top \mathbf{a}^\perp$. This maintains the same distribution, but now the secret key and ciphertext involve w_1, w_2 as follows.

$$\mathsf{sk}_{\mathsf{id}} := ([\mathbf{v}]_2, [\mathbf{k} + (\hat{\mathbf{W}}_1 + \mathsf{id} \cdot \hat{\mathbf{W}}_2)^\top \mathbf{v} + t(w_1 + \mathsf{id} w_2)\mathbf{a}^\perp]_2)$$

$$\mathsf{ct}^* := ([\mathbf{u}]_1, [(\hat{\mathbf{W}}_1 + \mathsf{id}^* \cdot \hat{\mathbf{W}}_2)\mathbf{u} + r'(w_1 + \mathsf{id}^* w_2)\mathbf{b}^\perp]_1) \text{ and } \mathsf{seskey}^* := [\mathbf{u}^\top \mathbf{k}]_T$$

where $\mathbf{u} = r\mathbf{a} + r'\mathbf{b}^\perp$ and $\mathbf{v} = s'\mathbf{b} + t\mathbf{a}^\perp$ for some $r, r', s', t \in \mathbb{Z}_p$. Since $\mathsf{id} \neq \mathsf{id}^*$, the following holds.

$$\{ w_1 + \mathsf{id}^* w_2, w_1 + \mathsf{id} w_2 \} \equiv \{ w_1 + \mathsf{id}^* w_2, w \}$$

where w is chosen uniformly at random. Therefore, we can change to

$$\mathsf{sk}_{\mathsf{id}} := ([\mathbf{v}]_2, [\mathbf{k} + (\hat{\mathbf{W}}_1 + \mathsf{id} \cdot \hat{\mathbf{W}}_2)^\top \mathbf{v} + w\mathbf{a}^\perp]_2)$$

$$\mathsf{ct}^* := ([\mathbf{u}]_1, [(\hat{\mathbf{W}}_1 + \mathsf{id}^* \cdot \hat{\mathbf{W}}_2)\mathbf{u} + r'(w_1 + \mathsf{id}^* w_2)\mathbf{b}^\perp]_1) \text{ and } \mathsf{seskey}^* := [\mathbf{u}^\top \mathbf{k}]_T$$

In the actual proof, multiple secret keys are involved, and this modification cannot be made if all the secret keys contain information about w_1 and w_2. However, by introducing additional hybrid steps and ensuring that at any moment only one secret key retains information about w_1 and w_2, we will carefully modify the secret keys one by one.

Now, we revert back to the original $\mathbf{W}_1, \mathbf{W}_2$ and use the DDH assumption to reach

$$\mathsf{sk}_{\mathsf{id}} := ([s\mathbf{b}]_2, [\mathbf{k} + s(\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top \mathbf{b} + w\mathbf{a}^\perp]_2)$$

$$\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1) \text{ and } \mathsf{seskey}^* := [\mathbf{u}^\top \mathbf{k}]_T$$

We now sample $k_1, k_2 \leftarrow \mathbb{Z}_p$ and set $\mathbf{k} := \frac{k_1}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_2}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp$. This modification results in $\mathsf{mpk} = [k_1]_T$. Note that this is merely a conceptual change. We now modify the secret key as follows:

$$\mathsf{sk}_{\mathsf{id}} := ([s\mathbf{b}]_2, [\frac{k_1}{|\mathbf{a}|^2} \cdot \mathbf{a} + s(\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top \mathbf{b} + w\mathbf{a}^\perp]_2)$$

$$\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1) \text{ and } \mathsf{seskey}^* := [\mathbf{u}^\top \mathbf{k}]_T$$

This change is indistinguishable because w is chosen uniformly at random. We can express $\mathbf{u} = u_1\mathbf{a} + u_2\mathbf{a}^\perp$ where $u_1, u_2 \leftarrow \mathbb{Z}_p$; except with negligible probability, we have $u_2 \neq 0$. Now, given a uniformly random $x \leftarrow \mathbb{Z}_p$, we can program the remaining coordinate of the master secret key as $k_2 = \dfrac{x - u_1 k_1}{u_2}$. Crucially, this choice need not be made immediately: the simulator can store (u_1, u_2, x) in its state and define k_2, and hence msk, only in the final response phase. For verification, let us check that generating a secret key for id^* and decrypting ct^* would yield $[x]_T$. A secret key for id^* would be

$$\mathsf{sk}_{\mathsf{id}^*} := ([s^*\mathbf{b}]_2, [\mathbf{k} + s^*(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)^\top \mathbf{b}]_2)$$

and decryption would result in

$$\begin{aligned}
\frac{e([\mathbf{u}^\top]_1, [\mathbf{k} + s^*(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)^\top \mathbf{b}]_2)}{e([((\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u})^\top]_1, [s^*\mathbf{b}]_2)} &= \frac{e([\mathbf{u}^\top]_1, [\mathbf{k}]_2) \cdot e([\mathbf{u}^\top]_1, [s^*(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)^\top \mathbf{b}]_2)}{e([((\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u})^\top]_1, [s^*\mathbf{b}]_2)} \\
&= e([\mathbf{u}^\top]_1, [\mathbf{k}]_2) = [\mathbf{u}^\top \mathbf{k}]_T \\
&= \left[(u_1\mathbf{a} + u_2\mathbf{a}^\perp)^\top \left(\frac{k_1}{|\mathbf{a}|^2}\mathbf{a} + \frac{k_2}{|\mathbf{a}^\perp|^2}\mathbf{a}^\perp \right) \right]_T \\
&= [u_1 k_1 + u_2 k_2]_T = [x]_T
\end{aligned}$$

The actual construction carries this out component-wise for all session-key coordinates and then applies the same response-phase programming idea in the ABE lift through predicate encodings. For the complete construction and proof, refer to Section B.

RNC-IBE from Batch Encryption

Let us start by reviewing batch encryption in the binary special case used here. The setup algorithm takes a binary secret key $\mathsf{sk} \in \{0,1\}^n$ and outputs a shorter public key pk. Given a matrix $M \in \mathcal{M}^{n \times 2}$, the encryption algorithm produces a ciphertext ct from which decryption with sk recovers exactly the entries indexed by sk, namely $m_i = M_{i,\mathsf{sk}[i]}$ for all $i \in [n]$.

The RNC-IBE scheme uses batch encryption and a garbling scheme as follows. Let $d = \log(\mathrm{poly}(\lambda))$ be an integer and $T = 2^d$ be a polynomial in the security parameter that denotes the number of identities the scheme supports, and let $\mathsf{int} : \{0,1\}^d \to [T]$ denote the lexicographic rank function. The setup algorithm generates T pairs of NCE public and secret keys, denoted as $\{ \mathsf{nce.pk}_{\mathsf{id}}, \mathsf{nce.sk}_{\mathsf{id}} \}_{\mathsf{id} \in \{0,1\}^d}$, where each pair corresponds to a different identity $\mathsf{id} \in \{0,1\}^d$. These keys together form the master secret key of the RNC-IBE scheme. The master public key is $\mathsf{mpk} := (\mathsf{be.pp}, \mathsf{be.pk})$, where $\mathsf{be.pp} \leftarrow \mathsf{BE.Params}(1^\lambda, 1^{nT})$ and $\mathsf{be.pk} \leftarrow \mathsf{BE.Setup}(\mathsf{be.pp}, \{ \mathsf{nce.pk}_{\mathsf{id}} \}_{\mathsf{id} \in \{0,1\}^d})$. In other words, the batch-encryption secret key is the concatenation of the NCE public keys.

The secret key for an identity $\mathsf{id} \in \{0,1\}^d$ consists of all the NCE public keys $\{ \mathsf{nce.pk}_k \}_{k \in \{0,1\}^d}$ together with the corresponding secret key $\mathsf{nce.sk}_{\mathsf{id}}$. To encrypt a message m for identity id, the encryption algorithm garbles, for each identity $k \in \{0,1\}^d$, a circuit that takes an NCE public key as input and outputs an NCE ciphertext under that public key. It does so as follows:

- For the target identity id, it generates $\left(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{ \mathsf{lab}_{j,b}^{(\mathsf{id})} \} \right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m))$.
- For every $k \in \{0,1\}^d \setminus \{ \mathsf{id} \}$, it generates $\left(\tilde{\mathcal{C}}^{(k)}, \{ \mathsf{lab}_{j,b}^{(k)} \} \right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^{(k)}))$, where $m^{(k)}$ is chosen uniformly at random.

A matrix $M \in [\{0,1\}^\lambda]^{nT \times 2}$ is then constructed such that $M[(\mathsf{int}(k) - 1) \cdot n + j, b] = \mathsf{lab}_{j,b}^{(k)}$ for all $k \in \{0,1\}^d$, $j \in [n]$, and $b \in \{0,1\}$. The batch encryption scheme is then used to produce $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{be.pk}, M)$. The final ciphertext is $\mathsf{ct} := \left(\{ \tilde{\mathcal{C}}^{(k)} \}_{k \in \{0,1\}^d}, \mathsf{be.ct} \right)$. Decryption uses the embedded public keys to recover exactly the labels

associated with identity id, evaluates $\tilde{\mathcal{C}}^{(\mathsf{id})}$ to obtain an NCE ciphertext under $\mathsf{nce.pk}_{\mathsf{id}}$, and then decrypts it using $\mathsf{nce.sk}_{\mathsf{id}}$.

The simulation works in four stages, mirroring the RNC-IBE definition. First, the simulator runs the first NCE simulator for every identity to obtain a simulated public key $\mathsf{nce.pk}_k$, a dummy ciphertext $\mathsf{nce.ct}_k$, and simulator state. It then constructs the batch-encryption public key from the simulated NCE public keys and uses the garbling simulator to produce $\left(\tilde{\mathcal{C}}^{(k)}, \{\, \mathsf{lab}_j^{(k)} \,\}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, \mathsf{nce.ct}^{(k)})$ for all $k \in \{0,1\}^d$. Encrypting the corresponding labels with the batch-encryption scheme already determines the eventual public challenge ciphertext, so Sim_1 can store that ciphertext in its state even though the target message is not yet known.

Whenever an admissible secret-key query for identity k arrives, Sim_2 opens the k-th dummy NCE instance to a random message and returns the corresponding $\mathsf{nce.sk}_k$, leaving the challenge identity unopened. After Sim_3 receives the challenge identity and releases the precomputed challenge ciphertext, Sim_4 finally opens the target instance to the target message and every remaining unopened instance to a random message. This reconstructs a master secret key consistent with the ciphertext and with all earlier key-query answers. The same completed openings also determine the full setup randomness, yielding the stronger response-phase variant used later. For more details, refer to Section 6.

3 Preliminaries

Let PPT denote probabilistic polynomial time. We denote the set of all positive integers up to n as $[n] := \{\, 1, \ldots, n \,\}$ and $[n]_0 := \{\, 0, 1, \ldots, n \,\}$. In addition, we use $[i,j]$ to denote the set of all non-negative integers between i and j, including i and j, i.e., $[i,j] := \{\, i, i+1, \ldots, j \,\}$. For any two binary strings x, y, we use the notation $x \preceq y$ (or $x \in \mathsf{prefix(y)}$) to imply that x is a prefix of y, and $x \| y$ to denote x concatenated with y. Also, $x[i,j]$ denotes the substring $x[i] \| x[i+1] \| \ldots \| x[j]$ when $i \leq j$, and $x[i,j] = \epsilon$ when $i > j$. Throughout this paper, unless specified, all polynomials we consider are positive polynomials. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Suppose S is an ordered set of n elements, i.e., $S = (a_1, \ldots, a_n)$; then we use the notation $(b_1, \ldots, b_n) \leftarrow S$ to denote that b_i is assigned the value a_i, for all $i \in [n]$. More generally, when the order is determined by indexing, we may write $\{\, a_i \,\}_{i \in [n]}$ for the ordered tuple $(a_1, \ldots, a_n)$.

We use a pairing group $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are all cyclic groups of prime order p, generated respectively by g_1, g_2, and $g_T := e(g_1, g_2)$, where e is a non-degenerate bilinear map, that is, for all $a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

We use bracket notations, where for all exponents $a \in \mathbb{Z}_p$ and all groups $s \in \{1, 2, T\}$, we denote by $[a]_s$ the group element g_s^a. We generalize this notation for vectors and matrices as follows. For any $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$, $[\mathbf{A}]_s := g_s^{\mathbf{A}}$ denotes an $n \times m$ matrix with elements from $\mathbb{G}_s$ such that the (i,j)-th element is $[\mathbf{A}_{i,j}]_s$. Since $[\cdot]_s$ is a linear function, we can obtain $[\mathbf{AB}]_s$ from $\mathbf{A}$ and $[\mathbf{B}]_s$ (or $[\mathbf{A}]_s$ and $\mathbf{B}$) for any matrices $\mathbf{A}, \mathbf{B}$ over $\mathbb{Z}_p$. Also, given $[\mathbf{A}]_1$ and $[\mathbf{B}]_2$, we define $[\mathbf{AB}]_T = e([\mathbf{A}]_1, [\mathbf{B}]_2)$.

Any $\mathbb{Z}_p$-linear function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^m$ can be trivially extended to $f : \mathbb{Z}_p^{\ell \times n} \to \mathbb{Z}_p^{\ell \times m}$ such that the i^{th} row of $f(\mathbf{X})$, where $\mathbf{X} \in \mathbb{Z}_p^{\ell \times n}$, is the evaluation of f on the i^{th} row of $\mathbf{X}$. Since any $\mathbb{Z}_p$-linear function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^m$ can also be represented as $f : \underbrace{\mathbb{Z}_p \times \cdots \times \mathbb{Z}_p}_{n} \to \underbrace{\mathbb{Z}_p \times \cdots \times \mathbb{Z}_p}_{m}$, the extended f can also be represented as $f : \underbrace{\mathbb{Z}_p^\ell \times \cdots \times \mathbb{Z}_p^\ell}_{n} \to \underbrace{\mathbb{Z}_p^\ell \times \cdots \times \mathbb{Z}_p^\ell}_{m}$. Equivalently, for an ordered tuple $\{\, \mathbf{u}_i \,\}_{i \in [n]}$ with $\mathbf{u}_i \in \mathbb{Z}_p^\ell$, we identify $\{\, \mathbf{u}_i \,\}_{i \in [n]}$ with the $\ell \times n$ matrix whose i^{th} column is $\mathbf{u}_i$, and apply f row-wise.

Therefore, for any $\mathbb{Z}_p$-linear function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^m$ and $x \in \mathbb{Z}_p^n$, we will use the notation $f([x]_s)$[2] to denote $[f(x)]_s$. More generally, for an ordered tuple $\{\, \mathbf{u}_i \,\}_{i \in [n]}$, we write $f(\{\, [\mathbf{u}_i]_s \,\}_{i \in [n]})$ for the group encoding of the

[2]Observe that given $[x]_s$, we compute $[f(x)]_s$ because f can be described using a matrix $C \in \mathbb{Z}_p^{m \times n}$. Therefore, $[f(x)]_s = [Cx]_s$ can be computed from $[x]_s$ by using group exponentiation and multiplication operations.

row-wise extension of f applied to that tuple. With this, we have

$$e([\mathbf{a}^\top]_1, f(\{\,[\mathbf{b}_i]_2\,\}_{i\in[n]})) = f(\{\,[\mathbf{a}^\top\mathbf{b}_i]_T\,\}_{i\in[n]})$$
$$e(f(\{\,[\mathbf{b}_i^\top]_2\,\}_{i\in[n]}), [\mathbf{a}]_1) = f(\{\,[\mathbf{b}_i^\top\mathbf{a}]_T\,\}_{i\in[n]})$$

3.1 Predicate Encoding

Let $P : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a predicate and p be a prime. A $\mathbb{Z}_p$-linear predicate encoding [Wee14, CGW15] with respect to P consists of five functions

$$\mathsf{sE} : \mathcal{X} \times \mathbb{Z}_p^n \to \mathbb{Z}_p^{n_s} \qquad \mathsf{rE} : \mathcal{Y} \times \mathbb{Z}_p^n \to \mathbb{Z}_p^{n_r}$$
$$\mathsf{kE} : \mathcal{Y} \times \mathbb{Z}_p \to \mathbb{Z}_p^{n_r}$$
$$\mathsf{sD} : \mathcal{X} \times \mathcal{Y} \times \mathbb{Z}_p^{n_s} \to \mathbb{Z}_p \qquad \mathsf{rD} : \mathcal{X} \times \mathcal{Y} \times \mathbb{Z}_p^{n_r} \to \mathbb{Z}_p$$

for some $n, n_s, n_r \in \mathbb{N}$ with the following properties. Whenever these maps are applied to indexed families such as $\{\,\mathbf{u}_i\,\}_{i\in[n]}$, we use the row-wise tuple extension described above and interpret $\{\,\mathbf{u}_i\,\}_{i\in[n]}$ as an ordered tuple.

1. Linearity: For all $(x,y) \in \mathcal{X} \times \mathcal{Y}$, $\mathsf{sE}(x,\cdot), \mathsf{rE}(y,\cdot), \mathsf{kE}(y,\cdot), \mathsf{sD}(x,y,\cdot), \mathsf{rD}(x,y,\cdot)$ are $\mathbb{Z}_p$-linear.
2. Reconstruction: For all $\mathbf{w} \in \mathbb{Z}_p^n, \alpha \in \mathbb{Z}_p$ and $(x,y) \in \mathcal{X} \times \mathcal{Y}$ such that $P(x,y) = 1$, we have

   $$\mathsf{sD}(x,y,\mathsf{sE}(x,\mathbf{w})) = \mathsf{rD}(x,y,\mathsf{rE}(y,\mathbf{w}))$$

   and

   $$\mathsf{rD}(x,y,\mathsf{kE}(y,\alpha)) = \alpha$$

3. Privacy: For all $\alpha \in \mathbb{Z}_p$ and $(x,y) \in \mathcal{X} \times \mathcal{Y}$ such that $P(x,y) = 0$, the following two distributions are identical

   $$\{x,y,\alpha,\mathsf{sE}(x,\mathbf{w}),\mathsf{kE}(y,\alpha)+\mathsf{rE}(y,\mathbf{w})\}_{\mathbf{w}} \equiv \{x,y,\alpha,\mathsf{sE}(x,\mathbf{w}),\mathsf{rE}(y,\mathbf{w})\}_{\mathbf{w}}$$

Remark 3.1. See Section A for an example of a predicate encoding for the identity predicate, that is, $P(x,y) = 1 \iff x = y$.

3.2 Hardness Assumptions

3.2.1 Decisional Diffie-Hellman:

The Decisional Diffie-Hellman (DDH) assumption with respect to a group generator $\mathcal{G}$ is that for every PPT adversary $\mathcal{A}$ it holds that

$$\Big|\Pr_{\substack{(\mathbb{G},g,p)\leftarrow\mathcal{G}(1^\lambda)\\ a,b\leftarrow\mathbb{Z}_p}}[\mathcal{A}(1^\lambda,(\mathbb{G},g,p),g^a,g^b,g^{a\cdot b})=1] - \Pr_{\substack{(\mathbb{G},g,p)\leftarrow\mathcal{G}(1^\lambda)\\ a,b,u\leftarrow\mathbb{Z}_p}}[\mathcal{A}(1^\lambda,(\mathbb{G},g,p),g^a,g^b,g^u)=1]\Big| = \mathsf{negl}(\lambda)$$

3.2.2 Computational Diffie-Hellman:

The Computational Diffie-Hellman (CDH) assumption with respect to a group generator $\mathcal{G}$ is that for every PPT adversary $\mathcal{A}$ it holds that

$$\Pr_{\substack{(\mathbb{G},g,p)\leftarrow\mathcal{G}(1^\lambda)\\ a,b\leftarrow\mathbb{Z}_p}}[\mathcal{A}(1^\lambda,(\mathbb{G},g,p),g^a,g^b)=g^{a\cdot b}] = \mathsf{negl}(\lambda)$$

3.2.3 Symmetric eXternal Diffie-Hellman:

The Symmetric eXternal Diffie-Hellman (SXDH) assumption holds for a pairing group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, g_T, e) \leftarrow \mathcal{G}(1^\lambda)$ if DDH holds for $\mathbb{G}_1$ and $\mathbb{G}_2$.

3.2.4 Learning with Error:

The Learning with Error assumption holds if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $k = k(\lambda)$, $q = q(\lambda)$, and an error distribution D_σ,

$$|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]| = \mathsf{negl}(\lambda)$$

for all $n \in \mathbb{N}$ such that $\mathbf{A} \leftarrow \mathbb{Z}_q^{k \times n}, \mathbf{s} \leftarrow \mathbb{Z}_q^k, \mathbf{e} \leftarrow D_\sigma^n$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

3.3 Batch Encryption

Let $B = B(\lambda, n)$ be a global parameter, and let $\{ \mathcal{W}_{\lambda,n} \}_{\lambda,n \in \mathbb{N}}$ be a family of entry spaces. A batch encryption (BE) scheme for $\{ \mathcal{W}_{\lambda,n} \}_{\lambda,n \in \mathbb{N}}$ consists of the following four algorithms.

$\mathsf{Params}(1^\lambda, 1^n)$: The algorithm takes as input the security parameter 1^λ and a parameter 1^n and outputs a public parameter pp.

$\mathsf{Setup}(\mathsf{pp}, \mathsf{sk})$: The setup algorithm takes as input a public parameter pp and a secret key $\mathsf{sk} \in [B]^n$ and outputs a public key pk.

$\mathsf{Enc}(\mathsf{pk}, M)$: The encryption algorithm takes as input a public key pk and a matrix $M \in \mathcal{W}_{\lambda,n}^{n \times B}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either $\perp$ or a vector $m \in \mathcal{W}_{\lambda,n}^n$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, $\mathsf{sk} \in [B]^n$, and $M \in \mathcal{W}_{\lambda,n}^{n \times B}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m \ : \ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, 1^n) \\ \mathsf{pk} \leftarrow \mathsf{Setup}(\mathsf{pp}, \mathsf{sk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, M) \end{array}\right] = 1$$

where the probability is over the random bits used in the Params, Setup, and Enc algorithms, and $m[i] = M[i, \mathsf{sk}[i]]$ for all $i \in [n]$.

IND-based Security. Consider the following experiment with an adversary $\mathcal{A}$.

- **Initialization Phase:** The adversary takes 1^λ as input, and sends $1^n, \mathsf{sk} \in [B]^n$ to the challenger. The challenger runs $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, 1^n)$, computes $\mathsf{pk} \leftarrow \mathsf{Setup}(\mathsf{pp}, \mathsf{sk})$, and sends $(\mathsf{pp}, \mathsf{pk})$ to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ outputs two messages $M_0, M_1 \in \mathcal{W}_{\lambda,n}^{n \times B}$ such that $M_0[i, \mathsf{sk}[i]] = M_1[i, \mathsf{sk}[i]]$ for all $i \in [n]$. The challenger randomly chooses $b \in \{0, 1\}$, computes a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b)$, and sends ct^* to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 3.2. *A BE scheme satisfies indistinguishability-based security if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

In this work, we will require an *oblivious* batch encryption that has the following properties.

1. Params outputs pp without using any randomness other than pp itself.
2. Setup is a deterministic algorithm.

In other words, the randomness used in Params and Setup is pp only. We emphasize that the constructions given in [BLSV18] for blind batch encryptions are oblivious.

Theorem 3.3 ([BLSV18]). *If either* CDH *or* LWE *is hard, then there exists a secure oblivious BE scheme whose public key has size polynomial in λ, i.e., $|\mathsf{pk}| = \mathrm{poly}(\lambda)$.*

3.4 Non-Committing Encryption

A non-committing encryption (NCE) scheme consists of the following algorithms.

