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

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

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

February 6, 2026

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 leads 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 $\mathsf{DDH}, \mathsf{LWE}$. This leads to a relaxed incompressible IBE scheme with strong security from 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

*A preliminary version of this paper was accepted at PKC 2025.

†Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

‡This work is done while the author was a Post-Doctoral Fellow at IIT Delhi, India.

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 ciphertext without knowledge of the underlying message. 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 ideal-world. Today, we have several constructions of NCE, from a wide range of assumptions [CDSMW09, HORR15, CPR17, YKT19, BBD+20] as well 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 which 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], adaptive 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 to go 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] is a further generalization of IBE where secret keys are associated with predicates and users can encryption 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, if it does not have a secret key for f such that $f(x^*) = 1$, then the adversary cannot decrypt an encryption for x^*.

In the non-commmitting setting, identity-based encryption (and more generally, attribute-based encryption) was introduced by Hiroka, Morimae, Nishimaki and Yamakawa [HMNY21] in the context of attribute-based quantum encryption with certified deletion. 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 $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ 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 challenge messages 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, when the adversary sends the challenge identity id^* and the challenge message m^*, the simulator must first produce ct^* using just id^*. Later, when it receives m^*, it must produce a master secret key msk. The adversary finally receives ct^* and msk, and must distinguish 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 understanding with respect to RNC-IBE and RNC-ABE is very limited. Hiroka *et al.* [HMNY21] gave a construction

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

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 win the following game (with non-negligible probability): 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 $(|m| + S + \mathrm{poly}(\lambda))$ (Dziembowzki [Dzi06] gave a construction of such 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 $\mathsf{ct} := (\mathsf{nce.ct}, \mathsf{inc.ct})$. Note that this scheme is ciphertext-rate preserving, as the size of $\mathsf{nce.ct}$ is $|\mathsf{nce.ct}| = S + |m| + \mathrm{poly}'(\lambda)$.

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], is a natural generalization of incompressible PKE to the ABE and IBE settings. Here again, the syntax is 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 IBE. However, instead of receiving the secret key for id^* (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 secure 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, RNC-IBE and receiver non-committing attribute based and identity-based key encapsulation mechanism, based on various standard assumptions. These construction, in turn, give us the first strong incompressible ABE and IBE schemes *with* optimal ciphertext size.

Main Results: The first construction is based on the bilinear DDH 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 Remark 10.1 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* 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 randomness of setup, 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 also resolves questions Q2 and Q3 simultaneously! In fact, 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. This is obtained by combining the RNC-AB-KEM or RNC-IB-KEM with the incompressible secret key encryption scheme of [BDD22] (based on the LWE or DCR assumptions). In this incompressible encryption scheme, the size of the ciphertext is $\max(S, |m|) + \mathrm{poly}(\lambda)$. The size of the secret key grows with $|S|$, but since the size of the RNC-AB-KEM or RNC-IB-KEM's ciphertext is independent of the session-key size, this does not affect the final ciphertext size.

Theorem 1.2. *Assuming the existence of RNC-IB-KEM (RNC-AB-KEM for predicate classes with predicate encodings) such that the size of the ciphertext is independent of the size of the session key, there exists an adaptively secure strongly incompressible IBE (ABE for predicate classes with predicate encoding) scheme with optimal ciphertext size ($\max(S, |m|) + \mathrm{poly}(\lambda)$). In particular, we get adaptively secure strongly incompressible IBE with optimal ciphertext size, assuming the hardness of* SXDH *and* LWE *(or* DCR*).*

RNC-IBE and Incompressible IBE Constructions for polynomially bounded identity space: The second RNC-IBE construction supports polynomially many identities with compact master public key. It can be instantiated from a broader class of assumptions such as $\{\mathsf{DDH}, \mathsf{LWE}\}$. This construction introduces an additional feature where 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 is the first construction to offer such 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 NC-IBE that supports polynomially many identities.*

By combining these results with an incompressible secret key encryption scheme in a hybrid encryption framework, we obtain the first strongly incompressible IBE schemes from DDH/LWE. The incompressible IBE scheme supports polynomially many identities with compact master public key. 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) strongly incompressible IBE schemes 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. While [HMNY21] also gave a construction of RNC-ABE using iO and one way functions, our approach differs substantially.

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

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 Dziembowzki's incompressible SKE scheme (with secret key size of $\mathrm{poly}(\lambda)$), we can obtain rate-$\frac{1}{2}$ strongly incompressible IBE schemes (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 schemes.*

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. After a decade, 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 achieves 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 gain 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. This makes DE a stronger notion 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 our RNC-IBE construction, which produces a session key in $\mathbb{G}_T$. By using predicate encoding, we can lift this construction to the ABE setting – 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. These proofs tend to be intricate and delicate. At a high level, our security proof constructs the semi-functional secret keys and the master public key using a portion of the master secret key. This approach provides the flexibility to reprogram the remaining part of the master secret key, ensuring that the challenge ciphertext can be decrypted to yield 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 sample a random element $s \leftarrow \mathbb{Z}_p$ and output $\mathsf{sk}_{\mathsf{id}} := ([s\mathbf{b}]_2, [\mathbf{k} + s(\mathbf{W}_1 + \mathsf{id} \cdot \mathbf{W}_2)^\top\mathbf{b}]_2)$. 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 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 the adversary has queried a single identity, denoted by id, and that the adversary's target identity is denoted as id^*. We begin with the following:

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

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

By applying the SXDH assumption, we can replace $r^*\mathbf{a}$ and $s\mathbf{b}$ with a truly random elements $\mathbf{u}, \mathbf{v} \leftarrow \mathbb{Z}_p$.