$\mathsf{Setup}(1^\lambda; r_{\mathsf{Setup}})$: The setup algorithm takes as input the security parameter 1^λ. Using the random coins r_{Setup}, it outputs the public key pk and secret key sk.

$\mathsf{Enc}(\mathsf{pk}, m; r_{\mathsf{Enc}})$: The encryption algorithm takes as input a public key pk and a message m, and using random coins r_{Enc} outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message m or $\perp$.

$\mathsf{Sim}_1(1^\lambda)$: The first simulator takes as input the security parameter 1^λ and outputs a public key pk, a ciphertext ct^* and a state st_1.

$\mathsf{Sim}_2(\mathsf{st}_1, m)$: The second simulator takes as input a state st_1 and a message m and outputs $(r_{\mathsf{Enc}}, r_{\mathsf{Setup}})$.

Correctness. For correctness, we require that there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}(1^\lambda)$, and any message m,

$$\Pr_r[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)] = 1 - \mathsf{negl}(\lambda)$$

where r is sampled uniformly at random.

Non-Committing Security. Consider the following two experiments with an adversary $\mathcal{A}$.

Real World:

- **Initialization Phase:** The challenger computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda; r_{\mathsf{Setup}})$ and sends pk to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends m^* to the challenger. The challenger computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m^*; r_{\mathsf{Enc}})$ and returns $(\mathsf{ct}^*, r_{\mathsf{Enc}}, r_{\mathsf{Setup}})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs d.

Simulated World:

- **Initialization Phase:** The challenger computes $(\mathsf{pk}, \mathsf{ct}^*, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(1^\lambda)$ and sends pk to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends m^* to the challenger. The challenger computes $(r_{\mathsf{Enc}}, r_{\mathsf{Setup}}) \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, m^*)$ and returns $(\mathsf{ct}^*, r_{\mathsf{Enc}}, r_{\mathsf{Setup}})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs d.

Let p_{real} and p_{sim} be the probabilities with which $\mathcal{A}$ outputs 0 in the real world and simulated world, respectively.

Definition 3.4. *An NCE scheme is said to be secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,*

$$|p_{\text{real}} - p_{\text{sim}}| = \mathsf{negl}(\lambda)$$

Theorem 3.5 ([BBD+20, YKXT20]). *Under standard assumptions, there exists a secure non-committing encryption scheme.*

3.5 Incompressible Secret Key Encryption

An incompressible secret key encryption scheme $\mathsf{IncSKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{\mathcal{M}_\lambda\}_\lambda$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm is a randomized algorithm that takes as input the security parameter 1^λ, a parameter 1^S and outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{sk}, m)$: The encryption algorithm is a randomized algorithm that takes as input a secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \mathcal{M}_\lambda$ or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}, m \in \mathcal{M}_\lambda$ and $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m] = 1$$

where the probability is over the random bits used in the encryption algorithm.

Definition 3.6 (Incompressible SKE Security). *Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.*

- ***Initialization Phase:*** *$\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$.*
- ***Challenge Phase:*** *$\mathcal{A}_1$ outputs two equal-length messages $(m_0, m_1) \in \mathcal{M}_\lambda^2$, along with auxiliary information aux. The challenger randomly chooses $b \in \{0,1\}$. It computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, m_b)$ and sends it to $\mathcal{A}_1$.*
- ***First Response Phase:*** *$\mathcal{A}_1$ computes a state st such that $|\mathsf{st}| \leq S$.*
- ***Second Response Phase:*** *$\mathcal{A}_2$ receives $(\mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.*

An SKE scheme is said to be incompressible secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

The rate of a scheme is defined as the ratio between the size of a message and the size of a ciphertext, i.e., $\frac{|m|}{|\mathsf{ct}|}$. We say a scheme has rate-1 if $\frac{|m|}{|\mathsf{ct}|} = 1 - o(1)$.

Theorem 3.7 ([BDD22]). *If either LWE or DCR is hard, then there exists a rate-1 incompressible SKE whose secret key size is $|\mathsf{sk}| = |m|(1 + o(1)) + \mathrm{poly}(\lambda)$, where m is the message.*

Theorem 3.8 ([Dzi06]). *If one-way functions exist, then there exists a rate-$\frac{1}{2}$ incompressible SKE whose secret key size is $|\mathsf{sk}| = \mathrm{poly}(\lambda)$.*

3.6 Indistinguishable Obfuscator

An indistinguishable obfuscator (IO) for circuits is a probabilistic algorithm iO that takes as input the security parameter λ and a polynomial-size circuit $C : \{0,1\}^n \to \{0,1\}^m$ and outputs another circuit $\tilde{\mathcal{C}} : \{0,1\}^n \to \{0,1\}^m$ such that

Correctness. For all $x \in \{0,1\}^n$, $C(x) = \tilde{\mathcal{C}}(x)$.

Compactness. $|\tilde{\mathcal{C}}| \leq |C| \cdot \mathrm{poly}(\lambda)$.

Security. For all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathcal{A}(\tilde{\mathcal{C}}) = b : \begin{array}{c}(C_0, C_1) \leftarrow \mathcal{A}(1^\lambda) \\ \tilde{\mathcal{C}} \leftarrow i\mathcal{O}(1^\lambda, C_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where C_0 and C_1 are equivalent circuits, i.e., for all $x \in \{0,1\}^n$, $C_0(x) = C_1(x)$.

3.7 Programmable Pseudorandom Functions

A programmable pseudorandom function $\mathsf{PPRF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Program}, \mathsf{PEval})$ with key space $\{\mathcal{K}_\lambda\}_\lambda$, input space $\{\mathcal{X}_\lambda\}_\lambda$, and output space $\{\mathcal{Y}_\lambda\}_\lambda$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda)$: The setup algorithm is a randomized algorithm that takes as input the security parameter 1^λ and outputs a master secret key $\mathsf{msk} \in \mathcal{K}_\lambda$.

$\mathsf{Eval}(\mathsf{msk}, x)$: The evaluation algorithm is a deterministic algorithm that takes as input a master secret key $\mathsf{msk} \in \mathcal{K}_\lambda$ and $x \in \mathcal{X}_\lambda$ and outputs $y \in \mathcal{Y}_\lambda$.

$\mathsf{Program}(\mathsf{msk}, x, y)$: The program algorithm takes as input the master secret key msk, an input $x \in \mathcal{X}_\lambda$ and an output $y \in \mathcal{Y}_\lambda$. It outputs a secret key $k \in \mathcal{K}_\lambda$.

$\mathsf{PEval}(k, x)$: The programmed evaluation algorithm is a deterministic algorithm that takes as input a secret key $k \in \mathcal{K}_\lambda$ and $x \in \mathcal{X}_\lambda$ and outputs $y \in \mathcal{Y}_\lambda$.

Correctness. A programmable PRF is correct if for all $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, all inputs $x \in \mathcal{X}_\lambda$, and $k \leftarrow \mathsf{Program}(\mathsf{msk}, x^*, y^*)$, we have

$$\mathsf{PEval}(k, x) = \begin{cases} y^* & \text{if } x = x^* \\ \mathsf{Eval}(\mathsf{msk}, x) & \text{otherwise} \end{cases}$$

.

Definition 3.9 (Privacy). *A programmable PRF is private if for all efficient $\mathcal{A}$, the quantity $\mathsf{Adv}^{ppriv}[\mathcal{A}] := \left|\Pr\left[\mathsf{Expt}_0^{\mathsf{ppriv}}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Expt}_1^{\mathsf{ppriv}}(\mathcal{A}) = 1\right]\right|$ is negligible, where $\mathsf{Expt}_b^{\mathsf{ppriv}}(\mathcal{A})$ is given below:*

$\mathsf{Expt}_b^{\mathsf{ppriv}}$ is defined as an experiment between a challenger and an adversary $\mathcal{A}$ for $\lambda \in \mathbb{N}$, where $\mathcal{A}$ can make evaluation and challenge queries.

- The challenger obtains $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ and samples $y^* \leftarrow \mathcal{Y}_\lambda$ uniformly at random.
- The challenger responds to each oracle query type made by $\mathcal{A}$ in the following manner:
  - **Evaluation Oracle:** On input $x \in \mathcal{X}_\lambda$, the challenger returns $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$.
  - **Challenge Oracle:** For a pair of inputs $x_0, x_1 \in \mathcal{X}_\lambda$, the challenger returns $k \leftarrow \mathsf{Program}(\mathsf{msk}, x_b, y^*)$. Note that $\mathcal{A}$ can make only one query to the challenge oracle and does not query the evaluation oracle on these points.
- $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is also output by $\mathsf{Expt}_b^{\mathsf{ppriv}}$.

Theorem 3.10 ([PS18]). *There exists a privately programmable PRF from the LWE assumption.*

The following theorem states that a privately programmable PRF can be built from IO and a puncturable PRF. A puncturable PRF is a variant of a PRF that can be constructed by modifying the GGM PRF [GGM86].

Theorem 3.11 ([BLW17]). *Assuming the existence of iO and one-way functions, there exist secure privately programmable PRFs.*

3.8 One-Time Signature Encryption

A one-time signature encryption (OTSE) scheme consists of the following algorithms.

$\mathsf{Gen}(1^\lambda, 1^\ell)$: The generation algorithm takes as input the security parameter 1^λ and the message length 1^ℓ, and outputs public parameters pp.

$\mathsf{Setup}(\mathsf{pp})$: The setup algorithm takes as input public parameters pp and outputs a pair of verification and signing keys $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, x)$: The signing algorithm takes as input a signing key sk and an input x and outputs a signature σ.

$\mathsf{Enc}(\mathsf{pp}, \mathsf{vk}, i, b, m)$: The encryption algorithm takes as input a verification key vk, an index $i \in [\ell]$, a bit $b \in \{0,1\}$ and a plaintext m. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{vk}, x, \sigma, \mathsf{ct})$: The decryption algorithm takes as input a verification key vk, an input x, a signature σ, and a ciphertext ct. It either outputs a plaintext m or $\perp$.

Definition 3.12 (Correctness). *For correctness, we require that there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda, \ell \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$, *every input* x, *every message* m, *and every index* $i \in [\ell]$,

$$\Pr_{r_1, r_2}\left[\mathsf{Dec}(\mathsf{vk}, x, \sigma, \mathsf{ct}) = m \mid \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{vk}, i, x_i, m; r_1), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x; r_2)\right] \geq 1 - \mathsf{negl}(\lambda)$$

Definition 3.13 (Security). *Consider the following experiment with a* PPT *adversary* $\mathcal{A}$.

- ***Initialization Phase:*** *The challenger generates* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$ *and sends it to the adversary* $\mathcal{A}$. $\mathcal{A}$ *sends back an* x *to the challenger. The challenger generates* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$ *and computes* $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, x)$. *It sends* (vk, σ) *to* $\mathcal{A}$.
- ***Challenge Phase:*** $\mathcal{A}$ *sends* (i, m_0, m_1) *where* $i \in [\ell]$. *The challenger chooses uniformly at random* $b \in \{0,1\}$ *and computes* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{vk}, i, 1 - x_i, m_b)$ *and sends it to* $\mathcal{A}$.
- ***Response Phase:*** $\mathcal{A}$ *outputs* $b' \in \{0,1\}$ *and wins if* $b = b'$.

An OTSE is selectively secure if for all PPT *adversaries* $\mathcal{A}$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $\Pr[\mathcal{A}$ *wins in the above game*$] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Theorem 3.14 ([DG17a]). *Assuming the existence of a selectively secure identity-based encryption scheme, there exists a secure one-time signature encryption scheme.*

3.9 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme IBE for a family of identity spaces $\{\mathcal{ID}_n\}_{n\in\mathbb{N}}$ and message spaces $\mathcal{M}$ consists of four polynomial-time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and the identity length n. It outputs the public parameters mpk and the master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm takes as input the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_n$. It outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the public parameters mpk, a message $m \in \mathcal{M}$, and an identity $\mathsf{id} \in \mathcal{ID}_n$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to m/\perp$. The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext ct. It outputs either a message $m \in \mathcal{M}$ or $\perp$.

Correctness. We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies correctness if for all $\lambda, n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $\mathsf{id} \in \mathcal{ID}_n$, $m \in \mathcal{M}$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = m$.

Definition 3.15. *We say an IBE scheme* $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is adaptively secure if for any stateful PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$, *such that for all* $\lambda, n \in \mathbb{N}$, *the following holds*

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all identities id *queried by* $\mathcal{A}$ *satisfy* $\mathsf{id} \neq \mathsf{id}^*$.

Definition 3.16. *We say an IBE scheme* $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is selectively secure if for any stateful PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that for all* $\lambda, n \in \mathbb{N}$*, the following holds*

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} \mathsf{id}^* \leftarrow \mathcal{A}^{1^\lambda} \\ (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \quad b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all identities id *queried by* $\mathcal{A}$ *satisfy* $\mathsf{id} \neq \mathsf{id}^*$.

Theorem 3.17 ([GGH+13b]). *Assuming the existence of* iO*, there exists an adaptively secure IBE scheme.*

3.10 Incompressible IBE

In this section, we define the strong version of the incompressible security game for an IBE scheme[3] where Setup takes an additional input 1^S and the second adversary obtains the *master secret key*. The game is played against two adversaries $\mathcal{A}_1, \mathcal{A}_2$. The first adversary $\mathcal{A}_1$ receives the complete challenge ciphertext and outputs a compressed state st. Depending on the notion, the second adversary $\mathcal{A}_2$ later receives st together with auxiliary information and either certain secret keys (in weaker variants), the master public key and the master secret key (in the strong notion below), or the setup randomness r_{Setup} (in the super-strong notion defined later). When Setup is randomized, we denote its random coins by r_{Setup}.

Definition 3.18. *(Incompressible IBE Security). Let* $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IBE scheme in which the setup algorithm takes an additional parameter* 1^S *as input. Consider the following experiment with an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Initialization Phase: $\mathcal{A}_1$ *on input* 1^λ*, outputs an upper bound on the state size* 1^S*. The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ *and sends* mpk *to* $\mathcal{A}_1$.

Pre-Challenge Query Phase: *In this phase,* $\mathcal{A}_1$ *is allowed to make polynomially many key queries. For each query* id *sent to the challenger, the challenger computes* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *and returns* $\mathsf{sk}_{\mathsf{id}}$ *to* $\mathcal{A}_1$.

Challenge Phase: $\mathcal{A}_1$ *outputs two equal-length messages* m_0, m_1*, an identity* id^* *along with auxiliary information* aux*. If there exists a query for* id^* *made by* $\mathcal{A}_1$*, the challenger aborts the game. Else, it randomly chooses* $b \in \{0,1\}$ *and computes a ciphertext* $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b)$ *and sends it to* $\mathcal{A}_1$.

Post-Challenge Query Phase: *This is similar to the pre-challenge query phase. The adversary* $\mathcal{A}_1$ *is allowed to send polynomially many key queries. For each query* id*, if* $\mathsf{id}^* = \mathsf{id}$*, the challenger sends* $\perp$*. Else, computes* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *and returns* $\mathsf{sk}_{\mathsf{id}}$ *to* $\mathcal{A}_1$.

First Response Phase: $\mathcal{A}_1$ *computes a state* st *such that* $|st| \leq S$.

Second Response Phase: $\mathcal{A}_2$ *receives* $(\mathsf{mpk}, \mathsf{msk}, aux, st)$*. Finally,* $\mathcal{A}_2$ *outputs* b'*.* $\mathcal{A}$ *wins the experiment if* $b = b'$.

An IBE scheme is said to be ***strongly*** *incompressible secure if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Definition 3.19. *An IBE scheme is said to be* ***super-strong*** *incompressible secure if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

provided the second adversary $\mathcal{A}_2$ *receives the random coins* r_{Setup} *used in the setup algorithm instead of* $\mathsf{mpk}, \mathsf{msk}$.

[3] The only difference between a standard IBE scheme and an incompressible IBE scheme is that the setup algorithm takes an additional parameter 1^S that specifies the compression size.

3.11 Attribute-Based Encryption

An Attribute-Based Encryption (ABE) scheme ABE for a set of attribute spaces $\mathcal{X}$, predicate spaces $\mathcal{F}$, and message spaces $\mathcal{M}$ consists of four polynomial-time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ. It outputs the public parameters mpk and the master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}$. It outputs a secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}$. The encryption algorithm takes as input the public parameters mpk, an attribute $x \in \mathcal{X}$, and a message $m \in \mathcal{M}$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to m/\perp$. The decryption algorithm takes as input a secret key sk_f and a ciphertext ct. It outputs either a message $m \in \mathcal{M}$ or $\perp$.

Correctness. We say an ABE scheme $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies correctness if for all λ, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $f \in \mathcal{F}$, $x \in \mathcal{X}$ such that $f(x) = 1$, $m \in \mathcal{M}$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = m$.

Definition 3.20. *We say an ABE scheme* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is adaptively secure if for any stateful PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that for all* $\lambda \in \mathbb{N}$*, the following holds*

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, x^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all functions f *queried by* $\mathcal{A}$ *satisfy* $f(x^*) = 0$*, and the messages* $m_0, m_1 \in \mathcal{M}$ *have equal length.*

Definition 3.21. *We say an ABE scheme* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is selectively secure if for any stateful PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that for all* $\lambda \in \mathbb{N}$*, the following holds*

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}x^* \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \quad b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all functions f *queried by* $\mathcal{A}$ *satisfy* $f(x^*) = 0$*, and the messages* $m_0, m_1 \in \mathcal{M}$ *have equal length.*

3.12 Incompressible ABE

In this section, we define the strong version of the incompressible security game for an ABE scheme where Setup takes an additional input 1^S. The game is played against two adversaries $\mathcal{A}_1, \mathcal{A}_2$. The first adversary $\mathcal{A}_1$ receives the complete challenge ciphertext and outputs a compressed state st. Depending on the notion, the second adversary $\mathcal{A}_2$ later receives st together with auxiliary information and either certain secret keys (in weaker variants), the master public key and the master secret key (in the strong notion below), or the setup randomness r_{Setup} (in the super-strong notion defined later). When Setup is randomized, we denote its random coins by r_{Setup}.

Definition 3.22. *(Incompressible ABE Security). Let* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an ABE scheme in which the setup algorithm takes an additional parameter* 1^S *as input. Consider the following experiment with an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*.*

Initialization Phase: $\mathcal{A}_1$ *on input* 1^λ*, outputs an upper bound on the state size* 1^S*. The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ *and sends* mpk *to* $\mathcal{A}_1$*.*

Pre-Challenge Query Phase: *In this phase, $\mathcal{A}_1$ is allowed to make polynomially many key queries. For each query f sent to the challenger, the challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}_1$.*

Challenge Phase: *$\mathcal{A}_1$ outputs two equal-length messages m_0, m_1, an attribute x^* along with auxiliary information aux. If there exists a predicate function f queried by $\mathcal{A}_1$ such that $f(x^*) = 1$, the challenger aborts the game. Else, it randomly chooses $b \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{mpk}, x^*, m_b)$ and sends it to $\mathcal{A}_1$.*

Post-Challenge Query Phase: *This is similar to the pre-challenge query phase. The adversary $\mathcal{A}_1$ is allowed to send polynomially many key queries. For each query f, if $f(x^*) = 1$, the challenger sends $\perp$. Else, the challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}_1$.*

First Response Phase: *$\mathcal{A}_1$ computes a state st such that $|st| \leq S$.*

Second Response Phase: *$\mathcal{A}_2$ receives $(\mathsf{mpk}, \mathsf{msk}, aux, st)$. Finally, $\mathcal{A}_2$ outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.*

*An ABE scheme is said to be **strongly** incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Definition 3.23. *An ABE scheme is said to be **super-strong** incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

provided the second adversary $\mathcal{A}_2$ receives the random coins r_{Setup} used in the setup algorithm instead of $\mathsf{mpk}, \mathsf{msk}$.

4 Receiver Non-Committing Attribute- and Identity-Based Primitives: Definitions

In this section, we will present the definitions of a receiver non-committing identity-based encryption (RNC-IBE) scheme and receiver non-committing identity-based key-encapsulation mechanism (RNC-IB-KEM) where the challenger reveals the master secret key msk only at the end of the game, instead of revealing the random coins used in the Setup and Enc (or Encap in the KEM setting) algorithms. In the challenge phase, the challenger returns only the challenge ciphertext (and the session key in the KEM setting). The definitions for receiver non-committing attribute-based encryption (RNC-ABE) and receiver non-committing attribute-based key-encapsulation mechanism (RNC-AB-KEM) follow analogously and are omitted here.

4.1 Receiver Non-Committing Identity-Based Encryption

A receiver non-committing identity-based encryption (RNC-IBE) scheme for a family of identity spaces $\{\mathcal{ID}_n\}_{n\in\mathbb{N}}$ and messages in $\mathcal{M}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm takes as input the security parameter 1^λ and the length of the identities 1^n (in some cases the number of identities 1^T). It outputs the master public key mpk and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: The key generation algorithm takes as input a master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_n$ and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$: The encryption algorithm takes as input a master public key mpk, an identity $\mathsf{id} \in \mathcal{ID}_n$, and a message $m \in \mathcal{M}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message m or $\perp$.

$\mathsf{Sim}_1(1^\lambda, 1^n)$: The first simulator takes as input the security parameter 1^λ and the length of the identities 1^n (in some cases the number of identities 1^T) and outputs a master public key mpk and a state st.

$\mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$: The second simulator takes as input a state st and an identity $\mathsf{id} \in \mathcal{ID}_n$, and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ together with an updated state st.

$\mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$: The third simulator takes as input a state st and an identity $\mathsf{id}^* \in \mathcal{ID}_n$, and outputs a ciphertext ct^* together with an updated state st.

$\mathsf{Sim}_4(\mathsf{st}, m^*)$: The fourth simulator takes as input a state st and a message m^* and outputs a master secret key msk.

Correctness. For correctness, we require that there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, any identity $\mathsf{id} \in \mathcal{ID}_n$, and any message $m \in \mathcal{M}$,

$$\Pr_{r_1, r_2}\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = m \mid \begin{array}{c} \mathsf{ct} = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m; r_1) \\ \mathsf{sk}_{\mathsf{id}} = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}; r_2) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Security. Consider the following two experiments with an adversary $\mathcal{A}$.

Real World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^n. The challenger computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make multiple queries id. For each id, the challenger returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends m^*, id^* to the challenger. If id^* was queried in the pre-challenge query phase, the challenger aborts. Else, it computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$ and returns ct^* to $\mathcal{A}$.
- **Post-Challenge Query Phase:** This is similar to the pre-challenge query phase. The adversary $\mathcal{A}$ is allowed to make multiple queries id. For each id, if $\mathsf{id} = \mathsf{id}^*$, the challenger returns $\perp$. Else, it returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
- **Response Phase:** The challenger sends msk to $\mathcal{A}$. Then, $\mathcal{A}$ outputs d.

Simulated World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^n. The challenger computes $(\mathsf{mpk}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make multiple queries id. For each id, the challenger computes $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$ and returns $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends m^*, id^* to the challenger. If id^* was queried in the pre-challenge query phase, the challenger aborts. Else, it computes $(\mathsf{ct}^*, \mathsf{st}) \leftarrow \mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$ and returns ct^* to $\mathcal{A}$.
- **Post-Challenge Query Phase:** This is similar to the pre-challenge query phase. The adversary $\mathcal{A}$ is allowed to make multiple queries id. For each id, if $\mathsf{id} = \mathsf{id}^*$, the challenger returns $\perp$. Else, it computes $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$ and returns $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Response Phase:** The challenger computes $\mathsf{msk} \leftarrow \mathsf{Sim}_4(\mathsf{st}, m^*)$, sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs d.

It is important to note that the adversary receives the challenge ciphertext in the challenge phase and the master secret key only in the response phase; however, it never obtains the randomness used by the Setup algorithm or the Enc algorithm. Let p_{real} and p_{sim} be the probabilities with which $\mathcal{A}$ outputs 0 in the real world and simulated world, respectively.

Definition 4.1. *An RNC-IBE scheme is said to be **adaptive** secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that, for all* $\lambda \in \mathbb{N}$,

$$|p_{\text{real}} - p_{\text{sim}}| = \mathsf{negl}(\lambda)$$

Definition 4.2. *An RNC-IBE scheme is said to be **selectively** secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that, for all* $\lambda \in \mathbb{N}$,

$$|p_{\text{real}} - p_{\text{sim}}| = \mathsf{negl}(\lambda)$$

provided the adversary commits to the challenge identity id^* *at the beginning of the game and* Sim_1 *additionally takes* id^* *as an input.*

4.2 Receiver Non-Committing Identity-Based Key-Encapsulation Mechanism

A receiver non-committing identity-based key-encapsulation mechanism (RNC-IB-KEM) for a family of identity spaces $\{\mathcal{ID}_n\}_{n\in\mathbb{N}}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm takes as input the security parameter 1^λ and the length of the identities 1^n (in some cases the number of identities 1^T). It outputs the master public key mpk and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: The key generation algorithm takes as input a master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_n$ and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Encap}(\mathsf{mpk}, \mathsf{id})$: The encryption algorithm takes as input a master public key mpk and an identity $\mathsf{id} \in \mathcal{ID}_n$ and outputs a ciphertext ct and a session key seskey, where seskey lies in the seed space of a pseudorandom generator.

$\mathsf{Decap}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a session key seskey or $\perp$.

$\mathsf{Sim}_1(1^\lambda, 1^n)$: The first simulator takes as input the security parameter 1^λ and the length of the identities 1^n (in some cases the number of identities 1^T) and outputs a master public key mpk and a state st.

$\mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$: The second simulator takes as input a state st and an identity $\mathsf{id} \in \mathcal{ID}_n$, and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ together with an updated state st.

$\mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$: The third simulator takes as input a state st, an identity $\mathsf{id}^* \in \mathcal{ID}_n$, and outputs a ciphertext ct^*, a session key seskey, and an updated state st.

$\mathsf{Sim}_4(\mathsf{st}, \mathsf{seskey})$: The fourth simulator takes as input a state st and a session key seskey, and outputs a master secret key msk.

Correctness. For correctness, we require that there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, and any identity $\mathsf{id} \in \mathcal{ID}_n$,

$$\Pr_{r_1, r_2}\left[\mathsf{Decap}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = \mathsf{seskey} \ : \ \begin{array}{c}(\mathsf{ct}, \mathsf{seskey}) = \mathsf{Encap}(\mathsf{mpk}, \mathsf{id}; r_1) \\ \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}; r_2)\end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Security. Consider the following two experiments with an adversary $\mathcal{A}$.

Real World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^n. The challenger computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make multiple queries id. For each id, the challenger returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ to $\mathcal{A}$.

- **Challenge Phase:** The adversary $\mathcal{A}$ sends id^* to the challenger. If id^* was queried in the pre-challenge query phase, the challenger aborts. Else, it computes $(\mathsf{ct}^*, \mathsf{seskey}) \leftarrow \mathsf{Encap}(\mathsf{mpk}, \mathsf{id}^*)$ and returns $(\mathsf{ct}^*, \mathsf{seskey})$ to $\mathcal{A}$.
- **Post-Challenge Query Phase:** This is similar to the pre-challenge query phase. The adversary $\mathcal{A}$ is allowed to make multiple queries id. For each id, if $\mathsf{id} = \mathsf{id}^*$, the challenger returns $\perp$. Else, it returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
- **Response Phase:** The challenger sends msk to $\mathcal{A}$. Then, $\mathcal{A}$ outputs d.

Simulated World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^n. The challenger computes $(\mathsf{mpk}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make multiple queries id. For each id, the challenger computes $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$ and returns $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends id^* to the challenger. If id^* was queried in the pre-challenge query phase, the challenger aborts. Else, it computes $(\mathsf{ct}^*, \mathsf{seskey}, \mathsf{st}) \leftarrow \mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$ and returns $(\mathsf{ct}^*, \mathsf{seskey})$ to $\mathcal{A}$.
- **Post-Challenge Query Phase:** This is similar to the pre-challenge query phase. The adversary $\mathcal{A}$ is allowed to make multiple queries id. For each id, if $\mathsf{id} = \mathsf{id}^*$, the challenger returns $\perp$. Else, it computes $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$ and returns $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Response Phase:** The challenger computes $\mathsf{msk} \leftarrow \mathsf{Sim}_4(\mathsf{st}, \mathsf{seskey})$, sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs d.

It is important to note that the adversary receives the challenge ciphertext and the session key in the challenge phase and the master secret key only in the response phase; however, it never obtains the randomness used by the Setup algorithm or the Encap algorithm to generate them. Let p_{real} and p_{sim} be the probabilities with which $\mathcal{A}$ outputs 0 in the real world and simulated world, respectively.

Definition 4.3. *An RNC-IB-KEM scheme is said to be **adaptive** secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,*

$$|p_{\text{real}} - p_{\text{sim}}| = \mathsf{negl}(\lambda)$$

Definition 4.4. *An RNC-IB-KEM scheme is said to be **selectively** secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,*

$$|p_{\text{real}} - p_{\text{sim}}| = \mathsf{negl}(\lambda)$$

provided the adversary commits to the challenge identity id^* at the beginning of the game and Sim_1 additionally takes id^* as an input.

By combining an RNC-IB-KEM with a pseudorandom generator PRG in a hybrid encryption approach, we can obtain an RNC-IBE scheme. We assume that messages are bit strings and that the session key lies in the seed space of PRG. Specifically, the session key is used as a seed for PRG, and the output of PRG is XORed with the message. This is possible because the security definition for RNC-IB-KEM (and RNC-IBE) requires disclosure of the master secret key, but not of the randomness used by the Setup and/or Encap algorithms. The same transformation also preserves the stronger response-phase guarantee where the simulator reveals setup randomness instead of the master secret key. However, the ciphertext size will be the sum of the RNC-IB-KEM ciphertext size and the message length.

Theorem 4.5. *If there exists an adaptively secure (respectively, selectively secure) RNC-IB-KEM, then there exists an adaptively secure (respectively, selectively secure) RNC-IBE scheme. Moreover, if the underlying RNC-IB-KEM remains secure even when the setup randomness is revealed in the response phase, then the derived RNC-IBE enjoys the analogous stronger guarantee.*

The analogous statement holds immediately in the ABE setting.

Theorem 4.6. *If there exists an adaptively secure (respectively, selectively secure) RNC-AB-KEM for a predicate class, then there exists an adaptively secure (respectively, selectively secure) RNC-ABE scheme for the same predicate class. Moreover, if the underlying RNC-AB-KEM remains secure even when the setup randomness is revealed in the response phase, then the derived RNC-ABE enjoys the analogous stronger guarantee for the same predicate class.*

5 Receiver Non-Committing AB-KEM and ABE from Bilinear Groups

In this section, we present an adaptively secure receiver non-committing attribute-based key encapsulation mechanism (RNC-AB-KEM) using the concepts of dual system encryption [Wat09] and predicate encoding [Wee14, CGW15].

5.1 Construction

Our construction is as follows. Let $\mathsf{HC} : \mathbb{G}_T \times \{0,1\}^{\log(p)} \to \{0,1\}$ denote a 1-bit randomness extractor over a group element, and let $\ell := \ell(\lambda)$ be a polynomial in λ. Let $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ be a $\mathbb{Z}_p$-linear predicate encoding for a predicate class $\mathcal{P}$ with parameters n, n_s, n_r (see Section 3.1). Whenever these maps are applied to indexed families such as $\{ [\mathbf{u}_i]_s \}_{i\in[n]}$, we use the ordered-tuple, row-wise extension from the preliminaries.

$\mathsf{Gen}(1^\lambda)$:

- Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \ldots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.
- Output $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$. We assume that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$ is included in pp and omit it from the notation.

$\mathsf{Setup}(\mathsf{pp})$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$.
- Generate $h \leftarrow \{0,1\}^{\log(p)}$.
- Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2$ for all $i \in [\ell]$.
- Output $\mathsf{mpk} := \left(\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_{i\in[\ell]}, h \right)$ and $\mathsf{msk} := \{ \mathbf{k}_i \}_{i\in[\ell]}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, y \in \mathcal{Y})$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$ and $\mathsf{msk} = \{ \mathbf{k}_i \}_{i\in[\ell]}$.
- Generate $s_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Output $\mathsf{sk}_y := \{ \left([s_i\mathbf{b}]_2, \mathsf{kE}(y, [\mathbf{k}_i]_2) \cdot \mathsf{rE}(y, \{ [s_i\mathbf{W}_j^\top \mathbf{b}]_2 \}_{j\in[n]}) \right) \}_{i\in[\ell]}$. Here $[\mathbf{u}]_2 \cdot [\mathbf{v}]_2$ denotes component-wise group multiplication.

$\mathsf{Encap}(\mathsf{pp}, \mathsf{mpk}, x \in \mathcal{X})$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$ and $\mathsf{mpk} = \left(\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_{i\in[\ell]}, h \right)$.
- Generate $r \leftarrow \mathbb{Z}_p$.
- Output $\mathsf{ct} := \left([r\mathbf{a}]_1, \mathsf{sE}(x, \{ [r\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}) \right)$ and $\mathsf{seskey} := \{ \mathsf{HC}([r\mathbf{a}^\top \mathbf{k}_i]_T, h) \}_{i\in[\ell]}$.

$\mathsf{Decap}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct})$:

- Parse $\mathsf{mpk} = \left(\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_{i\in[\ell]}, h \right)$.
- Parse $\mathsf{sk}_y = \{ ([\mathbf{d}_i]_2, [\mathbf{d}_i']_2) \}_{i\in[\ell]}$ and $\mathsf{ct} = ([\mathbf{c}]_1, [\mathbf{c}']_1)$.
- Output $\mathsf{seskey} := \{ \mathsf{HC}\left(\frac{e([\mathbf{c}]_1^\top, \mathsf{rD}(x,y,[\mathbf{d}_i']_2))}{e(\mathsf{sD}(x,y,[\mathbf{c}']_1)^\top, [\mathbf{d}_i]_2)}, h \right) \}_{i\in[\ell]}$.

$\mathsf{Sim}_1(1^\lambda)$:

- Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \ldots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.
- Generate $k_{i,1} \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$, and generate $h \leftarrow \{0,1\}^{\log(p)}$.
- Output $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$, $\mathsf{mpk} := \left(\{ [k_{i,1}]_T \}_{i\in[\ell]}, h\right)$, and $\mathsf{st}_1 := \left(\{ k_{i,1} \}_{i\in[\ell]}, \mathbf{a}, h\right)$.

$\mathsf{Sim}_2(\mathsf{st}_1, y)$:

- Parse $\mathsf{st}_1 = \left(\{ k_{i,1} \}_{i\in[\ell]}, \mathbf{a}, h\right)$.
- Generate $s_i, w_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Output $\mathsf{sk}_y := \{ \left([s_i\mathbf{b}]_2, \mathsf{kE}(y, [\frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + w_i\mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y, \{ [s_i\mathbf{W}_j^\top \mathbf{b}]_2 \}_{j\in[n]}) \right) \}_{i\in[\ell]}$ and $\mathsf{st}_2 := \mathsf{st}_1$.

$\mathsf{Sim}_3(\mathsf{st}_2, x^*)$:

- Parse $\mathsf{st}_2 = \left(\{ k_{i,1} \}_{i\in[\ell]}, \mathbf{a}, h\right)$.
- Generate $u_1 \leftarrow \mathbb{Z}_p$, $u_2 \leftarrow \mathbb{Z}_p^*$, and $x_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Set $\mathbf{u} := u_1\mathbf{a} + u_2\mathbf{a}^\perp$.
- Output $\mathsf{ct}^* := \left([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i\mathbf{u}]_1 \}_{i\in[n]})\right)$, $\mathsf{seskey} := \{ \mathsf{HC}([x_i]_T, h) \}_{i\in[\ell]}$, and $\mathsf{st}_3 := \left(\mathsf{st}_2, u_1, u_2, \{ x_i \}_{i\in[\ell]}\right)$.

$\mathsf{Sim}_4(\mathsf{st}_3, \mathsf{seskey} \in \{0,1\}^\ell)$:

- Parse $\mathsf{st}_3 = \left(\mathsf{st}_2, u_1, u_2, \{ x_i \}_{i\in[\ell]}\right)$, where $\mathsf{st}_2 = \left(\{ k_{i,1} \}_{i\in[\ell]}, \mathbf{a}, h\right)$.
- Set $k_{i,2} := (x_i - u_1 k_{i,1})/u_2$ for all $i \in [\ell]$.
- Output $\mathsf{msk} := \{ \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp \}_{i\in[\ell]}$.

Whenever a 2-dimensional vector is supplied to kE, sD, or rD, we mean the row-wise two-row extension from the preliminaries. In particular, expressions such as $\mathsf{kE}(y, [\mathbf{k}_i]_2)$, $\mathsf{rD}(x, y, [\mathbf{d}_i']_2)$, and $\mathsf{sD}(x, y, [\mathbf{c}']_1)$ denote encodings of 2-vectors obtained by applying the corresponding linear map to the two coordinates row-wise.

As is standard in predicate-encoding-based ABE, the attribute x associated with a ciphertext is public. Formally, a ciphertext is the pair (x, ct), where ct denotes the group-element portion written above. To keep the notation light, we suppress this metadata and continue to write ct when no confusion can arise.
The input seskey to Sim_4 is included only to match the abstract interface from the preliminaries; in this construction, the simulator state already determines the value needed to reconstruct msk.

Remark 5.1. The randomness used in the setup algorithm consists of h, which is part of the master public key, and the set $\{ \mathbf{k}_i \}_{i\in[\ell]}$, which constitutes the entire master secret key. Since the challenger outputs h in the initialization phase and reveals the master secret key only in the response phase, the adversary eventually obtains all the randomness used in the setup algorithm, but never the randomness used by the Encap algorithm.

Parameters. The size of a ciphertext is $\mathrm{poly}(\lambda)$, i.e., it is independent of ℓ, whereas the sizes of the master public key, master secret key, and user secret keys depend on ℓ.

Correctness. Fix any $(x,y) \in \mathcal{X} \times \mathcal{Y}$ such that $P(x,y) = 1$. Define

$$\begin{aligned}
\mathsf{sk}_y &:= \left\{ \left([s_i \mathbf{b}]_2,\ \mathsf{kE}(y, [\mathbf{k}_i]_2) \cdot \mathsf{rE}(y, \{\, [s_i \mathbf{W}_j^\top \mathbf{b}]_2 \,\}_{j\in[n]}) \right) \right\}_{i\in[\ell]}, \\
\mathsf{ct} &:= \left([r\mathbf{a}]_1,\ \mathsf{sE}(x, \{\, [r\mathbf{W}_j \mathbf{a}]_1 \,\}_{j\in[n]}) \right), \\
\mathsf{seskey} &:= \{\, \mathsf{HC}([r\mathbf{a}^\top \mathbf{k}_i]_T, h) \,\}_{i\in[\ell]}.
\end{aligned}$$

Using the row-wise two-dimensional extensions of kE, sD, and rD throughout, we have

$$\begin{aligned}
&\frac{e\left([r\mathbf{a}]_1^\top, \mathsf{rD}\left(x, y, \mathsf{kE}(y, [\mathbf{k}_i]_2) \cdot \mathsf{rE}\left(y, \{\, [s_i \mathbf{W}_j^\top \mathbf{b}]_2 \,\}_{j\in[n]}\right)\right)\right)}{e\left(\mathsf{sD}\left(x, y, \mathsf{sE}\left(x, \{\, [r\mathbf{W}_j\mathbf{a}]_1 \,\}_{j\in[n]}\right)\right)^\top, [s_i\mathbf{b}]_2\right)} \\
&= \frac{e\left([r\mathbf{a}]_1^\top, \mathsf{rD}(x, y, \mathsf{kE}(y, [\mathbf{k}_i]_2))\right) \cdot e\left([r\mathbf{a}]_1^\top, \mathsf{rD}\left(x, y, \mathsf{rE}\left(y, \{\, [s_i \mathbf{W}_j^\top \mathbf{b}]_2 \,\}_{j\in[n]}\right)\right)\right)}{e\left(\mathsf{sD}\left(x, y, \mathsf{sE}\left(x, \{\, [r\mathbf{W}_j\mathbf{a}]_1 \,\}_{j\in[n]}\right)\right)^\top, [s_i\mathbf{b}]_2\right)} \\
&= \frac{e([r\mathbf{a}]_1^\top, [\mathbf{k}_i]_2) \cdot e\left([r\mathbf{a}]_1^\top, \mathsf{rD}\left(x, y, \mathsf{rE}\left(y, \{\, [s_i \mathbf{W}_j^\top \mathbf{b}]_2 \,\}_{j\in[n]}\right)\right)\right)}{e\left(\mathsf{sD}\left(x, y, \mathsf{sE}\left(x, \{\, [r\mathbf{W}_j\mathbf{a}]_1 \,\}_{j\in[n]}\right)\right)^\top, [s_i\mathbf{b}]_2\right)} \\
&= e([r\mathbf{a}]_1^\top, [\mathbf{k}_i]_2) \cdot \frac{\mathsf{rD}\left(x, y, \mathsf{rE}\left(y, \{\, [r\mathbf{a}^\top s_i \mathbf{W}_j^\top \mathbf{b}]_T \,\}_{j\in[n]}\right)\right)}{\mathsf{sD}\left(x, y, \mathsf{sE}\left(x, \{\, [(r\mathbf{W}_j\mathbf{a})^\top s_i\mathbf{b}]_T \,\}_{j\in[n]}\right)\right)} \\
&= [r\mathbf{a}^\top \mathbf{k}_i]_T.
\end{aligned}$$

The first equality follows from the linearity of $\mathsf{rD}(x, y, \cdot)$ together with component-wise multiplication in the group encoding. The second equality uses the row-wise extension of $\mathsf{rD}(x, y, \mathsf{kE}(y, \alpha)) = \alpha$. The third equality follows from Equation (1). The last equality follows from the reconstruction property $\mathsf{sD}(x, y, \mathsf{sE}(x, \mathbf{w})) = \mathsf{rD}(x, y, \mathsf{rE}(y, \mathbf{w}))$ because $P(x,y) = 1$.

Theorem 5.2. *Assuming the hardness of* SXDH*, the above scheme is an adaptively secure receiver non-committing attribute-based key encapsulation mechanism.*

Proof. Consider an adversary $\mathcal{A}$ against the RNC-AB-KEM security game. Without loss of generality, we may assume that $\mathcal{A}$ makes only admissible key queries, namely queries y satisfying $P(x^*, y) = 0$ for the eventual challenge attribute x^*. Indeed, any pre-challenge query y with $P(x^*, y) = 1$ causes the experiment to abort, while any post-challenge query y with $P(x^*, y) = 1$ is answered with $\perp$. Let q be the total number of admissible key queries made across the pre-challenge and post-challenge query phases, indexed as $y^1, \ldots, y^q$ in the order they are asked.

Hybrid Hyb_0**:** This is the real experiment.

1. The challenger generates $\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}$ as follows.
   - Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
   - Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \ldots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2$ for all $i \in [\ell]$ and $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{\, [\mathbf{W}_i\mathbf{a}]_1 \,\}_{i\in[n]}, \{\, [\mathbf{W}_i^\top \mathbf{b}]_2 \,\}_{i\in[n]})$, $\mathsf{mpk} := \left(\{\, [\mathbf{a}^\top \mathbf{k}_i]_T \,\}_{i\in[\ell]}, h\right)$, and $\mathsf{msk} := \{\, \mathbf{k}_i \,\}_{i\in[\ell]}$.

The challenger sends pp and mpk to $\mathcal{A}$.

2. **Pre-Challenge Query Phase:** $\mathcal{A}$ has access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(y^j)$**:** Given the j-th admissible query $y^j \in \mathcal{Y}$, return sk_{y^j} generated as follows.

   - Generate $s_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.
   - Set $\mathsf{sk}_{y^j} := \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_\alpha^\top \mathbf{b}]_2 \}_{\alpha \in [n]}) \right) \}_{d \in [\ell]}$.

3. **Challenge Phase:** $\mathcal{A}$ outputs $x^* \in \mathcal{X}$. If one of the previously asked queries y^j satisfies $P(x^*, y^j) = 1$, then the challenger aborts. Otherwise, the challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.

   - Generate $r \leftarrow \mathbb{Z}_p$.
   - Set $\mathsf{ct}^* := \left([r\mathbf{a}]_1, \mathsf{sE}(x^*, \{ [r\mathbf{W}_i \mathbf{a}]_1 \}_{i \in [n]}) \right)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([r\mathbf{a}^\top \mathbf{k}_i]_T, h) \}_{i \in [\ell]}$.

   The challenger sends $(\mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.

4. **Post-Challenge Query Phase:** $\mathcal{A}$ continues to have access to $O_{\mathsf{UserKeyGen}}$. For each new query y, the challenger returns $\perp$ if $P(x^*, y) = 1$ and otherwise answers exactly as in the pre-challenge query phase. By our w.l.o.g. assumption, we only need to consider the admissible queries y^j.

5. **Response Phase:** The challenger sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs $d \in \{0, 1\}$.

Using a sequence of hybrid experiments, we transform the real experiment into the simulated experiment. We denote the probability that $\mathcal{A}$ outputs 0 in hybrid Hyb_i by $p_{\mathcal{A},H_i}$.

Below, we assume that $\mathbf{a}$ and $\mathbf{b}^\perp$ are linearly independent and that $\mathbf{a}^\perp$ and $\mathbf{b}$ are linearly independent. This holds with overwhelming probability over the choice of $\mathbf{a}$ and $\mathbf{b}$.

Changing the challenge ciphertext into semi-functional mode.

Hybrid Hyb_1**:** This is the same as Hyb_0 except that $(\mathsf{ct}^*, \mathsf{seskey}^*)$ is generated as $\mathsf{ct}^* := \left([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i \mathbf{u}]_1 \}_{i \in [n]}) \right)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_i]_T, h) \}_{i \in [\ell]}$, where $\mathbf{u} \leftarrow \mathbb{Z}_p^2$.

We have $|p_{\mathcal{A},H_1} - p_{\mathcal{A},H_0}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_1$.

Lemma 5.3. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_1} - p_{\mathcal{A},H_0}| \leq p_{\mathcal{B},\mathsf{DDH}}$.*

We define $\mathsf{Hyb}_{1,0,5}$ to be Hyb_1. For each $i \in [q]$, we now transform $\mathsf{Hyb}_{1,i-1,5}$ into $\mathsf{Hyb}_{1,i,5}$ through the intermediate hybrids $\mathsf{Hyb}_{1,i,1}, \ldots, \mathsf{Hyb}_{1,i,4}$.

Changing the user secret keys into semi-functional mode. We change the user secret keys in both query phases into semi-functional mode one admissible query at a time, using hybrids $\mathsf{Hyb}_{1,i,1}, \ldots, \mathsf{Hyb}_{1,i,5}$ for $i \in [q]$. Throughout this part of the proof, the index j refers to the position of the admissible query in the global order across the pre-challenge and post-challenge query phases. Define $\mathbf{W}_0 := \mathbf{b}^\perp (\mathbf{a}^\perp)^\top / ((\mathbf{b}^\perp)^\top \mathbf{a}^\perp)$.

Hybrid $\mathsf{Hyb}_{1,i,1}$**:** This is the same as $\mathsf{Hyb}_{1,i-1,5}$ except that $O_{\mathsf{UserKeyGen}}$ behaves as follows on the j-th admissible query $y^j \in \mathcal{Y}$.

- If $j < i$, return sk_{y^j} generated as follows.
  - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.
  - Set $\mathsf{sk}_{y^j} := \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_\alpha^\top \mathbf{b}]_2 \}_{\alpha \in [n]}) \right) \}_{d \in [\ell]}$.
- If $j = i$, return sk_{y^i} generated as follows.
  - Generate $\mathbf{v}_d^i \leftarrow \mathbb{Z}_p^2$ for all $d \in [\ell]$.

– Set $\mathsf{sk}_{y^i} := \{ \left([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d]_2) \cdot \mathsf{rE}(y^i, \{ [\mathbf{W}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]}) \right) \}_{d\in[\ell]}$.

- If $j > i$, return sk_{y^j} generated exactly as in Hyb_1.

For every $i \in [q]$, we have $|p_{\mathcal{A},H_{1,i,1}} - p_{\mathcal{A},H_{1,i-1,5}}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$. Since $\mathsf{Hyb}_{1,0,5} = \mathsf{Hyb}_1$, this includes the first transition from Hyb_1 to $\mathsf{Hyb}_{1,1,1}$.

Lemma 5.4. *For all PPT adversaries $\mathcal{A}$ and all $i \in [q]$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_{1,i,1}} - p_{\mathcal{A},H_{1,i-1,5}}| \le \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Hybrid $\mathsf{Hyb}_{1,i,2}$: This is the same as $\mathsf{Hyb}_{1,i,1}$ except that we generate $\hat{\mathbf{W}}_\alpha \leftarrow \mathbb{Z}_p^{2\times 2}$ and $w_\alpha \leftarrow \mathbb{Z}_p$, and set $\mathbf{W}_\alpha := \hat{\mathbf{W}}_\alpha + w_\alpha \mathbf{W}_0$ for all $\alpha \in [n]$.

We have $|p_{\mathcal{A},H_{1,i,2}} - p_{\mathcal{A},H_{1,i,1}}| = 0$ since the distribution of $\{ \mathbf{W}_\alpha \}_{\alpha\in[n]}$ does not change.

Lemma 5.5. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_{1,i,2}} - p_{\mathcal{A},H_{1,i,1}}| = 0$.*

We now show that the values w_α appear only in ct^* and sk_{y^i}. First, $[\mathbf{W}_\alpha \mathbf{a}]_1 = [\hat{\mathbf{W}}_\alpha \mathbf{a}]_1$ and $[\mathbf{W}_\alpha^\top \mathbf{b}]_2 = [\hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2$ for all $\alpha \in [n]$. For ct^*, write $\mathbf{u} = r\mathbf{a} + r'\mathbf{b}^\perp$. Then

$$\begin{aligned}\mathsf{ct}^* &= \left([\mathbf{u}]_1, \mathsf{sE}\left(x^*, \{ [\mathbf{W}_\alpha \mathbf{u}]_1 \}_{\alpha\in[n]}\right)\right) \\ &= \left([\mathbf{u}]_1, \mathsf{sE}\left(x^*, \{ [\hat{\mathbf{W}}_\alpha \mathbf{u}]_1 \}_{\alpha\in[n]}\right) \cdot \mathsf{sE}\left(x^*, \{ [r' w_\alpha \mathbf{b}^\perp]_1 \}_{\alpha\in[n]}\right)\right).\end{aligned}$$

For sk_{y^i}, write $\mathbf{v}_d^i = s_d^i \mathbf{b} + t_d^i \mathbf{a}^\perp$. Then

$$\begin{aligned}\mathsf{sk}_{y^i} &= \{ \left([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d]_2) \cdot \mathsf{rE}\left(y^i, \{ [\mathbf{W}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]}\right)\right) \}_{d\in[\ell]} \\ &= \{ \left([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d]_2) \cdot \mathsf{rE}\left(y^i, \{ [\hat{\mathbf{W}}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]}\right) \cdot \mathsf{rE}\left(y^i, \{ [t_d^i w_\alpha \mathbf{a}^\perp]_2 \}_{\alpha\in[n]}\right)\right) \}_{d\in[\ell]}.\end{aligned}$$

Moreover, for $j \neq i$, we can write

$$\mathsf{sk}_{y^j} = \begin{cases} \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}\left(y^j, \{ [s_d^j \hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2 \}_{\alpha\in[n]}\right)\right) \}_{d\in[\ell]} & \text{if } j < i, \\ \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d]_2) \cdot \mathsf{rE}\left(y^j, \{ [s_d^j \hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2 \}_{\alpha\in[n]}\right)\right) \}_{d\in[\ell]} & \text{if } j > i. \end{cases}$$

Hybrid $\mathsf{Hyb}_{1,i,3}$: This is the same as $\mathsf{Hyb}_{1,i,2}$ except that for the i-th admissible query y^i, $O_{\mathsf{UserKeyGen}}$ returns $\mathsf{sk}_{y^i} := \{ \left([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d + w_d^i \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^i, \{ [\mathbf{W}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]}) \right) \}_{d\in[\ell]}$, where $w_d^i \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

Since the i-th query is admissible, we have $P(x^*, y^i) = 0$. Let $\mathbf{w} := (w_\alpha)_{\alpha\in[n]} \in \mathbb{Z}_p^n$. For the d-th component of the i-th secret key, the additional semi-functional terms in $\mathsf{Hyb}_{1,i,2}$ and $\mathsf{Hyb}_{1,i,3}$ are obtained from $\mathbf{w}$ by the row-wise two-dimensional extensions of the linear maps $\mathbf{w} \mapsto \mathsf{sE}(x^*, r'\mathbf{w})$ and $\mathbf{w} \mapsto \mathsf{rE}(y^i, t_d^i \mathbf{w})$, followed by multiplication with the fixed offsets displayed above. Conditioned on $t_d^i \neq 0$, scalar multiplication by t_d^i is an invertible linear map on $\mathbb{Z}_p^n$, so predicate privacy allows us to add the term $\mathsf{kE}(y^i, [w_d^i \mathbf{a}^\perp]_2)$ without changing the distribution of that component. A routine subhybrid over $d \in [\ell]$ therefore gives the desired transition, and the only bad event is that some t_d^i is zero, which happens with probability at most ℓ/p. Therefore, $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$, where the negligible term also absorbs the overwhelmingly unlikely event that the sampled basis vectors fail to be linearly independent.

Lemma 5.6. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$.*

Hybrid $\mathsf{Hyb}_{1,i,4}$**:** This is the same as $\mathsf{Hyb}_{1,i,3}$ except that we undo the change between $\mathsf{Hyb}_{1,i,1}$ and $\mathsf{Hyb}_{1,i,2}$, i.e., we generate $\mathbf{W}_\alpha \leftarrow \mathbb{Z}_p^{2\times 2}$ for all $\alpha \in [n]$.

We have $|p_{\mathcal{A},H_{1,i,4}} - p_{\mathcal{A},H_{1,i,3}}| = 0$.

Lemma 5.7. *For all PPT adversaries $\mathcal{A}$, we have $|p_{\mathcal{A},H_{1,i,4}} - p_{\mathcal{A},H_{1,i,3}}| = 0$.*

Hybrid $\mathsf{Hyb}_{1,i,5}$**:** This is the same as $\mathsf{Hyb}_{1,i,4}$ except that for the i-th admissible query y^i, $O_{\mathsf{UserKeyGen}}$ returns $\mathsf{sk}_{y^i} := \{ \left([s_d^i \mathbf{b}]_2, \mathsf{kE}(y^i, [\mathbf{k}_d + w_d^i \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^i, \{ [s_d^i \mathbf{W}_\alpha^\top \mathbf{b}]_2 \}_{\alpha\in[n]}) \right) \}_{d\in[\ell]}$, where $s_d^i, w_d^i \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

We have $|p_{\mathcal{A},H_{1,i,5}} - p_{\mathcal{A},H_{1,i,4}}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$.

Lemma 5.8. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_{1,i,5}} - p_{\mathcal{A},H_{1,i,4}}| \le \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Final steps towards non-committing mode.

Hybrid Hyb_2**:** Iterating the above five transitions for $i = 1, \ldots, q$ yields $\mathsf{Hyb}_2 := \mathsf{Hyb}_{1,q,5}$. Concretely, it is the following experiment.

1. The challenger generates $\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}$ exactly as in Hyb_0.
2. **Pre-Challenge Query Phase:** Every admissible query y^j is answered by generating $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$ and returning $\mathsf{sk}_{y^j} := \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]}) \right) \}_{d\in[\ell]}$.
3. **Challenge Phase:** On challenge attribute x^*, the challenger generates $\mathbf{u} \leftarrow \mathbb{Z}_p^2$ and returns $\mathsf{ct}^* := ([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i \mathbf{u}]_1 \}_{i\in[n]}))$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_d]_T, h) \}_{d\in[\ell]}$.
4. **Post-Challenge Query Phase:** Every admissible query is answered exactly as in the pre-challenge query phase, while inadmissible queries are answered with $\perp$.
5. **Response Phase:** The challenger sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs $d \in \{0,1\}$.

Hybrid Hyb_3**:** This is the same as Hyb_2 except that we generate $k_{d,1}, k_{d,2} \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$ and set $\mathbf{k}_d := \frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{d,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp$. In particular, $\mathsf{mpk} = \left(\{ [k_{d,1}]_T \}_{d\in[\ell]}, h \right)$.

This is only a conceptual change, so $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.

Lemma 5.9. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.*

Hybrid Hyb_4**:** This is the same as Hyb_3 except that every admissible query y^j in either query phase is answered with $\mathsf{sk}_{y^j} := \{ \left([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_\alpha^\top \mathbf{b}]_2 \}_{\alpha\in[n]}) \right) \}_{d\in[\ell]}$, where $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

We have $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = 0$ since $\frac{k_{d,2}}{|\mathbf{a}^\perp|^2} + w_d^j$ and w_d^j are identically distributed for uniformly random w_d^j.

Lemma 5.10. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = 0$.*

Hybrid Hyb_5**:** This is the same as Hyb_4 except that we generate $u_1 \leftarrow \mathbb{Z}_p$ and $u_2 \leftarrow \mathbb{Z}_p^*$, and set $\mathbf{u} := u_1 \mathbf{a} + u_2 \mathbf{a}^\perp$.

The only difference between Hyb_4 and Hyb_5 is that Hyb_5 conditions on the event that the coefficient of $\mathbf{a}^\perp$ in $\mathbf{u}$ is nonzero. This event fails with probability exactly $1/p$, and therefore $|p_{\mathcal{A},H_5} - p_{\mathcal{A},H_4}| = \mathsf{negl}(\lambda)$.

Lemma 5.11. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have* $|p_{\mathcal{A},H_5} - p_{\mathcal{A},H_4}| = \mathsf{negl}(\lambda)$.

Hybrid Hyb_6: This is the same as Hyb_5 except that we generate $x_d \leftarrow \mathbb{Z}_p$ and set $k_{d,2} := (x_d - u_1 k_{d,1})/u_2$ for all $d \in [\ell]$. By this change, $\mathsf{seskey}^* = \{\, \mathsf{HC}([x_d]_T, h) \,\}_{d\in[\ell]}$.

We have $|p_{\mathcal{A},H_6} - p_{\mathcal{A},H_5}| = 0$ since $u_2 \in \mathbb{Z}_p^*$ and hence $k_{d,2}$ remains uniformly distributed when x_d is chosen uniformly at random.

Lemma 5.12. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have* $|p_{\mathcal{A},H_6} - p_{\mathcal{A},H_5}| = 0$.

Hybrid Hyb_7: This is the same as Hyb_6 except that we defer the generation of $k_{d,2}$, and hence of msk, to the response phase. The detailed description is as follows.

1. The challenger generates pp and mpk as follows.
   - Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
   - Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \ldots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $k_{d,1} \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$, and generate $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{\, [\mathbf{W}_i\mathbf{a}]_1 \,\}_{i\in[n]}, \{\, [\mathbf{W}_i^\top \mathbf{b}]_2 \,\}_{i\in[n]})$ and $\mathsf{mpk} := \Big(\{\, [k_{d,1}]_T \,\}_{d\in[\ell]}, h\Big)$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. **Pre-Challenge Query Phase:** For every admissible query $y^j \in \mathcal{Y}$, the challenger generates $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$ and returns $\mathsf{sk}_{y^j} := \Big\{\, \Big([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{\, [s_d^j \mathbf{W}_i^\top \mathbf{b}]_2 \,\}_{i\in[n]}) \Big) \,\Big\}_{d\in[\ell]}$.
3. **Challenge Phase:** $\mathcal{A}$ outputs $x^* \in \mathcal{X}$. The challenger generates $u_1 \leftarrow \mathbb{Z}_p$, $u_2 \leftarrow \mathbb{Z}_p^*$, and $x_d \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$, sets $\mathbf{u} := u_1\mathbf{a} + u_2\mathbf{a}^\perp$, and returns $\mathsf{ct}^* := \Big([\mathbf{u}]_1, \mathsf{sE}(x^*, \{\, [\mathbf{W}_i\mathbf{u}]_1 \,\}_{i\in[n]})\Big)$ and $\mathsf{seskey}^* := \{\, \mathsf{HC}([x_d]_T, h) \,\}_{d\in[\ell]}$.
4. **Post-Challenge Query Phase:** The challenger answers every admissible query exactly as in the pre-challenge query phase and returns $\perp$ on inadmissible queries.
5. **Response Phase:** The challenger sets $k_{d,2} := (x_d - u_1 k_{d,1})/u_2$ for all $d \in [\ell]$, defines $\mathsf{msk} := \{\, \frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{d,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp \,\}_{d\in[\ell]}$, sends msk to $\mathcal{A}$, and finally $\mathcal{A}$ outputs $d \in \{0,1\}$.

The view of $\mathcal{A}$ in Hyb_7 is identical to its view in Hyb_6, and therefore $|p_{\mathcal{A},H_7} - p_{\mathcal{A},H_6}| = 0$.

Lemma 5.13. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have* $|p_{\mathcal{A},H_7} - p_{\mathcal{A},H_6}| = 0$.

Hybrid Hyb_7 is exactly the simulated world realized by the simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$. Combining the hybrid transitions, we obtain

$$|p_{\mathcal{A},H_0} - p_{\mathcal{A},H_7}| \le p_{\mathcal{B}_0,\mathsf{DDH}} + \sum_{i=1}^{q} \ell \cdot p_{\mathcal{B}_{i,1},\mathsf{DDH}} + \sum_{i=1}^{q} \ell \cdot p_{\mathcal{B}_{i,5},\mathsf{DDH}} + \mathsf{negl}(\lambda)$$

for suitable PPT adversaries $\mathcal{B}_0, \{\, \mathcal{B}_{i,1} \,\}_{i\in[q]}, \{\, \mathcal{B}_{i,5} \,\}_{i\in[q]}$. Since ℓ and q are polynomially bounded and the SXDH assumption gives DDH hardness in both $\mathbb{G}_1$ and $\mathbb{G}_2$, the right-hand side is negligible. Therefore the theorem follows. □

Combining Theorem 4.6 and Theorem 5.2, we obtain the following.

Theorem 5.14. *Assuming the hardness of* SXDH*, there exist adaptively secure RNC-ABE schemes for predicate classes with predicate encodings.*

6 Receiver Non-Committing IBE from Batch Encryption

In this section, we construct an adaptively secure RNC-IBE scheme that supports polynomially many identities. Let

- $\mathsf{BE} = (\mathsf{BE.Params}, \mathsf{BE.Setup}, \mathsf{BE.Enc}, \mathsf{BE.Dec})$ be an oblivious batch encryption.
- $\mathsf{GC} = (\mathsf{GC.Grbl}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ be a garbling scheme.
- $\mathsf{NCE} = (\mathsf{NCE.Setup}, \mathsf{NCE.Enc}, \mathsf{NCE.Dec})$ be a NCE scheme with simulators $(\mathsf{NCE.Sim}_1, \mathsf{NCE.Sim}_2)$.

Let d be the identity length such that $T = 2^d = \mathrm{poly}(\lambda)$, let n be the length of the public keys of NCE, and let $\mathsf{int} : \{0,1\}^d \to [T]$ denote the lexicographic rank function.

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

- Generate $\mathsf{be.pp} \leftarrow \mathsf{BE.Params}(1^\lambda, 1^{nT})$.
- Generate $(\mathsf{nce.pk}_i, \mathsf{nce.sk}_i) \leftarrow \mathsf{NCE.Setup}(1^\lambda; r^{(i)}_{\mathsf{NCE.Setup}})$ for $i \in \{0,1\}^d$.
- Generate $\mathsf{be.pk} = \mathsf{BE.Setup}(\mathsf{be.pp}, \{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d})$.
- Output $\mathsf{msk} := \{\mathsf{nce.pk}_i, \mathsf{nce.sk}_i\}_{i\in\{0,1\}^d}$ and $\mathsf{mpk} := (\mathsf{be.pp}, \mathsf{be.pk})$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^d)$:

- Output $\mathsf{sk_{id}} := \left(\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{nce.sk_{id}}\right)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$:

- Generate r uniformly at random.
- Generate $\left(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{\mathsf{lab}^{(\mathsf{id})}_{i,b}\}\right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m; r))$.
- Generate $\left(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}^{(k)}_{i,b}\}\right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^k; r^k))$ where m^k, r^k are randomly generated for all $k \in \{0,1\}^d \setminus \{\mathsf{id}\}$.
- Generate a matrix $M \in [\{0,1\}^\lambda]^{nT\times 2}$ such that $M[(\mathsf{int}(k)-1)\cdot n + j, b] = \mathsf{lab}^{(k)}_{j,b}$ for all $k \in \{0,1\}^d$, $b \in \{0,1\}$, and $j \in [n]$.
- Compute $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{be.pk}, M)$.
- Output $\mathsf{ct} := \left(\{\tilde{\mathcal{C}}^{(k)}\}_{k\in\{0,1\}^d}, \mathsf{be.ct}\right)$.

$\mathsf{Dec}(\mathsf{sk_{id}}, \mathsf{ct})$:

- Compute $L \leftarrow \mathsf{BE.Dec}(\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{be.ct})$.
- Set $\mathsf{lab}_i := L[(\mathsf{int}(\mathsf{id})-1)\cdot n + i]$ for all $i \in [n]$.
- Compute $\mathsf{nce.ct_{id}} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{\mathsf{lab}_i\}_{i\in[n]})$.
- Output $m \leftarrow \mathsf{NCE.Dec}(\mathsf{nce.sk_{id}}, \mathsf{nce.ct_{id}})$.

$\mathsf{Sim}_1(1^\lambda, 1^T)$:

- Compute $\mathsf{be.pp} \leftarrow \mathsf{BE.Params}(1^\lambda, 1^{nT})$.
- Generate $(\mathsf{nce.pk}_i, \mathsf{nce.ct}_i, \mathsf{nce.st}_i) \leftarrow \mathsf{NCE.Sim}_1(1^\lambda)$ for $i \in \{0,1\}^d$.
- Compute $\mathsf{be.pk} := \mathsf{BE.Setup}(\mathsf{be.pp}, \{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d})$.

- Set $\mathsf{mpk} := (\mathsf{be.pp}, \mathsf{be.pk})$.
- Generate $\left(\tilde{\mathcal{C}}^{(k)}, \{\, \mathsf{lab}_j^{(k)} \,\}_{j\in[n]}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, \mathsf{nce.ct}_k)$ for all $k \in \{\,0,1\,\}^d$.
- Generate a matrix $M \in [\{\,0,1\,\}^\lambda]^{nT\times 2}$ and set $M[(\mathsf{int}(k)-1)\cdot n + j, b] = \mathsf{lab}_j^{(k)}$ for all $k \in \{\,0,1\,\}^d$, $j \in [n]$, and $b \in \{\,0,1\,\}$.
- Compute $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{be.pk}, M)$.
- Set $\mathsf{ct}^* := \left(\{\, \tilde{\mathcal{C}}^{(k)} \,\}_{k\in\{\,0,1\,\}^d}, \mathsf{be.ct}\right)$ and $\mathsf{st} := \left(\mathsf{be.pp}, \{\, (\mathsf{nce.pk}_i, \mathsf{nce.ct}_i, \mathsf{nce.st}_i) \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{ct}^*, \mathcal{R} = \emptyset, \bot\right)$.
- Output $(\mathsf{mpk}, \mathsf{st})$.

$\mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$:

- Parse $\mathsf{st} = \left(\mathsf{be.pp}, \{\, (\mathsf{nce.pk}_i, \mathsf{nce.ct}_i, \mathsf{nce.st}_i) \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{ct}^*, \mathcal{R}, \mathsf{id}^*\right)$.
- If $\mathcal{R}[\mathsf{id}] = \bot$, compute $(\cdot, r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_{\mathsf{id}}, m_{\mathsf{id}})$, where m_{id} is chosen uniformly at random, and set $\mathcal{R}[\mathsf{id}] := r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}$.
- Compute $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathcal{R}[\mathsf{id}])$.
- Output $\mathsf{sk}_{\mathsf{id}} := \left(\{\, \mathsf{nce.pk}_i \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{nce.sk}_{\mathsf{id}}\right)$ together with the updated state st.

$\mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$:

- Parse $\mathsf{st} = \left(\mathsf{be.pp}, \{\, (\mathsf{nce.pk}_i, \mathsf{nce.ct}_i, \mathsf{nce.st}_i) \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{ct}^*, \mathcal{R}, \bot\right)$.
- Output ct^* together with the updated state st, where the challenge identity is now recorded as id^*.

$\mathsf{Sim}_4(\mathsf{st}, m)$:

- Parse $\mathsf{st} = \left(\mathsf{be.pp}, \{\, (\mathsf{nce.pk}_i, \mathsf{nce.ct}_i, \mathsf{nce.st}_i) \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{ct}^*, \mathcal{R}, \mathsf{id}^*\right)$.
- If $\mathcal{R}[\mathsf{id}^*] = \bot$, compute $(\cdot, r^{(\mathsf{id}^*)}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_{\mathsf{id}^*}, m)$ and set $\mathcal{R}[\mathsf{id}^*] := r^{(\mathsf{id}^*)}_{\mathsf{NCE.Setup}}$.
- For every $i \in \{\,0,1\,\}^d \setminus \{\,\mathsf{id}^*\,\}$ such that $\mathcal{R}[i] = \bot$, compute $(\cdot, r^{(i)}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_i, m_i)$, where m_i is chosen uniformly at random, and set $\mathcal{R}[i] := r^{(i)}_{\mathsf{NCE.Setup}}$.
- Recompute $(\mathsf{nce.pk}_i, \mathsf{nce.sk}_i) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathcal{R}[i])$ for all $i \in \{\,0,1\,\}^d$.
- Output $\mathsf{msk} := \{\, \mathsf{nce.pk}_i, \mathsf{nce.sk}_i \,\}_{i\in\{\,0,1\,\}^d}$.

Remark 6.1. By the obliviousness of BE, the randomness used in the setup algorithm is $r_{\mathsf{Setup}} = \{\mathsf{be.pp}, \{r^{(i)}_{\mathsf{NCE.Setup}}\}_{i\in\{\,0,1\,\}^d}\}$. Moreover, the completed randomness map $\mathcal{R}$ produced inside Sim_4 also determines this string, so the same simulation additionally yields the stronger response-phase variant in which the challenger reveals r_{Setup} instead of msk.

Parameters. The size of a ciphertext is $\mathrm{poly}(2^d, |m|, \lambda)$, whereas the sizes of the master public key, master secret key, and secret keys are of the form $\mathrm{poly}(2^d, \lambda)$.

Correctness. For a correctly generated secret key $\mathsf{sk}_{\mathsf{id}} := \left(\{\, \mathsf{nce.pk}_i \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{nce.sk}_{\mathsf{id}}\right)$ and a ciphertext $\mathsf{ct} := \left(\{\, \tilde{\mathcal{C}}^{(k)} \,\}_{k\in\{\,0,1\,\}^d}, \mathsf{be.ct}\right)$, after performing the BE decryption $L \leftarrow \mathsf{BE.Dec}(\{\, \mathsf{nce.pk}_i \,\}_{i\in\{\,0,1\,\}^d}, \mathsf{be.ct})$, we have $L[(\mathsf{int}(\mathsf{id})-1)\cdot n + i] = \mathsf{lab}_{i,\mathsf{nce.pk}_{\mathsf{id}}[i]}$. This is due to the correctness of the BE scheme. From the correctness of the GC scheme, we have $\mathsf{nce.ct}_{\mathsf{id}} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{\, \mathsf{lab}_i \,\}_{i\in[n]})$ where $\mathsf{nce.ct}_{\mathsf{id}} = \mathsf{NCE.Enc}(\mathsf{nce.pk}_{\mathsf{id}}, m)$. Finally, by the decryption correctness of the NCE, we have $m \leftarrow \mathsf{NCE.Dec}(\mathsf{nce.sk}_{\mathsf{id}}, \mathsf{nce.ct}_{\mathsf{id}})$.

Theorem 6.2. *Assuming* BE *is a secure oblivious batch encryption,* GC *is a secure garbling scheme, and* NCE *is a secure non-committing encryption scheme, the above scheme is an adaptively secure RNC-IBE that supports polynomially many identities. Moreover, it remains secure even if the setup randomness is revealed to the adversary.*

Proof. We prove the stronger response-phase variant in which the challenger reveals r_{Setup} instead of msk. Since r_{Setup} deterministically determines msk by rerunning $\mathsf{NCE.Setup}$ for each identity, this immediately implies the standard adaptive RNC-IBE security as well.

Hybrid Hyb_0 : This is the stronger adaptive RNC-IBE game in which the challenger reveals r_{Setup} in the response phase.

1. The challenger generates $\mathsf{mpk}, \mathsf{msk}$ as follows.
   - Generate $\mathsf{be.pp} \leftarrow \mathsf{BE.Params}(1^\lambda, 1^{nT})$.
   - Generate $(\mathsf{nce.pk}_i, \mathsf{nce.sk}_i) \leftarrow \mathsf{NCE.Setup}(1^\lambda; r^{(i)}_{\mathsf{NCE.Setup}})$ for $i \in \{0,1\}^d$.
   - Generate $\mathsf{be.pk} = \mathsf{BE.Setup}(\mathsf{be.pp}, \{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d})$.
   - Set $\mathsf{msk} := \{\mathsf{nce.pk}_i, \mathsf{nce.sk}_i\}_{i\in\{0,1\}^d}$ and $\mathsf{mpk} := (\mathsf{be.pp}, \mathsf{be.pk})$.

   The challenger sends mpk to $\mathcal{A}$.
2. During the pre-challenge and post-challenge query phases, $\mathcal{A}$ has access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-th query id^j as input, the oracle returns $\mathsf{sk}_{\mathsf{id}^j} = \left(\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}^j}\right)$.
3. $\mathcal{A}$ outputs id^*, m^*. The challenger generates ct^* as follows.
   - Generate r uniformly at random.
   - Generate $\left(\tilde{\mathcal{C}}^{(\mathsf{id}^*)}, \{\mathsf{lab}^{(\mathsf{id}^*)}_{i,b}\}\right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^*; r))$.
   - Generate $\left(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}^{(k)}_{i,b}\}\right) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^k; r^k))$ where m^k, r^k are randomly generated for all $k \in \{0,1\}^d \setminus \{\mathsf{id}^*\}$.
   - Generate a matrix $M \in [\{0,1\}^\lambda]^{nT\times 2}$ such that $M[(\mathsf{int}(k)-1)\cdot n + j, b] = \mathsf{lab}^{(k)}_{j,b}$ for all $k \in \{0,1\}^d$, $b \in \{0,1\}$, and $j \in [n]$.
   - Compute $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{be.pk}, M)$.
   - Set $\mathsf{ct}^* := \left(\{\tilde{\mathcal{C}}^{(k)}\}_{k\in\{0,1\}^d}, \mathsf{be.ct}\right)$ and $r_{\mathsf{Setup}} := \{\mathsf{be.pp}, \{r^{(i)}_{\mathsf{NCE.Setup}}\}_{i\in\{0,1\}^d}\}$.

   The challenger sends ct^* to $\mathcal{A}$.
4. $\mathcal{A}$ may continue to query $O_{\mathsf{UserKeyGen}}(\mathsf{id})$ on any identity $\mathsf{id} \neq \mathsf{id}^*$. Finally, the challenger sends r_{Setup} to $\mathcal{A}$.
5. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Hybrid Hyb_1 : In this game, the matrix M during the challenge phase is computed as $M[(\mathsf{int}(k)-1)\cdot n + j, b] = \mathsf{lab}^{(k)}_{j,\mathsf{nce.pk}_k[j]}$ for all $j \in [n]$, $b \in \{0,1\}$, and $k \in \{0,1\}^d$.

Lemma 6.3. *Assume that* BE *is a secure batch encryption scheme. Then for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},\mathsf{Hyb}_1} - p_{\mathcal{A},\mathsf{Hyb}_0}| \leq \mathsf{negl}(\lambda)$.

Proof. This follows from the security of batch encryption. The reduction can generate $\{\mathsf{nce.pk}_i\}$ and send them to the challenger, who replies with $(\mathsf{be.pp}, \mathsf{be.pk})$. It can then set $\mathsf{mpk} := (\mathsf{be.pp}, \mathsf{be.pk})$. The reduction can generate the two matrices used in Hyb_0 and Hyb_1 and send them to the challenger. Note that these matrices differ only at indices $((\mathsf{int}(k)-1)\cdot n + j, 1 - \mathsf{nce.pk}_k[j])$. It receives $\mathsf{be.ct}$ from the challenger and then simulates the entire game using these values. Observe that the reduction generates all the keys $(\mathsf{nce.pk}_{\mathsf{id}}, \mathsf{nce.sk}_{\mathsf{id}})$, so it has both $r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}$ and $\mathsf{be.pp}$, and hence it can also form r_{Setup} exactly. □

Hybrid Hyb_2 : In this game, the garbled circuit $\tilde{\mathcal{C}}^{(k)}$ and labels are simulated, i.e., $(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}_j^{(k)}\}_{j\in[n]}) \leftarrow \mathsf{GC.Sim}(1^\lambda, \mathsf{nce.ct}^{(k)})$ where $\mathsf{nce.ct}^{(k)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}_k, m^{(k)}; r^{(k)})$ for all $k \in \{0,1\}^d \setminus \{\mathsf{id}^*\}$. Also, $\mathsf{nce.ct}^{(\mathsf{id}^*)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}_{\mathsf{id}^*}, m^*)$.

Lemma 6.4. *Assume that* GC *is a secure garbling scheme. Then for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},\mathsf{Hyb}_2} - p_{\mathcal{A},\mathsf{Hyb}_1}| \leq \mathsf{negl}(\lambda)$.

Proof. This directly follows from the security of GC because the matrix M requires only $\mathsf{lab}^{(\mathsf{id})}_{i,\mathsf{nce.pk}_{\mathsf{id}}[i]}$ labels for all $\mathsf{id} \in \{0,1\}^d$. □

Hybrid Hyb_3 : In this game, $\mathsf{nce.ct}^{(\mathsf{id})}$ and $\mathsf{nce.sk}_{\mathsf{id}}$ are all simulated using the NCE simulators.

Lemma 6.5. *Assume that* NCE *is a secure NCE scheme. Then for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},\mathsf{Hyb}_3} - p_{\mathcal{A},\mathsf{Hyb}_2}| \leq \mathsf{negl}(\lambda)$.

Proof. We will show that an adversary $\mathcal{A}$ that can distinguish Hyb_3 from Hyb_2 can be transformed into an adversary $\mathcal{B}$ that breaks the NCE security. We achieve this by considering $T+1$ intermediate hybrids $\mathsf{Hyb}_{2,i}$ where $\mathsf{Hyb}_{2,0} = \mathsf{Hyb}_2$ and $\mathsf{Hyb}_{2,T} = \mathsf{Hyb}_3$. The description of $\mathsf{Hyb}_{2,i}$ is as follows.

- The challenger generates the first i public keys using the first NCE simulator. Recall that the simulator will also generate fake ciphertexts. The remaining public-secret keys are generated honestly. It also maintains a partial map $\mathcal{R}$ from identities to setup randomness, initialized to $\emptyset$.
- To respond to a key query, if $\mathsf{int}(\mathsf{id}) > i$, it returns $\mathsf{sk}_{\mathsf{id}} = \left(\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}}\right)$. If $\mathsf{int}(\mathsf{id}) \leq i$, it first checks whether $\mathcal{R}[\mathsf{id}] \neq \perp$. If so, it computes $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathcal{R}[\mathsf{id}])$ and returns $\mathsf{sk}_{\mathsf{id}}$ accordingly. Otherwise, it calls the second simulator of the NCE on a random message m, obtains $(\cdot, r^{(\mathsf{id})}_{\mathsf{NCE.Setup}})$, sets $\mathcal{R}[\mathsf{id}] := r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}$, computes $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathcal{R}[\mathsf{id}])$, and then returns $\mathsf{sk}_{\mathsf{id}}$ accordingly.
- In the challenge phase, when it receives the challenge identity id^* and message m^*, it generates $\mathsf{nce.ct}^{(k)}$ for all k such that $\mathsf{int}(k) > i$ honestly, i.e., using the $\mathsf{NCE.Enc}$ algorithm. The remaining $\mathsf{nce.ct}^{(k)}$ for k such that $\mathsf{int}(k) \leq i$ are the ciphertexts simulated by the first simulator in the initialization phase. It then generates $(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}_j^{(k)}\}_{j\in[n]})$ for all k and sets up M appropriately to generate $\mathsf{be.ct}$. Finally,
  - if $\mathsf{int}(\mathsf{id}^*) \leq i$, it uses the NCE simulator on m^* to obtain $r^{(\mathsf{id}^*)}_{\mathsf{NCE.Setup}}$ and sets $\mathcal{R}[\mathsf{id}^*] := r^{(\mathsf{id}^*)}_{\mathsf{NCE.Setup}}$.
  - For all id such that $\mathsf{int}(\mathsf{id}) \leq i$ that were not queried and are not equal to id^*, it uses the NCE simulator on a random message to obtain $r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}$ and sets $\mathcal{R}[\mathsf{id}] := r^{(\mathsf{id})}_{\mathsf{NCE.Setup}}$.

  Finally, it outputs $\mathsf{ct}^* := \left(\{\tilde{\mathcal{C}}^{(k)}\}_{k\in\{0,1\}^d}, \mathsf{be.ct}\right)$ together with $r_{\mathsf{Setup}} := \{\mathsf{be.pp}, \{\mathcal{R}[\mathsf{id}]\}_{\mathsf{id}\in\{0,1\}^d}\}$.

It is easy to see that an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hyb}_{2,i}$ from $\mathsf{Hyb}_{2,i+1}$ can be transformed into an adversary $\mathcal{B}$ that breaks the NCE security. □

Using the above lemmas and the triangle inequality, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},\mathsf{Hyb}_0} - p_{\mathcal{A},\mathsf{Hyb}_3}| \leq \mathsf{negl}(\lambda)$. □

Using Theorem 3.5 and Theorem 3.3, we have Theorem 1.3.

7 Receiver Non-Committing Identity-Based Encryption from Indistinguishable Obfuscation

In this section, we construct a selectively secure receiver non-committing identity-based encryption scheme using indistinguishability obfuscation (iO). Let

- $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Setup}, \mathsf{SS.Sign}, \mathsf{SS.Enc}, \mathsf{SS.Dec})$ be a secure one-time signature encryption scheme.
- $\mathsf{PPRF} = (\mathsf{PPRF.Setup}, \mathsf{PPRF.Eval}, \mathsf{PPRF.Program}, \mathsf{PPRF.PEval})$ be a secure programmable PRF[4].
- iO be a secure indistinguishable obfuscator.
- $\mathsf{NCE} = (\mathsf{NCE.Setup}, \mathsf{NCE.Enc}, \mathsf{NCE.Dec})$ be a non-committing encryption scheme with simulators $(\mathsf{NCE.Sim}_1, \mathsf{NCE.Sim}_2)$.

An n-bit string id is denoted as $\mathsf{id}[1]\|\mathsf{id}[2]\|\ldots\|\mathsf{id}[n]$, and we use the notation $\mathsf{id}[i,j]$ to denote $\mathsf{id}[i]\|\mathsf{id}[i+1]\|\ldots\|\mathsf{id}[j]$ for all $1 \le i \le j \le n$. For any $i > j$, we set $\mathsf{id}[i,j] = \epsilon$. We first describe two deterministic helper algorithms whose randomness is derived from the programmed PRF key s.

$\mathsf{Node}(\mathsf{pp}, v, s)$:

- Generate $(\mathsf{ss.vk}^{(v)}, \mathsf{ss.sk}^{(v)}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PPRF.PEval}(s, v))$.
- Generate $(\mathsf{ss.vk}^{(v\|0)}, \mathsf{ss.sk}^{(v\|0)}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PPRF.PEval}(s, v\|0))$.
- Generate $(\mathsf{ss.vk}^{(v\|1)}, \mathsf{ss.sk}^{(v\|1)}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PPRF.PEval}(s, v\|1))$.
- Set $x^{(v)} := \mathsf{ss.vk}^{(v\|0)}\|\mathsf{ss.vk}^{(v\|1)}$.
- Compute $\mathsf{ss}.\sigma^{(v)} \leftarrow \mathsf{SS.Sign}(\mathsf{pp}, \mathsf{ss.sk}^{(v)}, x^{(v)})$.
- Return $(\mathsf{ss.vk}^{(v)}, x^{(v)}, \mathsf{ss}.\sigma^{(v)})$.

$\mathsf{Leaf}(\mathsf{pp}, v, s)$:

- Generate $(\mathsf{ss.vk}^{(v)}, \mathsf{ss.sk}^{(v)}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PPRF.PEval}(s, v))$.
- Generate $(\mathsf{nce.pk}^{(v\|0)}, \mathsf{nce.sk}^{(v\|0)}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.PEval}(s, v\|0))$.
- Generate $(\mathsf{nce.pk}^{(v\|1)}, \mathsf{nce.sk}^{(v\|1)}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.PEval}(s, v\|1))$.
- Set $x^{(v)} := \mathsf{nce.pk}^{(v\|0)}\|\mathsf{nce.pk}^{(v\|1)}$.
- Compute $\mathsf{ss}.\sigma^{(v)} \leftarrow \mathsf{SS.Sign}(\mathsf{pp}, \mathsf{ss.sk}^{(v)}, x^{(v)})$.
- Return $(\mathsf{ss.vk}^{(v)}, x^{(v)}, \mathsf{ss}.\sigma^{(v)}, \{\mathsf{nce.sk}^{(v\|b)}\}_{b\in\{0,1\}})$.

We now describe the algorithms for our RNC-IBE scheme. For simplicity, we assume that NCE public keys and OTSE verification keys both have length λ.

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

- Generate $\mathsf{pprf.msk} \leftarrow \mathsf{PPRF.Setup}(1^\lambda, 1^{n+1})$.
- Generate independent uniformly random values $\{t_i\}_{i\in[n]_0}$.
- Compute $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{0^i\}_{i\in[n]_0}, \{t_i\}_{i\in[n]_0})$.
- Generate $\mathsf{pp} \leftarrow \mathsf{SS.Gen}(1^\lambda, 1^{2\lambda})$.
- Compute $(\mathsf{ss.vk}^{(\epsilon)}, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$.
- Define the circuits $C_{\mathsf{Node}}(v) := \mathsf{Node}(\mathsf{pp}, v, s)$ and $C_{\mathsf{Leaf}}(v) := \mathsf{Leaf}(\mathsf{pp}, v, s)$.

[4] As mentioned in the introduction, we can also use a punctured PRF.

- Set $\mathsf{mpk} := (\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)})$ and $\mathsf{msk} := (\mathsf{iO}(C_{\mathsf{Node}}), \mathsf{iO}(C_{\mathsf{Leaf}}))$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$:

- Parse $(C_{\mathsf{Node}}, C_{\mathsf{Leaf}}) \leftarrow \mathsf{msk}$.
- Compute $(\mathsf{ss.vk}^{(\epsilon)}, x^{(\epsilon)}, \sigma^{(\epsilon)}) \leftarrow C_{\mathsf{Node}}(\epsilon)$.
- For $j = 1$ to $n-2$, compute $(\mathsf{ss.vk}^{(\mathsf{id}[1,j])}, x^{(\mathsf{id}[1,j])}, \sigma^{(\mathsf{id}[1,j])}) \leftarrow C_{\mathsf{Node}}(\mathsf{id}[1,j])$.
- Compute $(\cdot, x^{(\mathsf{id}[1,n-1])}, \sigma^{(\mathsf{id}[1,n-1])}, \{\mathsf{nce.sk}^{(\mathsf{id}[1,n-1]\|b)}\}_{b\in\{0,1\}}) \leftarrow C_{\mathsf{Leaf}}(\mathsf{id}[1,n-1])$.
- Return $\mathsf{sk}_{\mathsf{id}} := (\mathsf{ss.vk}^{(\epsilon)}, \{(x^{(v)}, \sigma^{(v)})\}_{v\in\{\mathsf{id}[1,j]\}_{j\in[n-1]_0}}, \mathsf{nce.sk}^{(\mathsf{id})})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$:

- Parse $(\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)}) \leftarrow \mathsf{mpk}$.
- Let $T_m(\mathsf{pk})$ be the circuit that computes $\mathsf{NCE.Enc}(\mathsf{pk}, m)$.
- For $\beta \in \{0,1\}$ and labels $\{z_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$, let $Q_{\beta,\{z_{i,b}\}}(\mathsf{vk})$ be the circuit that computes $\{\mathsf{SS.Enc}(\mathsf{pp}, \mathsf{vk}, \beta\cdot\lambda + i, b, z_{i,b})\}_{i\in[\lambda],b\in\{0,1\}}$.
- Generate $(\tilde{\mathcal{T}}, \{\mathsf{lab}_{i,b}^{(n)}\}_{i\in[\lambda],b\in\{0,1\}}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, T_m)$.
- For $k = n-1$ down to 0, generate $(\tilde{Q}^{(k)}, \{\mathsf{lab}_{i,b}^{(k)}\}_{i\in[\lambda],b\in\{0,1\}}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, Q_{\mathsf{id}[k+1],\{\mathsf{lab}_{i,b}^{(k+1)}\}})$.
- Return $\mathsf{ct} := (\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}\}_{i\in[\lambda]}, \{\tilde{Q}^{(k)}\}_{k\in[n-1]_0}, \tilde{\mathcal{T}})$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$:

- Parse $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ss.vk}^{(\epsilon)}, \{(x^{(v)}, \sigma^{(v)})\}_{v\in\{\mathsf{id}[1,j]\}_{j\in[n-1]_0}}, \mathsf{nce.sk}^{(\mathsf{id})})$.
- Parse $\mathsf{ct} = (\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}\}_{i\in[\lambda]}, \{\tilde{Q}^{(k)}\}_{k\in[n-1]_0}, \tilde{\mathcal{T}})$.
- Set $v_0 := \epsilon$ and $v_k := \mathsf{id}[1,k]$ for all $k \in [n-1]$.
- Set $\mathsf{vk}^{(0)} := \mathsf{ss.vk}^{(\epsilon)}$ and $\mathsf{lab}_i^{(0)} := \mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}$ for all $i \in [\lambda]$.
- For $k = 0$ to $n-1$:
  1. Compute $\{\mathsf{ss.ct}_{i,b}^{(k)}\}_{i\in[\lambda],b\in\{0,1\}} \leftarrow \mathsf{GC.Eval}(\tilde{Q}^{(k)}, \{\mathsf{lab}_i^{(k)}\}_{i\in[\lambda]})$.
  2. Write $x^{(v_k)} = x_0^{(v_k)} \| x_1^{(v_k)}$ where each block has length λ, and set $w^{(k+1)} := x^{(v_k)}_{\mathsf{id}[k+1]}$.
  3. For all $i \in [\lambda]$, compute $\mathsf{lab}_i^{(k+1)} \leftarrow \mathsf{SS.Dec}(\mathsf{pp}, \mathsf{vk}^{(k)}, x^{(v_k)}, \sigma^{(v_k)}, \mathsf{ss.ct}^{(k)}_{i,w^{(k+1)}[i]})$.
  4. If $k < n-1$, set $\mathsf{vk}^{(k+1)} := w^{(k+1)}$. Otherwise, set $\mathsf{nce.pk}^{(\mathsf{id})} := w^{(n)}$.
- Compute $\mathsf{nce.ct} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{T}}, \{\mathsf{lab}_i^{(n)}\}_{i\in[\lambda]})$.
- Return $m \leftarrow \mathsf{NCE.Dec}(\mathsf{nce.sk}^{(\mathsf{id})}, \mathsf{nce.ct})$.

$\mathsf{Sim}_1(1^\lambda, 1^n, \mathsf{id}^*)$:

- Generate $\mathsf{pprf.msk} \leftarrow \mathsf{PPRF.Setup}(1^\lambda, 1^{n+1})$.
- Generate uniformly random values $\{r^{(\mathsf{id}^*[1,k])}\}_{k\in[n-1]_0}$.
- Compute $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,k]\}_{k\in[n-1]_0}, \{r^{(\mathsf{id}^*[1,k])}\}_{k\in[n-1]_0})$.
- Generate $\mathsf{pp} \leftarrow \mathsf{SS.Gen}(1^\lambda, 1^{2\lambda})$.

- Compute $(\mathsf{ss.vk}^{(\epsilon)}, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$ and set $\mathsf{mpk} := (\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)})$.
- Generate $(\mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.ct}^{(\mathsf{id}^*)}, \mathsf{nce.st}^{(\mathsf{id}^*)}) \leftarrow \mathsf{NCE.Sim}_1(1^\lambda)$.
- Compute $(\tilde{\mathcal{T}}, \{\mathsf{lab}_i^{(n)}\}_{i\in[\lambda]}) \leftarrow \mathsf{GC.Sim}(1^\lambda, \mathsf{nce.ct}^{(\mathsf{id}^*)})$.
- For $k = n-1$ down to 0, compute $(\tilde{\mathcal{Q}}^{(k)}, \{\mathsf{lab}_i^{(k)}\}_{i\in[\lambda]}) \leftarrow \mathsf{GC.Sim}(1^\lambda, Q_{\mathsf{id}^*[k+1], \{\mathsf{lab}_i^{(k+1)}\}}(\mathsf{ss.vk}^{(\mathsf{id}^*[1,k])}))$, where $(\mathsf{ss.vk}^{(\mathsf{id}^*[1,k])}, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \mathsf{id}^*[1,k], s)$.
- Set $\mathsf{ct}^* := (\{\mathsf{lab}^{(0)}_{i, \mathsf{ss.vk}^{(\epsilon)}[i]}\}_{i\in[\lambda]}, \{\tilde{\mathcal{Q}}^{(k)}\}_{k\in[n-1]_0}, \tilde{\mathcal{T}})$.
- Set $\mathsf{st} := (\mathsf{mpk}, \mathsf{id}^*, \mathsf{pprf.msk}, s, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k\in[n-1]_0}, \mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.st}^{(\mathsf{id}^*)}, \mathsf{ct}^*)$.
- Return $(\mathsf{mpk}, \mathsf{st})$.

$\mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$:

- Parse $\mathsf{st} = ((\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)}), \mathsf{id}^*, \mathsf{pprf.msk}, s, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k\in[n-1]_0}, \mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.st}^{(\mathsf{id}^*)}, \mathsf{ct}^*)$.
- Compute $(\cdot, x^{(\epsilon)}, \sigma^{(\epsilon)}) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$.
- For $j = 1$ to $n-2$, compute $(\cdot, x^{(\mathsf{id}[1,j])}, \sigma^{(\mathsf{id}[1,j])}) \leftarrow \mathsf{Node}(\mathsf{pp}, \mathsf{id}[1,j], s)$.
- Set $v := \mathsf{id}[1, n-1]$.
- If id is not the sibling of id^*, compute $(\cdot, x^{(v)}, \sigma^{(v)}, \{\mathsf{nce.sk}^{(v\|b)}\}_{b\in\{0,1\}}) \leftarrow \mathsf{Leaf}(\mathsf{pp}, v, s)$.
- Otherwise, generate $(\mathsf{ss.vk}^{(v)}, \mathsf{ss.sk}^{(v)}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; r^{(v)})$.
- In the sibling case, generate $(\mathsf{nce.pk}^{(\mathsf{id})}, \mathsf{nce.sk}^{(\mathsf{id})}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.PEval}(s, \mathsf{id}))$.
- If $\mathsf{id}[n] = 0$, set $x^{(v)} := \mathsf{nce.pk}^{(\mathsf{id})} \| \mathsf{nce.pk}^{(\mathsf{id}^*)}$.
- Otherwise, set $x^{(v)} := \mathsf{nce.pk}^{(\mathsf{id}^*)} \| \mathsf{nce.pk}^{(\mathsf{id})}$.
- In the sibling case, compute $\sigma^{(v)} \leftarrow \mathsf{SS.Sign}(\mathsf{pp}, \mathsf{ss.sk}^{(v)}, x^{(v)})$.
- Set $\mathsf{sk}_{\mathsf{id}} := (\mathsf{ss.vk}^{(\epsilon)}, \{(x^{(v)}, \sigma^{(v)})\}_{v\in\{\,\mathsf{id}[1,j]\,\}_{j\in[n-1]_0}}, \mathsf{nce.sk}^{(\mathsf{id})})$.
- Return $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st})$.

$\mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$:

- Parse $\mathsf{st} = (\mathsf{mpk}, \mathsf{id}^*, \mathsf{pprf.msk}, s, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k\in[n-1]_0}, \mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.st}^{(\mathsf{id}^*)}, \mathsf{ct}^*)$.
- Return $(\mathsf{ct}^*, \mathsf{st})$.

$\mathsf{Sim}_4(\mathsf{st}, m^*)$:

- Parse $\mathsf{st} = ((\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)}), \mathsf{id}^*, \mathsf{pprf.msk}, s, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k\in[n-1]_0}, \mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.st}^{(\mathsf{id}^*)}, \mathsf{ct}^*)$.
- Compute $(r^{(\mathsf{id}^*)}_{\mathsf{Enc}}, r^{(\mathsf{id}^*)}_{\mathsf{Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}^{(\mathsf{id}^*)}, m^*)$.
- Set $r^{(\mathsf{id}^*)} := r^{(\mathsf{id}^*)}_{\mathsf{Setup}}$ and compute $s^* \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\, \mathsf{id}^*[1,k] \,\}_{k\in[n]_0}, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k\in[n]_0})$.
- Define the circuits $C^*_{\mathsf{Node}}(v) := \mathsf{Node}(\mathsf{pp}, v, s^*)$ and $C^*_{\mathsf{Leaf}}(v) := \mathsf{Leaf}(\mathsf{pp}, v, s^*)$.
- Return $\mathsf{msk} := (\mathsf{iO}(C^*_{\mathsf{Node}}), \mathsf{iO}(C^*_{\mathsf{Leaf}}))$.

Parameters. The size of a ciphertext is $\mathrm{poly}(n, |m|, \lambda)$ and the size of the master public key is $\mathrm{poly}(\lambda)$. The sizes of the master secret key and user secret keys are of the form $\mathrm{poly}(n, \lambda)$.

Correctness. Fix an identity $\mathsf{id} \in \{0,1\}^n$, let $v_0 := \epsilon$ and $v_k := \mathsf{id}[1,k]$ for all $k \in [n-1]$, and consider a well-formed ciphertext $\mathsf{ct} = (\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}\}_{i\in[\lambda]}, \{\tilde{\mathcal{Q}}^{(k)}\}_{k\in[n-1]_0}, \tilde{\mathcal{T}})$ and a correctly generated secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ss.vk}^{(\epsilon)}, \{(x^{(v)}, \sigma^{(v)})\}_{v\in\{\mathsf{id}[1,j]\}_{j\in[n-1]_0}}, \mathsf{nce.sk}^{(\mathsf{id})})$. By the correctness of the garbling scheme, for every $k \in [n-1]_0$, we have $\mathsf{GC.Eval}(\tilde{\mathcal{Q}}^{(k)}, \{\mathsf{lab}^{(k)}_i\}_{i\in[\lambda]}) \to \{\mathsf{ss.ct}^{(k)}_{i,b} = \mathsf{SS.Enc}(\mathsf{pp}, \mathsf{ss.vk}^{(v_k)}, \mathsf{id}[k+1]\cdot\lambda + i, b, \mathsf{lab}^{(k+1)}_{i,b})\}_{i\in[\lambda],b\in\{0,1\}}$. Writing $x^{(v_k)} = x_0^{(v_k)} \| x_1^{(v_k)}$, the correctness of the OTSE scheme implies that for $w^{(k+1)} := x^{(v_k)}_{\mathsf{id}[k+1]}$ we recover $\mathsf{lab}^{(k+1)}_i = \mathsf{SS.Dec}(\mathsf{pp}, \mathsf{ss.vk}^{(v_k)}, x^{(v_k)}, \sigma^{(v_k)}, \mathsf{ss.ct}^{(k)}_{i,w^{(k+1)}[i]})$ for every $i \in [\lambda]$. Iterating this argument yields the labels $\{\mathsf{lab}^{(n)}_i\}_{i\in[\lambda]}$ corresponding to the challenge public key $\mathsf{nce.pk}^{(\mathsf{id})}$. A final application of garbling correctness gives $\mathsf{GC.Eval}(\tilde{\mathcal{T}}, \{\mathsf{lab}^{(n)}_i\}_{i\in[\lambda]}) = \mathsf{NCE.Enc}(\mathsf{nce.pk}^{(\mathsf{id})}, m)$, and the correctness of the NCE scheme then implies that $\mathsf{NCE.Dec}(\mathsf{nce.sk}^{(\mathsf{id})}, \mathsf{nce.ct}) = m$.

Theorem 7.1. *Assuming* SS *is a secure OTSE scheme,* iO *is a secure indistinguishable obfuscator,* PPRF *is a secure privately programmable PRF, and* NCE *is a secure non-committing encryption scheme, the above scheme is a selectively secure RNC-IBE scheme.*

Proof sketch. Let $\mathcal{A}$ be an adversary in the selective RNC-IBE security game, and let id^* be the identity it commits to at the beginning of the game. We write $p_{\mathcal{A},H}$ for the probability that $\mathcal{A}$ outputs 0 in hybrid H.

Hybrid Hyb_0**:** This is the real selective RNC-IBE experiment.

1. $\mathcal{A}$ outputs the challenge identity $\mathsf{id}^* \in \{0,1\}^n$.
2. **Initialization Phase:** The challenger computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
3. **Pre-Challenge Query Phase:** For every query $\mathsf{id} \neq \mathsf{id}^*$, the challenger returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
4. **Challenge Phase:** $\mathcal{A}$ outputs a message m^*. The challenger computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$ and returns ct^*.
5. **Post-Challenge Query Phase:** The challenger continues to answer every query $\mathsf{id} \neq \mathsf{id}^*$ with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ and returns $\perp$ on the query id^*.
6. **Response Phase:** The challenger sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs a bit d.

We now transform Hyb_0 into the simulated world. For convenience, define $H_{1,-1,4} := \mathsf{Hyb}_0$.

Step 1: Simulating the garbled circuits on the challenge path. For every $k \in [n-2]_0$, we consider hybrids $H_{1,k,0}, \ldots, H_{1,k,4}$.

Hybrid $\mathsf{Hyb}_{1,k,0}$**:** This is the same as $H_{1,k-1,4}$ except that the garbled circuit $\tilde{\mathcal{Q}}^{(k)}$ in the challenge ciphertext is generated by the garbling simulator on the output of $Q_{\mathsf{id}^*[k+1],\{\mathsf{lab}^{(k+1)}_{i,b}\}}(\mathsf{ss.vk}^{(\mathsf{id}^*[1,k])})$.

Only the labels consistent with $\mathsf{ss.vk}^{(\mathsf{id}^*[1,k])}$ are ever used, so the security of the garbling scheme implies that $|p_{\mathcal{A},H_{1,k,0}} - p_{\mathcal{A},H_{1,k-1,4}}| \leq \mathsf{negl}(\lambda)$.

Lemma 7.2. *Assume that* GC *is a secure garbling scheme. Then for all PPT adversaries $\mathcal{A}$, all $k \in [n-2]_0$, and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_{1,k,0}} - p_{\mathcal{A},H_{1,k-1,4}}| \leq \mathsf{negl}(\lambda)$.*

Hybrid $\mathsf{Hyb}_{1,k,1}$**:** This is the same as $H_{1,k,0}$ except that, instead of programming the point 0^k in the PRF, we program the point $\mathsf{id}^*[1,k]$ with an independent uniformly random value.

This change is hidden by the privacy of the programmable PRF, and hence $|p_{\mathcal{A},H_{1,k,1}} - p_{\mathcal{A},H_{1,k,0}}| \leq \mathsf{negl}(\lambda)$.

Lemma 7.3. *Assume that* PPRF *is a secure programmable PRF scheme. Then for all* PPT *adversaries* $\mathcal{A}$*, all* $k \in [n-2]_0$*, and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{1,k,1}} - p_{\mathcal{A},H_{1,k,0}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,2}$**:** This is the same as $H_{1,k,1}$ except that, inside the response-phase master secret key, we replace $\mathsf{iO}(C_{\mathsf{Node}})$ by an obfuscation of a circuit $P_{\mathsf{Node},k}$ that behaves exactly like C_{Node} on every input except $v = \mathsf{id}^*[1,k]$, where it uses hardcoded OTSE keys and the corresponding signature instead of deriving them from s.

Since the point $\mathsf{id}^*[1,k]$ is already programmed to a uniformly random value in $H_{1,k,1}$, the hardcoded keys in $P_{\mathsf{Node},k}$ have exactly the same distribution as the keys derived from $\mathsf{PPRF.PEval}(s, \mathsf{id}^*[1,k])$. Therefore C_{Node} and $P_{\mathsf{Node},k}$ are functionally equivalent, and indistinguishability follows from the security of iO.

Lemma 7.4. *Assume that* iO *is a secure indistinguishable obfuscator. Then for all* PPT *adversaries* $\mathcal{A}$*, all* $k \in [n-2]_0$*, and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{1,k,2}} - p_{\mathcal{A},H_{1,k,1}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,3}$**:** This is the same as $H_{1,k,2}$ except that in $\tilde{\mathcal{Q}}^{(k)}$ we switch the hidden OTSE branch from the sibling verification key to the challenge-path verification key.

This is exactly the selective OTSE security game on the signature of $x^{(\mathsf{id}^*[1,k])}$, and therefore $|p_{\mathcal{A},H_{1,k,3}} - p_{\mathcal{A},H_{1,k,2}}| \leq \mathsf{negl}(\lambda)$.

Lemma 7.5. *Assume that* SS *is a secure OTSE scheme. Then for all* PPT *adversaries* $\mathcal{A}$*, all* $k \in [n-2]_0$*, and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{1,k,3}} - p_{\mathcal{A},H_{1,k,2}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,4}$**:** This is the same as $H_{1,k,3}$ except that we replace the obfuscation of $P_{\mathsf{Node},k}$ back by $\mathsf{iO}(C_{\mathsf{Node}})$.

After the branch switch, the hardcoded circuit and the honest circuit are again functionally equivalent, so indistinguishability follows from iO.

Lemma 7.6. *Assume that* iO *is a secure indistinguishable obfuscator. Then for all* PPT *adversaries* $\mathcal{A}$*, all* $k \in [n-2]_0$*, and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{1,k,4}} - p_{\mathcal{A},H_{1,k,3}}| \leq \mathsf{negl}(\lambda)$.

Step 2: Handling the final leaf. We next perform the analogous sequence of transitions for the last level of the tree.

Hybrid $\mathsf{Hyb}_{1,n-1,0}$**:** This is the same as $H_{1,n-2,4}$ except that $\tilde{\mathcal{Q}}^{(n-1)}$ is generated by the garbling simulator on $Q_{\mathsf{id}^*[n],\{\mathsf{lab}_{i,b}^{(n)}\}}(\mathsf{ss.vk}^{(\mathsf{id}^*[1,n-1])})$.

Hybrid $\mathsf{Hyb}_{1,n-1,1}$**:** This is the same as $H_{1,n-1,0}$ except that we program the point $\mathsf{id}^*[1,n-1]$ with a uniformly random value.

Hybrid $\mathsf{Hyb}_{1,n-1,2}$**:** This is the same as $H_{1,n-1,1}$ except that in the response phase we replace $\mathsf{iO}(C_{\mathsf{Leaf}})$ by an obfuscation of a circuit that hardcodes the leaf keys at the node $\mathsf{id}^*[1,n-1]$.

Hybrid $\mathsf{Hyb}_{1,n-1,3}$**:** This is the same as $H_{1,n-1,2}$ except that the hidden OTSE branch in $\tilde{\mathcal{Q}}^{(n-1)}$ is switched from the sibling NCE public key to the challenge NCE public key.

Hybrid $\mathsf{Hyb}_{1,n-1,4}$**:** This is the same as $H_{1,n-1,3}$ except that we replace the hardcoded obfuscation back by $\mathsf{iO}(C_{\mathsf{Leaf}})$.

Exactly the same arguments as in Step 1 apply here, with Leaf in place of Node and $\mathsf{nce.pk}^{(\mathsf{id}^*)}$ in place of the next verification key. Hence every transition in this final five-hybrid block is indistinguishable up to a negligible additive term.

Step 3: Removing the message dependence from the challenge ciphertext.

Hybrid $\mathsf{Hyb}_{2,0}$**:** This is the same as $H_{1,n-1,4}$ except that $\tilde{\mathcal{T}}$ is generated by the garbling simulator on a fake NCE ciphertext $\mathsf{nce.ct}^{(\mathsf{id}^*)}$ produced by $\mathsf{NCE.Sim}_1(1^\lambda)$.

Since the challenge ciphertext now reveals only the labels corresponding to $\mathsf{nce.pk}^{(\mathsf{id}^*)}$, this change is hidden by the security of the garbling scheme.

Lemma 7.7. *Assume that* GC *is a secure garbling scheme. Then for all* PPT *adversaries* $\mathcal{A}$ *and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{2,0}} - p_{\mathcal{A},H_{1,n-1,4}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{2,1}$**:** This is the same as $H_{2,0}$ except that we also program the full challenge identity id^* with an independent uniformly random value.

This change is hidden by the privacy of the programmable PRF.

Lemma 7.8. *Assume that* PPRF *is a secure programmable PRF scheme. Then for all* PPT *adversaries* $\mathcal{A}$ *and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{2,1}} - p_{\mathcal{A},H_{2,0}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{2,2}$**:** This is the same as $H_{2,1}$ except that we defer the choice of the programmed value at id^* until the response phase. Concretely, in the initialization phase the challenger runs $\mathsf{Sim}_1(1^\lambda, 1^n, \mathsf{id}^*)$ and keeps the resulting state. In both query phases it answers every admissible query $\mathsf{id} \neq \mathsf{id}^*$ using $\mathsf{Sim}_2(\mathsf{st}, \mathsf{id})$. In the challenge phase, on input m^*, it returns the precomputed challenge ciphertext from $\mathsf{Sim}_3(\mathsf{st}, \mathsf{id}^*)$. Finally, in the response phase it computes $\mathsf{msk} \leftarrow \mathsf{Sim}_4(\mathsf{st}, m^*)$ and sends msk to $\mathcal{A}$.

The indistinguishability of $H_{2,2}$ from $H_{2,1}$ follows directly from the security of the underlying NCE scheme: the only remaining difference is whether the challenge NCE public key and the fake ciphertext were fixed honestly in advance or produced by the NCE simulators and later opened to m^* in the response phase.

Lemma 7.9. *Assume that* NCE *is a secure non-committing encryption scheme. Then for all* PPT *adversaries* $\mathcal{A}$ *and all* $\lambda \in \mathbb{N}$*, we have* $|p_{\mathcal{A},H_{2,2}} - p_{\mathcal{A},H_{2,1}}| \leq \mathsf{negl}(\lambda)$.

By construction, $H_{2,2}$ is exactly the simulated world realized by $(\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$ for the selective RNC-IBE definition from Theorem 4.2. Combining the above lemmas and applying the triangle inequality, we conclude that $|p_{\mathcal{A},\mathsf{Hyb}_0} - p_{\mathcal{A},H_{2,2}}| \leq \mathsf{negl}(\lambda)$, which proves the theorem. □

Using Theorem 3.14, Theorem 3.17, and Theorem 3.11, we obtain Theorem 1.5.

8 Rate-1 Strongly Incompressible ABE from RNC-AB-KEM

In this section, we record the ABE analogue of the KEM-based construction used for IBE. The resulting incompressible ABE scheme preserves the underlying predicate class. Assume that for every pair (λ, S), the session-key space of the chosen RNC-AB-KEM coincides with the output space of $\mathsf{IncSKE.Setup}(1^\lambda, 1^S)$; we suppress this dependence in the notation. Let

$$\mathsf{ABKEM} = (\mathsf{ABKEM.Setup}, \mathsf{ABKEM.KeyGen}, \mathsf{ABKEM.Encap}, \mathsf{ABKEM.Decap})$$

be an RNC-AB-KEM, and let $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible SKE scheme such that $\mathsf{IncSKE.Setup}(1^\lambda, 1^S)$ outputs a uniformly random string.

$\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm takes as input the security parameter λ and the upper bound on the state size S. It computes $(\mathsf{abkem.mpk}, \mathsf{abkem.msk}) \leftarrow \mathsf{ABKEM.Setup}(1^\lambda)$ and outputs $\mathsf{mpk} := \mathsf{abkem.mpk}$ and $\mathsf{msk} := \mathsf{abkem.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, f)$: The key generation algorithm takes as input a master secret key $\mathsf{msk} = \mathsf{abkem.msk}$ and a function f. It computes $\mathsf{abkem.sk}_f \leftarrow \mathsf{ABKEM.KeyGen}(\mathsf{abkem.msk}, f)$ and outputs $\mathsf{abkem.sk}_f$.

$\mathsf{Enc}(\mathsf{mpk}, x, m)$: The encryption algorithm takes as input a master public key $\mathsf{mpk} = \mathsf{abkem.mpk}$, an attribute x, and a message m. It generates $(\mathsf{abkem.ct}, \mathsf{seskey}) \leftarrow \mathsf{ABKEM.Encap}(\mathsf{abkem.mpk}, x)$, sets $\mathsf{inc.sk} := \mathsf{seskey}$, computes $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$, and returns $\mathsf{ct} := (\mathsf{abkem.ct}, \mathsf{inc.ct})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key $\mathsf{sk} = \mathsf{abkem.sk}_f$ and a ciphertext $\mathsf{ct} = (\mathsf{abkem.ct}, \mathsf{inc.ct})$. It first computes $\mathsf{inc.sk} \leftarrow \mathsf{ABKEM.Decap}(\mathsf{sk}, \mathsf{abkem.ct})$. It then computes $m \leftarrow \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})$ and returns m.

Correctness. The correctness is immediate from the correctness of the RNC-AB-KEM and the incompressible SKE scheme.
Parameters. The size of a ciphertext for a message m is $|\mathsf{abkem.ct}| + |\mathsf{inc.ct}|$. If the incompressible SKE scheme has rate-1, then $|\mathsf{inc.ct}| = |m|(1 + o(1)) + \mathrm{poly}(\lambda)$. If the RNC-AB-KEM generates ciphertexts whose size is independent of the session key, then $|\mathsf{abkem.ct}| = \mathrm{poly}(\lambda)$. Therefore, the rate of the incompressible ABE scheme is $1 - o(1)$.

Theorem 8.1. *Assuming that* ABKEM *is a secure RNC-AB-KEM scheme for a predicate class and* IncSKE *is an incompressible SKE scheme, the above construction is a secure strongly incompressible ABE scheme for the same predicate class. Moreover, if* ABKEM *remains secure even when the setup randomness is revealed in the response phase, then the above construction is a secure super-strong incompressible ABE scheme for the same predicate class.*

Proof sketch. We prove security by a short hybrid argument.

Hybrid H_0: This is the original strongly incompressible ABE security game.

Hybrid H_1: In this game, the AB-KEM part is switched to simulation mode. Concretely, the first simulator generates the master public key, the second simulator answers all admissible key queries, and the third simulator on input the challenge attribute x^* generates the simulated KEM ciphertext $\mathsf{abkem.ct}$. In the response phase, once the secret key $\mathsf{inc.sk}$ underlying the SKE challenge ciphertext is fixed, the fourth simulator is invoked on input $\mathsf{inc.sk}$ to generate a master secret key consistent with the transcript. For the super-strong variant, the analogous simulator instead outputs setup randomness consistent with the transcript. The indistinguishability of H_0 and H_1 follows directly from the corresponding security of the RNC-AB-KEM.

It remains to show that no PPT adversary can win in H_1 with non-negligible probability. Suppose that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins H_1. We build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the incompressible SKE scheme. In the initialization phase, $\mathcal{B}_1$ forwards the state bound 1^S chosen by $\mathcal{A}_1$ to the SKE challenger. It then uses the first two AB-KEM simulators to generate the master public key and to answer every admissible key query. In the challenge phase, when $\mathcal{A}_1$ outputs $(m_0, m_1, x^*, \mathsf{aux})$, adversary $\mathcal{B}_1$ runs the third AB-KEM simulator on input x^* to obtain the simulated KEM ciphertext $\mathsf{abkem.ct}$, forwards (m_0, m_1, aux) to the SKE challenger, and receives $\mathsf{inc.ct}^*$. It then returns $\mathsf{ct}^* := (\mathsf{abkem.ct}, \mathsf{inc.ct}^*)$ to $\mathcal{A}_1$. After receiving the compressed state st from $\mathcal{A}_1$, adversary $\mathcal{B}_2$ gets $(\mathsf{inc.sk}, \mathsf{aux}, \mathsf{st})$ from the SKE challenger. In the strong case, it invokes the fourth AB-KEM simulator on input $\mathsf{inc.sk}$ to obtain msk and then hands $(\mathsf{mpk}, \mathsf{msk}, \mathsf{aux}, \mathsf{st})$ to $\mathcal{A}_2$. In the super-strong case, it instead invokes the analogous simulator to obtain setup randomness r_{Setup} and then hands $(r_{\mathsf{Setup}}, \mathsf{aux}, \mathsf{st})$ to $\mathcal{A}_2$. Whatever bit $\mathcal{A}_2$ outputs is returned as the guess of $\mathcal{B}_2$. The simulated view is exactly H_1, so any non-negligible advantage of $\mathcal{A}$ would contradict incompressible security of the SKE scheme. □

Combining Theorem 8.1, Theorem 3.7, and Theorem 5.2 together with Theorem 5.1, we obtain the ABE part of Theorem 1.2 for predicate classes with predicate encodings.

Remark 8.2. We also note that a similar approach can be employed to construct incompressible ABE directly from an RNC-ABE scheme and an incompressible SKE scheme. In this approach, during the encryption process with inputs m and attribute x, a secret key $\mathsf{inc.sk}$ of the incompressible SKE is freshly generated and encrypted under the RNC-ABE scheme, producing $\mathsf{abe.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, x, \mathsf{inc.sk})$. Then the message m is encrypted as $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$, and the final ciphertext is $\mathsf{ct} := (\mathsf{abe.ct}, \mathsf{inc.ct})$. The resulting incompressible ABE scheme preserves the same predicate class as the underlying RNC-ABE scheme. Note that the size of the ciphertext ct depends on both $|\mathsf{inc.sk}|$ and

$|\mathsf{inc.ct}|$ because $\mathsf{abe.ct}$ is an encryption of $\mathsf{inc.sk}$. Therefore, instantiating this template with a rate-$\frac{1}{2}$ incompressible SKE such as Theorem 3.8 yields a rate-$\frac{1}{2}$ strongly incompressible ABE scheme for the same predicate class. If the underlying RNC-ABE scheme additionally remains secure when the setup randomness is revealed in the response phase, then the resulting incompressible ABE scheme is super-strong as well.

9 Declaration on the Use of AI Tools

All research ideas and technical results presented in this work were developed by the authors in 2024 and 2025, independently of generative AI tools and prior to their use in preparing this full version. Generative AI tools, including ChatGPT, were used only to assist with proofreading, language editing, improving the clarity and presentation of the manuscript, and identifying potential inconsistencies or points requiring further verification. No research ideas, definitions, constructions, theorems, proofs, or other technical contributions presented in this work originated from generative AI. All suggestions and checks arising from the use of these tools were independently reviewed and verified by the authors, who take full responsibility for the content of this manuscript.

References

[AB09] Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. *Manuscript, July*, 3, 2009. 2

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h) ibe in the standard model. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 553–572. Springer, 2010. 2

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 98–115. Springer, 2010. 2

[AFL16] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Deniable attribute based encryption for branching programs from lwe. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*, pages 299–329. Springer, 2016. 6

[AGM21] Shweta Agrawal, Shafi Goldwasser, and Saleet Mossel. Deniable fully homomorphic encryption from learning with errors. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 641–670. Springer, 2021. 6

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 557–577. Springer, 2014. 2, 6

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22*, pages 591–623. Springer, 2016. 2, 6

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 223–238. Springer, 2004. 2

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Annual International Cryptology Conference*, pages 443–459. Springer, 2004. 2

[BBD+20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 58–87. Springer, 2020. 2, 6, 12

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 440–456. Springer, 2005. 2

[BDD22] Pedro Branco, Nico Döttling, and Jesko Dujmović. Rate-1 incompressible encryption from standard assumptions. In *Theory of Cryptography Conference*, pages 33–69. Springer, 2022. 4, 5, 13

[Bea97] Donald Beaver. Plug and play encryption. In *Annual International Cryptology Conference*, pages 75–89. Springer, 1997. 6

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001. 2

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryptionwithout pairings. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 647–657. IEEE, 2007. 2

[BGK+25] Kaartik Bhushan, Rishab Goyal, Venkata Koppula, Varun Narayanan, Manoj Prabhakaran, and Mahesh Sreekumar Rajasree. Leakage-resilient incompressible cryptography: Constructions and barriers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 201–234. Springer, 2025. 5, 6

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–564. Springer, 2018. 2, 12

[BLW17] Dan Boneh, Kevin Lewi, and David J Wu. Constraining pseudorandom functions privately. In *IACR International Workshop on Public Key Cryptography*, pages 494–524. Springer, 2017. 14

[BNNO11] Rikke Bendlin, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Lower and upper bounds for deniable public-key encryption. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 125–142. Springer, 2011. 6

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007. 2

[CDNO97] Rein Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 90–104. Springer, 1997. 6

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*, pages 287–302. Springer, 2009. 2, 6

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, 1996. 1, 2, 6

[CGKW18] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded abe via bilinear entropy expansion, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 503–534. Springer, 2018. 6

[CGV22] Andrea Coladangelo, Shafi Goldwasser, and Umesh Vazirani. Deniable encryption in a quantum world. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1378–1391, 2022. 6

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system abe in prime-order groups via predicate encodings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 595–624. Springer, 2015. 2, 6, 10, 22, 46

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 255–271. Springer, 2003. 2

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *Theory of Cryptography Conference*, pages 150–168. Springer, 2005. 6

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25:601–639, 2012. 2

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503, 2002. 1, 2

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding: 8th IMA International Conference Cirencester, UK, December 17–19, 2001 Proceedings 8*, pages 360–363. Springer, 2001. 2

[CPR17] Ran Canetti, Oxana Poburinnaya, and Mariana Raykova. Optimal-rate non-committing encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 212–241. Springer, 2017. 2, 6

[CW13] Jie Chen and Hoeteck Wee. Fully,(almost) tightly secure ibe and dual system groups. In *Annual Cryptology Conference*, pages 435–460. Springer, 2013. 6

[DCIO16] Angelo De Caro, Vincenzo Iovino, and Adam O'Neill. Deniable functional encryption. In *Public-Key Cryptography–PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 196–222. Springer, 2016. 6

[DG17a] Nico Döttling and Sanjam Garg. From selective ibe to full ibe and selective hibe. In *Theory of Cryptography Conference*, pages 372–408. Springer, 2017. 2, 15

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Annual international cryptology conference*, pages 537–569. Springer, 2017. 2

[DGO19] Ivan Damgård, Chaya Ganesh, and Claudio Orlandi. Proofs of replicated storage without timing assumptions. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 355–380. Springer, 2019. 5

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings 20*, pages 432–450. Springer, 2000. 6

[Dzi06] Stefan Dziembowski. On Forward-Secure Storage. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, Lecture Notes in Computer Science, pages 251–270, Berlin, Heidelberg, 2006. Springer. 3, 5, 13

[FGC21] Shengyuan Feng, Junqing Gong, and Jie Chen. Master-key kdm-secure abe via predicate encoding. In *IACR International Conference on Public-Key Cryptography*, pages 543–572. Springer, 2021. 6

[GGH+13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. 5

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, August 2013. 16

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. 5

[GGH20] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. Master-key kdm-secure ibe from pairings. In *IACR International Conference on Public-Key Cryptography*, pages 123–152. Springer, 2020. 6

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986. 14

[GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *Theory of Cryptography Conference*, pages 437–456. Springer, 2009. 2

[GKK+25] Rishab Goyal, Fuyuki Kitagawa, Venkata Koppula, Ryo Nishimaki, Mahesh Sreekumar Rajasree, and Takashi Yamakawa. Non-committing identity based encryption: constructions and applications. In *IACR International Conference on Public-Key Cryptography*, pages 158–193. Springer, 2025. 1

[GKR25] Rishab Goyal, Venkata Koppula, and Mahesh Sreekumar Rajasree. A note on adaptive security in hierarchical identity-based encryption. Cryptology ePrint Archive, Paper 2025/291, 2025. 2

[GKRV25] Rishab Goyal, Venkata Koppula, Mahesh Sreekumar Rajasree, and Aman Verma. Incompressible Functional Encryption. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 3, 4, 5

[GLW20] Rachit Garg, George Lu, and Brent Waters. New techniques in replica encodings with client setup. In *Theory of Cryptography Conference*, pages 550–583. Springer, 2020. 5

[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, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98. ACM, 2006. 2

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008. 2

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8*, pages 548–566. Springer, 2002. 2

[GSW21] Rishab Goyal, Ridwan Syed, and Brent Waters. Bounded collusion abe for tms from ibe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 371–402. Springer, 2021. 2

[GW20] Junqing Gong and Hoeteck Wee. Adaptively secure abe for dfa from k-lin and more. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 278–308. Springer, 2020. 2

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. Abe for dfa from k-lin. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 732–764. Springer, 2019. 2, 6

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible Cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, Lecture Notes in Computer Science, pages 700–730, Cham, 2022. Springer International Publishing. 3, 4, 5

[GWZ23] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Multi-instance randomness extraction and security against bounded-storage mass surveillance. In *Theory of Cryptography Conference*, pages 93–122. Springer, 2023. 3, 5

[HKK+24] Goichiro Hanaoka, Shuichi Katsumata, Kei Kimura, Kaoru Takemure, and Shota Yamada. Tighter adaptive ibes and vrfs: Revisiting waters' artificial abort. In *Theory of Cryptography Conference*, pages 124–155. Springer, 2024. 2

[HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In *IACR International Workshop on Public Key Cryptography*, pages 799–822. Springer, 2015. 6

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *International conference on the theory and applications of cryptographic techniques*, pages 466–481. Springer, 2002. 2

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part I 27*, pages 606–636. Springer, 2021. 2, 3, 5, 6

[HOR15] Brett Hemenway, Rafail Ostrovsky, and Alon Rosen. Non-committing encryption from ϕ-hiding. In *Theory of Cryptography Conference*, pages 591–608. Springer, 2015. 6

[HORR15] Brett Hemenway, Rafail Ostrovsky, Silas Richelson, and Alon Rosen. Adaptive security with quasi-optimal rate. In *Theory of Cryptography Conference*, pages 525–541. Springer, 2015. 2, 6

[HPW15] Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In *Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part I 21*, pages 443–469. Springer, 2015. 2

[JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 221–242. Springer, 2000. 6

[LW10] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography Conference*, pages 455–479. Springer, 2010. 2, 6

[LW11] Allison Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 547–567. Springer, 2011. 6

[MW20] Tal Moran and Daniel Wichs. Incompressible Encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, Lecture Notes in Computer Science, pages 494–523, Cham, 2020. Springer International Publishing. 5

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Annual International Cryptology Conference*, pages 111–126. Springer, 2002. 2

[OPW11] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 525–542. Springer, 2011. 6

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 349–366. Springer, 2012. 6

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming prfs, the lwe way. In *IACR International Workshop on Public Key Cryptography*, pages 675–701. Springer, 2018. 14

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*, pages 47–53. Springer, 1985. 2

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005. 2

[SW08] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *International Colloquium on Automata, Languages, and Programming*, pages 560–578. Springer, 2008. 2

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014. 5, 6

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 114–127. Springer, 2005. 2

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Annual international cryptology conference*, pages 619–636. Springer, 2009. 4, 6, 22, 46

[Wat12] Brent Waters. Functional encryption for regular languages. In *Annual Cryptology Conference*, pages 218–235. Springer, 2012. 2

[WC23] Huangting Wu and Sherman SM Chow. Anonymous (hierarchical) identity-based encryption from broader assumptions. In *International Conference on Applied Cryptography and Network Security*, pages 366–395. Springer, 2023. 2

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In *Theory of Cryptography Conference*, pages 616–637. Springer, 2014. 6, 10, 22

[YKT19] Yusuke Yoshida, Fuyuki Kitagawa, and Keisuke Tanaka. Non-committing encryption with quasi-optimal ciphertext-rate based on the ddh problem. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part III 25*, pages 128–158. Springer, 2019. 2, 6

[YKXT20] Yusuke Yoshida, Fuyuki Kitagawa, Keita Xagawa, and Keisuke Tanaka. Non-committing encryption with constant ciphertext expansion from standard assumptions. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 36–65. Springer, 2020. 6, 12

A Construction of Predicate Encoding for Identity Predicate

The following construction was provided in [CGW15]. Let p be a prime and $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p$. The identity predicate P is defined by $P(x,y) = 1 \iff x = y$. The algorithms are as follows.

- $\mathsf{sE}(x, (\mathbf{w}_1, \mathbf{w}_2)) = \mathbf{w}_1 + x\mathbf{w}_2$ where $\mathbf{w}_i \in \mathbb{Z}_p^2$.
- $\mathsf{rE}(y, (\mathbf{w}_1, \mathbf{w}_2)) = \mathbf{w}_1 + y\mathbf{w}_2$ where $\mathbf{w}_i \in \mathbb{Z}_p^2$.
- $\mathsf{kE}(y, \boldsymbol{\alpha}) = \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} \in \mathbb{Z}_p^2$.
- $\mathsf{sD}(x, y, \mathbf{c}) = \mathbf{c}$ where $\mathbf{c} \in \mathbb{Z}_p^2$.
- $\mathsf{rD}(x, y, \mathbf{k}) = \mathbf{k}$ where $\mathbf{k} \in \mathbb{Z}_p^2$.

It is easy to verify the linearity properties. When $x = y$, $\mathsf{sE}(x, (\mathbf{w}_1, \mathbf{w}_2)) = \mathsf{rE}(y, (\mathbf{w}_1, \mathbf{w}_2))$. For the privacy condition, when $x \neq y$, the map $(\mathbf{w}_1, \mathbf{w}_2) \mapsto (\mathbf{w}_1 + x\mathbf{w}_2, \mathbf{w}_1 + y\mathbf{w}_2)$ is invertible, so the pair $(\mathbf{w}_1 + x\mathbf{w}_2, \mathbf{w}_1 + y\mathbf{w}_2)$ is jointly uniform over $(\mathbb{Z}_p^2)^2$ and hence independent of $\boldsymbol{\alpha}$.

B Receiver Non-Committing IB-KEM and IBE from Bilinear Groups

In this section, we present an adaptively secure receiver non-committing identity-based key encapsulation mechanism (RNC-IB-KEM) using the concepts of dual system encryption [Wat09].

B.1 Construction

Our construction is as follows. Let $\mathsf{HC} : \mathbb{G}_T \times \{0,1\}^{\log(p)} \to \{0,1\}$ denote a 1-bit randomness extractor over a group element, and let $\ell := \ell(\lambda)$ be a polynomial in λ.

$\mathsf{Gen}(1^\lambda)$:

- Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.
- Output $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$. We assume that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$ is included in pp and omit it from the notation.

$\mathsf{Setup}(\mathsf{pp})$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$.
- Generate $h \leftarrow \{0,1\}^{\log(p)}$.
- Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2$ for all $i \in [\ell]$.
- Output $\mathsf{mpk} := \left(\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_{i\in[\ell]}, h \right)$ and $\mathsf{msk} := \{ \mathbf{k}_i \}_{i\in[\ell]}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id} \in \mathbb{Z}_p)$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top\mathbf{b}]_2, [\mathbf{W}_2^\top\mathbf{b}]_2)$ and $\mathsf{msk} = \{\, \mathbf{k}_i \,\}_{i\in[\ell]}$.
- Generate $s_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Output $\mathsf{sk}_{\mathsf{id}} := \{\, ([s_i\mathbf{b}]_2, [\mathbf{k}_i + s_i(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b}]_2) \,\}_{i\in[\ell]}$.

$\mathsf{Encap}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id})$:

- Parse $\mathsf{pp} = ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top\mathbf{b}]_2, [\mathbf{W}_2^\top\mathbf{b}]_2)$ and $\mathsf{mpk} = \Big(\{\, [\mathbf{a}^\top\mathbf{k}_i]_T \,\}_{i\in[\ell]}, h\Big)$.
- Generate $r \leftarrow \mathbb{Z}_p$.
- Output $\mathsf{ct} := ([r\mathbf{a}]_1, [r(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)\mathbf{a}]_1)$ and $\mathsf{seskey} := \{\, \mathsf{HC}([r\mathbf{a}^\top\mathbf{k}_i]_T, h) \,\}_{i\in[\ell]}$.

$\mathsf{Decap}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$:

- Parse $\mathsf{mpk} = \Big(\{\, [\mathbf{a}^\top\mathbf{k}_i]_T \,\}_{i\in[\ell]}, h\Big)$.
- Parse $\mathsf{sk}_{\mathsf{id}} = \{\, ([\mathbf{d}_i]_2, [\mathbf{d}_i']_2) \,\}_{i\in[\ell]}$ and $\mathsf{ct} = ([\mathbf{c}]_1, [\mathbf{c}']_1)$.
- Output $\mathsf{seskey} := \{\, \mathsf{HC}\big(e([\mathbf{c}]_1^\top, [\mathbf{d}_i']_2)/e([\mathbf{c}']_1^\top, [\mathbf{d}_i]_2), h\big) \,\}_{i\in[\ell]}$.

$\mathsf{Sim}_1(1^\lambda)$:

- Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.
- Generate $k_{i,1} \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$, and generate $h \leftarrow \{0,1\}^{\log(p)}$.
- Output $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top\mathbf{b}]_2, [\mathbf{W}_2^\top\mathbf{b}]_2)$, $\mathsf{mpk} := \Big(\{\, [k_{i,1}]_T \,\}_{i\in[\ell]}, h\Big)$, and $\mathsf{st}_1 := \Big(\{\, k_{i,1} \,\}_{i\in[\ell]}, \mathbf{a}, h\Big)$.

$\mathsf{Sim}_2(\mathsf{st}_1, \mathsf{id})$:

- Parse $\mathsf{st}_1 = \Big(\{\, k_{i,1} \,\}_{i\in[\ell]}, \mathbf{a}, h\Big)$.
- Generate $s_i, w_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Output $\mathsf{sk}_{\mathsf{id}} := \{\, ([s_i\mathbf{b}]_2, [\frac{k_{i,1}}{|\mathbf{a}|^2}\cdot\mathbf{a} + s_i(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b} + w_i\mathbf{a}^\perp]_2) \,\}_{i\in[\ell]}$ and $\mathsf{st}_2 := \mathsf{st}_1$.

$\mathsf{Sim}_3(\mathsf{st}_2, \mathsf{id}^*)$:

- Parse $\mathsf{st}_2 = \Big(\{\, k_{i,1} \,\}_{i\in[\ell]}, \mathbf{a}, h\Big)$.
- Generate $u_1 \leftarrow \mathbb{Z}_p$, $u_2 \leftarrow \mathbb{Z}_p^*$, and $x_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
- Set $\mathbf{u} := u_1\mathbf{a} + u_2\mathbf{a}^\perp$.
- Output $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^*\cdot\mathbf{W}_2)\mathbf{u}]_1)$, $\mathsf{seskey} := \{\, \mathsf{HC}([x_i]_T, h) \,\}_{i\in[\ell]}$, and $\mathsf{st}_3 := \Big(\mathsf{st}_2, u_1, u_2, \{\, x_i \,\}_{i\in[\ell]}\Big)$.

$\mathsf{Sim}_4(\mathsf{st}_3, \mathsf{seskey} \in \{0,1\}^\ell)$:

- Parse $\mathsf{st}_3 = \Big(\mathsf{st}_2, u_1, u_2, \{\, x_i \,\}_{i\in[\ell]}\Big)$, where $\mathsf{st}_2 = \Big(\{\, k_{i,1} \,\}_{i\in[\ell]}, \mathbf{a}, h\Big)$.
- Set $k_{i,2} := (x_i - u_1 k_{i,1})/u_2$ for all $i \in [\ell]$.
- Output $\mathsf{msk} := \{\, \frac{k_{i,1}}{|\mathbf{a}|^2}\cdot\mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2}\cdot\mathbf{a}^\perp \,\}_{i\in[\ell]}$.

The input seskey to Sim_4 is included only to match the abstract interface from the preliminaries; in this construction, the simulator state already determines the value needed to reconstruct msk.

Remark B.1. The randomness used in the setup algorithm consists of h, which is part of the master public key, and the set $\{\mathbf{k}_i\}_{i\in[\ell]}$, which constitutes the entire master secret key. Since the challenger outputs h in the initialization phase and reveals the master secret key only in the response phase, the adversary eventually obtains all the randomness used in the setup algorithm, but never the randomness used by the Encap algorithm.

Parameters. The size of a ciphertext is $\mathrm{poly}(\lambda)$, i.e., it is independent of ℓ, whereas the sizes of the master public key, master secret key, and user secret keys depend on ℓ.

Correctness. For correctly generated $\mathsf{sk}_{\mathsf{id}} := \{([s_i\mathbf{b}]_2, [\mathbf{k}_i + s_i(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b}]_2)\}_{i\in[\ell]}$ and $\mathsf{ct} := ([r\mathbf{a}]_1, [r(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)\mathbf{a}]_1)$, we have

$$\begin{aligned}\frac{e([r\mathbf{a}]_1^\top, [\mathbf{k}_i + s_i(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b}]_2)}{e([r(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)\mathbf{a}]_1^\top, [s_i\mathbf{b}]_2)} &= \frac{[r\mathbf{a}^\top\mathbf{k}_i + rs_i\mathbf{a}^\top(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b}]_T}{[rs_i\mathbf{a}^\top(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b}]_T} \\ &= [r\mathbf{a}^\top\mathbf{k}_i]_T\end{aligned}$$

for all $i \in [\ell]$. Since $\mathsf{HC}(\cdot,\cdot)$ is deterministic, correctness follows immediately.

Theorem B.2. *Assuming the hardness of* SXDH*, the above scheme is an adaptively secure receiver non-committing identity-based key encapsulation mechanism.*

Proof. Consider an adversary $\mathcal{A}$ against the RNC-IB-KEM security game. Without loss of generality, we may assume that $\mathcal{A}$ never asks for a secret key corresponding to the eventual challenge identity id^*. Indeed, any pre-challenge query for id^* causes the real and simulated experiments to abort, while any post-challenge query for id^* is answered with $\perp$. Let q be the total number of admissible key queries made across the pre-challenge and post-challenge query phases, indexed as $\mathsf{id}^1, \ldots, \mathsf{id}^q$ in the order they are asked.

Hybrid Hyb_0**:** This is the real experiment.

1. The challenger generates pp, mpk, msk as follows.
   - Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
   - Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2$ for all $i \in [\ell]$ and $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top\mathbf{b}]_2, [\mathbf{W}_2^\top\mathbf{b}]_2)$, $\mathsf{mpk} := \left(\{[\mathbf{a}^\top\mathbf{k}_i]_T\}_{i\in[\ell]}, h\right)$, and $\mathsf{msk} := \{\mathbf{k}_i\}_{i\in[\ell]}$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. **Pre-Challenge Query Phase:** $\mathcal{A}$ has access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-th admissible query $\mathsf{id}^j \in \mathbb{Z}_p$, return $\mathsf{sk}_{\mathsf{id}^j}$ generated as follows.
   - Generate $s_i^j \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$.
   - Set $\mathsf{sk}_{\mathsf{id}^j} := \{([s_i^j\mathbf{b}]_2, [\mathbf{k}_i + s_i^j(\mathbf{W}_1 + \mathsf{id}^j\cdot\mathbf{W}_2)^\top\mathbf{b}]_2)\}_{i\in[\ell]}$.
3. **Challenge Phase:** $\mathcal{A}$ outputs $\mathsf{id}^* \in \mathbb{Z}_p$. If one of the previously asked queries equals id^*, then the challenger aborts. Otherwise, the challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $r \leftarrow \mathbb{Z}_p$.
   - Set $\mathsf{ct}^* := ([r\mathbf{a}]_1, [r(\mathbf{W}_1 + \mathsf{id}^*\cdot\mathbf{W}_2)\mathbf{a}]_1)$ and $\mathsf{seskey}^* := \{\mathsf{HC}([r\mathbf{a}^\top\mathbf{k}_i]_T, h)\}_{i\in[\ell]}$.

   The challenger sends $(\mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.

4. **Post-Challenge Query Phase:** $\mathcal{A}$ continues to have access to $O_{\mathsf{UserKeyGen}}$. For each new query id, the challenger returns $\perp$ if $\mathsf{id} = \mathsf{id}^*$ and otherwise answers exactly as in the pre-challenge query phase. By our w.l.o.g. assumption, we only need to consider the admissible queries id^j.
5. **Response Phase:** The challenger sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs $d \in \{0,1\}$.

Using a sequence of hybrid experiments, we transform the real experiment into the simulated experiment. We denote the probability that $\mathcal{A}$ outputs 0 in hybrid Hyb_i by $p_{\mathcal{A},H_i}$.

Below, we assume that $\mathbf{a}$ and $\mathbf{b}^\perp$ are linearly independent and that $\mathbf{a}^\perp$ and $\mathbf{b}$ are linearly independent. This holds with overwhelming probability over the choice of $\mathbf{a}$ and $\mathbf{b}$.

Changing the challenge ciphertext into semi-functional mode.

Hybrid Hyb_1: This is the same as Hyb_0 except that $(\mathsf{ct}^*, \mathsf{seskey}^*)$ is generated as $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_i]_T, h) \}_{i\in[\ell]}$, where $\mathbf{u} \leftarrow \mathbb{Z}_p^2$.

We have $|p_{\mathcal{A},H_1} - p_{\mathcal{A},H_0}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_1$.

Lemma B.3. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_1} - p_{\mathcal{A},H_0}| \le p_{\mathcal{B},\mathsf{DDH}}$.*

We define $\mathsf{Hyb}_{1,0,5}$ to be Hyb_1. For each $i \in [q]$, we now transform $\mathsf{Hyb}_{1,i-1,5}$ into $\mathsf{Hyb}_{1,i,5}$ through the intermediate hybrids $\mathsf{Hyb}_{1,i,1}, \ldots, \mathsf{Hyb}_{1,i,4}$.

Changing the user secret keys into semi-functional mode. We change the user secret keys in both query phases into semi-functional mode one admissible query at a time, using hybrids $\mathsf{Hyb}_{1,i,1}, \ldots, \mathsf{Hyb}_{1,i,5}$ for $i \in [q]$. Throughout this part of the proof, the index j refers to the position of the admissible query in the global order across the pre-challenge and post-challenge query phases. Define $\mathbf{W}_0 := \mathbf{b}^\perp (\mathbf{a}^\perp)^\top / ((\mathbf{b}^\perp)^\top \mathbf{a}^\perp)$.

Hybrid $\mathsf{Hyb}_{1,i,1}$: This is the same as $\mathsf{Hyb}_{1,i-1,5}$ except that $O_{\mathsf{UserKeyGen}}$ behaves as follows on the j-th admissible query $\mathsf{id}^j \in \mathbb{Z}_p$.

- If $j < i$, return $\mathsf{sk}_{\mathsf{id}^j}$ generated as follows.
  - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.
  - Set $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\mathbf{k}_d + s_d^j (\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_{d\in[\ell]}$.
- If $j = i$, return $\mathsf{sk}_{\mathsf{id}^i}$ generated as follows.
  - Generate $\mathbf{v}_d^i \leftarrow \mathbb{Z}_p^2$ for all $d \in [\ell]$.
  - Set $\mathsf{sk}_{\mathsf{id}^i} := \{ ([\mathbf{v}_d^i]_2, [\mathbf{k}_d + (\mathbf{W}_1 + \mathsf{id}^i \cdot \mathbf{W}_2)^\top \mathbf{v}_d^i]_2) \}_{d\in[\ell]}$.
- If $j > i$, return $\mathsf{sk}_{\mathsf{id}^j}$ generated exactly as in Hyb_1.

For every $i \in [q]$, we have $|p_{\mathcal{A},H_{1,i,1}} - p_{\mathcal{A},H_{1,i-1,5}}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$. Since $\mathsf{Hyb}_{1,0,5} = \mathsf{Hyb}_1$, this includes the first transition from Hyb_1 to $\mathsf{Hyb}_{1,1,1}$.

Lemma B.4. *For all PPT adversaries $\mathcal{A}$ and all $i \in [q]$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_{1,i,1}} - p_{\mathcal{A},H_{1,i-1,5}}| \le \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Hybrid $\mathsf{Hyb}_{1,i,2}$: This is the same as $\mathsf{Hyb}_{1,i,1}$ except that we generate $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$ and $w_1, w_2 \leftarrow \mathbb{Z}_p$, and set $\mathbf{W}_1 := \hat{\mathbf{W}}_1 + w_1 \mathbf{W}_0$ and $\mathbf{W}_2 := \hat{\mathbf{W}}_2 + w_2 \mathbf{W}_0$.

We have $|p_{\mathcal{A},H_{1,i,2}} - p_{\mathcal{A},H_{1,i,1}}| = 0$ since the distributions of $\mathbf{W}_1$ and $\mathbf{W}_2$ do not change.

Lemma B.5. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_{1,i,2}} - p_{\mathcal{A},H_{1,i,1}}| = 0$.*

We now show that w_1 and w_2 appear only in ct^* and $\mathsf{sk}_{\mathsf{id}^i}$. First, $[\mathbf{W}_\alpha \mathbf{a}]_1 = [\hat{\mathbf{W}}_\alpha \mathbf{a}]_1$ and $[\mathbf{W}_\alpha^\top \mathbf{b}]_2 = [\hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2$ for $\alpha \in \{1,2\}$. For ct^*, write $\mathbf{u} = r\mathbf{a} + r'\mathbf{b}^\perp$. Then

$$\begin{aligned}\mathsf{ct}^* &= ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1) \\ &= ([\mathbf{u}]_1, [(\hat{\mathbf{W}}_1 + \mathsf{id}^* \cdot \hat{\mathbf{W}}_2)\mathbf{u} + r'(w_1 + \mathsf{id}^* \cdot w_2)\mathbf{b}^\perp]_1).\end{aligned}$$

For $\mathsf{sk}_{\mathsf{id}^i}$, write $\mathbf{v}_d^i = s_d^i \mathbf{b} + t_d^i \mathbf{a}^\perp$. Then

$$\begin{aligned}\mathsf{sk}_{\mathsf{id}^i} &= \{ \left([\mathbf{v}_d^i]_2,\ [\mathbf{k}_d + (\mathbf{W}_1 + \mathsf{id}^i \cdot \mathbf{W}_2)^\top \mathbf{v}_d^i]_2 \right) \}_{d\in[\ell]} \\ &= \{ \left([\mathbf{v}_d^i]_2,\ [\mathbf{k}_d + (\hat{\mathbf{W}}_1 + \mathsf{id}^i \cdot \hat{\mathbf{W}}_2)^\top \mathbf{v}_d^i + t_d^i(w_1 + \mathsf{id}^i \cdot w_2)\mathbf{a}^\perp]_2 \right) \}_{d\in[\ell]}.\end{aligned}$$

Moreover, for $j \neq i$, we can write

$$\mathsf{sk}_{\mathsf{id}^j} = \begin{cases} \{ \left([s_d^j \mathbf{b}]_2,\ [\mathbf{k}_d + s_d^j(\hat{\mathbf{W}}_1 + \mathsf{id}^j \cdot \hat{\mathbf{W}}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2 \right) \}_{d\in[\ell]} & \text{if } j < i, \\ \{ \left([s_d^j \mathbf{b}]_2,\ [\mathbf{k}_d + s_d^j(\hat{\mathbf{W}}_1 + \mathsf{id}^j \cdot \hat{\mathbf{W}}_2)^\top \mathbf{b}]_2 \right) \}_{d\in[\ell]} & \text{if } j > i. \end{cases}$$

Hybrid $\mathsf{Hyb}_{1,i,3}$**:** This is the same as $\mathsf{Hyb}_{1,i,2}$ except that for the i-th admissible query id^i, $O_{\mathsf{UserKeyGen}}$ returns $\mathsf{sk}_{\mathsf{id}^i} := \{ ([\mathbf{v}_d^i]_2, [\mathbf{k}_d + (\mathbf{W}_1 + \mathsf{id}^i \cdot \mathbf{W}_2)^\top \mathbf{v}_d^i + w_d^i \mathbf{a}^\perp]_2) \}_{d\in[\ell]}$, where $w_d^i \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

Let $\gamma^* := w_1 + \mathsf{id}^* \cdot w_2$ and $\gamma_i := w_1 + \mathsf{id}^i \cdot w_2$. Since $\mathsf{id}^i \neq \mathsf{id}^*$, the map $(w_1, w_2) \mapsto (\gamma^*, \gamma_i)$ is bijective over $\mathbb{Z}_p^2$, so (γ^*, γ_i) is uniform over $\mathbb{Z}_p^2$. Conditioned on the event $\gamma_i \neq 0$, which fails with probability exactly $1/p$, the tuple $(t_d^i \gamma_i)_{d\in[\ell]}$ is distributed identically to a uniformly random tuple $(w_d^i)_{d\in[\ell]}$ because the coefficients t_d^i are independent and uniformly random. Moreover, γ^* remains uniform and independent of that tuple. Therefore the additional semi-functional terms appearing in ct^* and $\mathsf{sk}_{\mathsf{id}^i}$ have the same distribution in $\mathsf{Hyb}_{1,i,2}$ and $\mathsf{Hyb}_{1,i,3}$, except with probability $1/p$. Hence $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$.

Lemma B.6. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$.*

Hybrid $\mathsf{Hyb}_{1,i,4}$**:** This is the same as $\mathsf{Hyb}_{1,i,3}$ except that we undo the change between $\mathsf{Hyb}_{1,i,1}$ and $\mathsf{Hyb}_{1,i,2}$, i.e., we generate $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.

We have $|p_{\mathcal{A},H_{1,i,4}} - p_{\mathcal{A},H_{1,i,3}}| = 0$.

Lemma B.7. *For all PPT adversaries $\mathcal{A}$, we have $|p_{\mathcal{A},H_{1,i,4}} - p_{\mathcal{A},H_{1,i,3}}| = 0$.*

Hybrid $\mathsf{Hyb}_{1,i,5}$**:** This is the same as $\mathsf{Hyb}_{1,i,4}$ except that for the i-th admissible query id^i, $O_{\mathsf{UserKeyGen}}$ returns $\mathsf{sk}_{\mathsf{id}^i} := \{ ([s_d^i \mathbf{b}]_2, [\mathbf{k}_d + s_d^i(\mathbf{W}_1 + \mathsf{id}^i \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^i \mathbf{a}^\perp]_2) \}_{d\in[\ell]}$, where $s_d^i, w_d^i \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

We have $|p_{\mathcal{A},H_{1,i,5}} - p_{\mathcal{A},H_{1,i,4}}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$.

Lemma B.8. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $|p_{\mathcal{A},H_{1,i,5}} - p_{\mathcal{A},H_{1,i,4}}| \leq \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Final steps towards non-committing mode.

Hybrid Hyb_2**:** Iterating the above five transitions for $i = 1, \ldots, q$ yields $\mathsf{Hyb}_2 := \mathsf{Hyb}_{1,q,5}$. Concretely, it is the following experiment.

1. The challenger generates $\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}$ exactly as in Hyb_0.

2. **Pre-Challenge Query Phase:** Every admissible query id^j is answered by generating $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$ and returning $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\mathbf{k}_d + s_d^j(\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_{d \in [\ell]}$.
3. **Challenge Phase:** On challenge identity id^*, the challenger generates $\mathbf{u} \leftarrow \mathbb{Z}_p^2$ and returns $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_i]_T, h) \}_{i \in [\ell]}$.
4. **Post-Challenge Query Phase:** Every admissible query is answered exactly as in the pre-challenge query phase, while the query id^* is answered with $\perp$.
5. **Response Phase:** The challenger sends msk to $\mathcal{A}$, and then $\mathcal{A}$ outputs $d \in \{0,1\}$.

Hybrid Hyb_3**:** This is the same as Hyb_2 except that we generate $k_{i,1}, k_{i,2} \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$ and set $\mathbf{k}_i := \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp$. In particular, $\mathsf{mpk} = \left(\{ [k_{i,1}]_T \}_{i \in [\ell]}, h \right)$.

This is only a conceptual change, so $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.

Lemma B.9. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.*

Hybrid Hyb_4**:** This is the same as Hyb_3 except that every admissible query id^j in either query phase is answered with $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + s_d^j(\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_{d \in [\ell]}$, where $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$.

We have $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = 0$ since $\frac{k_{d,2}}{|\mathbf{a}^\perp|^2} + w_d^j$ and w_d^j are identically distributed for uniformly random w_d^j.

Lemma B.10. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = 0$.*

Hybrid Hyb_5**:** This is the same as Hyb_4 except that we generate $u_1 \leftarrow \mathbb{Z}_p$ and $u_2 \leftarrow \mathbb{Z}_p^*$, and set $\mathbf{u} := u_1 \mathbf{a} + u_2 \mathbf{a}^\perp$.

The only difference between Hyb_4 and Hyb_5 is that Hyb_5 conditions on the event that the coefficient of $\mathbf{a}^\perp$ in $\mathbf{u}$ is nonzero. This event fails with probability exactly $1/p$, and therefore $|p_{\mathcal{A},H_5} - p_{\mathcal{A},H_4}| = \mathsf{negl}(\lambda)$.

Lemma B.11. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_5} - p_{\mathcal{A},H_4}| = \mathsf{negl}(\lambda)$.*

Hybrid Hyb_6**:** This is the same as Hyb_5 except that we generate $x_i \leftarrow \mathbb{Z}_p$ and set $k_{i,2} := (x_i - u_1 k_{i,1})/u_2$ for all $i \in [\ell]$. By this change, $\mathsf{seskey}^* = \{ \mathsf{HC}([x_i]_T, h) \}_{i \in [\ell]}$.

We have $|p_{\mathcal{A},H_6} - p_{\mathcal{A},H_5}| = 0$ since $u_2 \in \mathbb{Z}_p^*$ and hence $k_{i,2}$ remains uniformly distributed when x_i is chosen uniformly at random.

Lemma B.12. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_6} - p_{\mathcal{A},H_5}| = 0$.*

Hybrid Hyb_7**:** This is the same as Hyb_6 except that we defer the generation of $k_{i,2}$, and hence of msk, to the response phase. The detailed description is as follows.

1. The challenger generates pp and mpk as follows.
   - Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
   - Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^2$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2 \times 2}$.
   - Generate $k_{i,1} \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$, and generate $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1 \mathbf{a}]_1, [\mathbf{W}_2 \mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$ and $\mathsf{mpk} := \left(\{ [k_{i,1}]_T \}_{i \in [\ell]}, h \right)$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. **Pre-Challenge Query Phase:** For every admissible query $\mathsf{id}^j \in \mathbb{Z}_p$, the challenger generates $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$ for all $d \in [\ell]$ and returns $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + s_d^j(\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_{d \in [\ell]}$.

3. **Challenge Phase:** $\mathcal{A}$ outputs $\mathsf{id}^* \in \mathbb{Z}_p$. The challenger generates $u_1 \leftarrow \mathbb{Z}_p$, $u_2 \leftarrow \mathbb{Z}_p^*$, and $x_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$, sets $\mathbf{u} := u_1\mathbf{a} + u_2\mathbf{a}^\perp$, and returns $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{\, \mathsf{HC}([x_i]_T, h)\,\}_{i\in[\ell]}$.
4. **Post-Challenge Query Phase:** The challenger answers every admissible query exactly as in the pre-challenge query phase and returns $\perp$ on the query id^*.
5. **Response Phase:** The challenger sets $k_{i,2} := (x_i - u_1 k_{i,1})/u_2$ for all $i \in [\ell]$, defines $\mathsf{msk} := \{\, \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp \,\}_{i\in[\ell]}$, sends msk to $\mathcal{A}$, and finally $\mathcal{A}$ outputs $d \in \{0,1\}$.

The view of $\mathcal{A}$ in Hyb_7 is identical to its view in Hyb_6, and therefore $|p_{\mathcal{A},H_7} - p_{\mathcal{A},H_6}| = 0$.

Lemma B.13. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have $|p_{\mathcal{A},H_7} - p_{\mathcal{A},H_6}| = 0$.*

Hybrid Hyb_7 is exactly the simulated world realized by the simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$. Combining the hybrid transitions, we obtain

$$|p_{\mathcal{A},H_0} - p_{\mathcal{A},H_7}| \le p_{\mathcal{B}_0,\mathsf{DDH}} + \sum_{i=1}^{q} \ell \cdot p_{\mathcal{B}_{i,1},\mathsf{DDH}} + \sum_{i=1}^{q} \ell \cdot p_{\mathcal{B}_{i,5},\mathsf{DDH}} + \mathsf{negl}(\lambda)$$

for suitable PPT adversaries $\mathcal{B}_0, \{\, \mathcal{B}_{i,1}\,\}_{i\in[q]}, \{\, \mathcal{B}_{i,5}\,\}_{i\in[q]}$. Since ℓ and q are polynomially bounded and the SXDH assumption gives DDH hardness in both $\mathbb{G}_1$ and $\mathbb{G}_2$, the right-hand side is negligible. Therefore the theorem follows. □

Combining Theorem 4.5 and Theorem B.2, we obtain the following.

Theorem B.14. *Assuming the hardness of* SXDH*, there exist adaptively secure RNC-IBE schemes.*

C Rate-1 Strongly Incompressible IBE from RNC-IB-KEM

In this section, we show a construction for strongly incompressible IBE using an RNC-IB-KEM and an incompressible SKE scheme. The ABE analogue for the same predicate class appears in Section 8. Assume that for every pair (λ, S), the session-key space of the chosen RNC-IB-KEM coincides with the output space of $\mathsf{IncSKE.Setup}(1^\lambda, 1^S)$; we suppress this dependence in the notation. Let $\mathsf{IBKEM} = (\mathsf{IBKEM.Setup}, \mathsf{IBKEM.KeyGen}, \mathsf{IBKEM.Encap}, \mathsf{IBKEM.Decap})$ be an RNC-IB-KEM and $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible SKE scheme such that $\mathsf{IncSKE.Setup}(1^\lambda, 1^S)$ outputs a uniformly random string.

$\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm takes as input the security parameter λ and the upper bound on the state size S. It computes $(\mathsf{ibkem.mpk}, \mathsf{ibkem.msk}) \leftarrow \mathsf{IBKEM.Setup}(1^\lambda)$ and outputs $\mathsf{mpk} := \mathsf{ibkem.mpk}$ and $\mathsf{msk} := \mathsf{ibkem.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: The key generation algorithm takes as input a master secret key $\mathsf{msk} = \mathsf{ibkem.msk}$ and an identity id and computes $\mathsf{ibkem.sk}_{\mathsf{id}} \leftarrow \mathsf{IBKEM.KeyGen}(\mathsf{ibkem.msk}, \mathsf{id})$. It outputs $\mathsf{ibkem.sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$: The encryption algorithm takes as input a master public key $\mathsf{mpk} = \mathsf{ibkem.mpk}$, an identity id, and a message m. It generates $(\mathsf{ibkem.ct}, \mathsf{seskey}) \leftarrow \mathsf{IBKEM.Encap}(\mathsf{ibkem.mpk}, \mathsf{id})$, sets $\mathsf{inc.sk} := \mathsf{seskey}$, computes $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$, and returns $\mathsf{ct} := (\mathsf{ibkem.ct}, \mathsf{inc.ct})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key $\mathsf{sk} = \mathsf{ibkem.sk}_{\mathsf{id}}$ and a ciphertext $\mathsf{ct} = (\mathsf{ibkem.ct}, \mathsf{inc.ct})$. It first computes $\mathsf{inc.sk} \leftarrow \mathsf{IBKEM.Decap}(\mathsf{sk}, \mathsf{ibkem.ct})$. It then computes $m \leftarrow \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})$ and returns m.

Correctness. The correctness is straightforward from the correctness of the RNC-IB-KEM scheme and the incompressible SKE scheme.

Parameters. The size of a ciphertext for a message m is $|\mathsf{ibkem.ct}| + |\mathsf{inc.ct}|$. If the incompressible SKE scheme has rate-1 (see Theorem 3.7), then $|\mathsf{inc.ct}| = |m|(1 + o(1)) + \mathrm{poly}(\lambda)$. If the RNC-IB-KEM generates ciphertexts whose size is independent of the session key, then $|\mathsf{ibkem.ct}| = \mathrm{poly}(\lambda)$. Therefore, the rate of the incompressible IBE scheme is $1 - o(1)$.

Theorem C.1. *Assuming that* IBKEM *is a secure RNC-IB-KEM scheme and* IncSKE *is an incompressible SKE scheme, the above construction is a secure strongly incompressible IBE scheme. Moreover, if* IBKEM *remains secure even when the setup randomness is revealed in the response phase, then the above construction is a secure super-strong incompressible IBE scheme.*

Proof sketch. We prove security by a short hybrid argument.

Hybrid H_0**:** This is the original strongly incompressible IBE security game.

Hybrid H_1**:** In this game, the IB-KEM part is switched to simulation mode. More precisely, the first simulator generates the master public key $\mathsf{ibkem.mpk}$, the second simulator answers all key queries, and the third simulator on input id^* produces the simulated KEM ciphertext $\mathsf{ibkem.ct}$. In the response phase, once the secret key $\mathsf{inc.sk}$ underlying the SKE challenge ciphertext is fixed, the fourth simulator is invoked on input $\mathsf{inc.sk}$ to generate a master secret key consistent with the transcript. For the super-strong variant, the analogous simulator instead outputs setup randomness consistent with the transcript. The indistinguishability of H_0 and H_1 follows directly from the corresponding security of the RNC-IB-KEM.

It remains to show that no PPT adversary can win in H_1 with non-negligible probability. Suppose that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins H_1. We build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the incompressible SKE scheme. In the initialization phase, $\mathcal{B}_1$ forwards the state bound 1^S chosen by $\mathcal{A}_1$ to the SKE challenger. It then uses the first two IB-KEM simulators to generate the master public key and to answer every identity query. In the challenge phase, when $\mathcal{A}_1$ outputs $(m_0, m_1, \mathsf{id}^*, \mathsf{aux})$, adversary $\mathcal{B}_1$ runs the third IB-KEM simulator on input id^* to obtain the simulated KEM ciphertext $\mathsf{ibkem.ct}$, forwards (m_0, m_1, aux) to the SKE challenger, and receives $\mathsf{inc.ct}^*$. It then returns $\mathsf{ct}^* := (\mathsf{ibkem.ct}, \mathsf{inc.ct}^*)$ to $\mathcal{A}_1$. After receiving the compressed state st from $\mathcal{A}_1$, adversary $\mathcal{B}_2$ gets $(\mathsf{inc.sk}, \mathsf{aux}, \mathsf{st})$ from the SKE challenger. In the strong case, it invokes the fourth IB-KEM simulator on input $\mathsf{inc.sk}$ to obtain $\mathsf{ibkem.msk}$ and then hands $(\mathsf{mpk}, \mathsf{ibkem.msk}, \mathsf{aux}, \mathsf{st})$ to $\mathcal{A}_2$. In the super-strong case, it instead invokes the analogous simulator to obtain setup randomness r_{Setup} and then hands $(r_{\mathsf{Setup}}, \mathsf{aux}, \mathsf{st})$ to $\mathcal{A}_2$. Whatever bit $\mathcal{A}_2$ outputs is returned as the guess of $\mathcal{B}_2$. The simulated view is exactly H_1, so any non-negligible advantage of $\mathcal{A}$ would contradict incompressible security of the SKE scheme. □

Combining Theorem C.1, Theorem 3.7, and Theorem B.2 together with Theorem B.1, we obtain the IBE part of Theorem 1.2.

Remark C.2. We highlight that a similar approach can be employed to construct incompressible IBE from an RNC-IBE scheme and an incompressible SKE scheme. In this approach, during the encryption process with inputs m and identity id, a secret key $\mathsf{inc.sk}$ of the incompressible SKE is freshly generated and encrypted, producing $\mathsf{ibe.ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{inc.sk})$. Then the message m is encrypted as $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$, and the final ciphertext is $\mathsf{ct} := (\mathsf{ibe.ct}, \mathsf{inc.ct})$. Note that the size of the ciphertext ct depends on both $|\mathsf{inc.sk}|$ and $|\mathsf{inc.ct}|$ because $\mathsf{ibe.ct}$ is an encryption of $\mathsf{inc.sk}$. Since the transformation from RNC-IBE to incompressible IBE preserves the setup-randomness-exposure variant, combining the stronger form of Theorem 6.2 with Theorem 3.8 gives Theorem 1.4.