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

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

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.

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

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

where $\mathbf{u} = r'\mathbf{a} + r\mathbf{b}^\perp$ and $\mathbf{v} = t'\mathbf{a} + t\mathbf{b}^\perp$ such that $r', r, t', t \in \mathbb{Z}_p$, i.e., we can express $\mathbf{u}, \mathbf{v}$ in terms of $\mathbf{a}, \mathbf{b}^\perp$ because they are linearly independent with high probability. 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

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

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

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 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 $u_1, u_2 \leftarrow \mathbb{Z}_p$ because $\mathbf{u}$ is chosen uniformly at random. Now, given a uniformly random $x \leftarrow \mathbb{Z}_p$, we can program the master secret key as $k_2 = \dfrac{x - u_1 k_1}{u_2}$. 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

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

For the complete construction and proof, refer to Section 10.

RNC-IBE from Batch Encryption

Let us start by reviewing the concept of batch encryption. A batch encryption scheme is a form of public key encryption where the key generation process is a projection—this means that the secret key is used to produce a shorter public key. When the secret key has a length of n, the scheme allows the simultaneous encryption of $n \times 2$ messages. During decryption, only one message from each pair can be recovered, and this is determined by the corresponding bit in the secret key.

To elaborate, the setup algorithm $\mathsf{BE.Setup}$ takes as input a secret key $\mathsf{sk} \in \{0,1\}^n$ and outputs a public key pk. The encryption algorithm encrypts a matrix $M \in \mathcal{M}^{n\times 2}$ using the public key to generate a ciphertext ct. Here, $\mathcal{M}$ is an appropriate message space. The decryption algorithm takes as input the secret key sk and the ciphertext ct and outputs a vector $m \in \mathcal{M}^n$ such that $m_i = M_{i,\mathsf{sk}[i]}$, for all $i \in [n]$.

The RNC-IBE scheme uses the batch encryption and 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. The setup algorithm generates T pairs of NCE public and secret keys, denoted as $\{ \mathsf{nce.pk}_j, \mathsf{nce.sk}_j \}_{j\in[T]}$, where each pair corresponds to a different identity. These keys together form the master secret key of the RNC-IBE scheme. The master public key is a public key of the batch encryption scheme generated by $\mathsf{be.pk} \leftarrow \mathsf{BE.Setup}(\{ \mathsf{nce.pk}_j \}_{j\in[T]})$, i.e., $\{ \mathsf{nce.pk}_j \}_{j\in[T]}$ is the secret key associated with $\mathsf{be.pk}$.

The secret key for the i^{th} identity consists of all the NCE public keys $\{ \mathsf{nce.pk}_j \}_{j\in[T]}$ along with the i^{th} secret key $\mathsf{nce.sk}_i$. To encrypt a message m for the i^{th} identity, the encryption algorithm generates T garbled circuit labels as follows:

- For the i^{th} identity, it generates $(\tilde{\mathcal{C}}^{(i)}, \{ \mathsf{lab}_{j,b}^{(i)} \}) \leftarrow \mathsf{GC.Grbl}(\mathsf{NCE.Enc}(\cdot, m))$.
- For the remaining identity, it generates $(\tilde{\mathcal{C}}^{(k)}, \{ \mathsf{lab}_{j,b}^{(k)} \}) \leftarrow \mathsf{GC.Grbl}(\mathsf{NCE.Enc}(\cdot, m^{(k)}))$ where $m^{(k)}$ is randomly generated.

A matrix $M \in \{0,1\}^{nT\times 2}$ is then constructed such that $M[k \cdot n + j, b] = \mathsf{lab}_{j,b}^{(k)}$. The batch encryption scheme is then used to produce $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{be.pk}, M)$. The final ciphertext is $\mathsf{ct} := (\{ \tilde{\mathcal{C}}^{(k)} \}_k, \mathsf{be.ct})$.

The simulation works as follows. First, the simulator generates the NCE public keys and corresponding ciphertexts using the NCE simulators and constructs the master public key. The simulator then produces the non-committing ciphertext by simulating all garbled circuit labels, i.e., $(\tilde{\mathcal{C}}^{(k)}, \{ \mathsf{lab}_{j}^{(k)} \}) \leftarrow \mathsf{GC.Sim}(\mathsf{nce.ct}^{(k)})$.

Upon receiving the target identity i and target message m, the simulator uses the NCE simulator to simulate $\mathsf{nce.sk}_i$ such that the ciphertext $\mathsf{nce.ct}_i$ will decrypt to m using $\mathsf{nce.sk}_i$. For the other identities, the simulator uses the NCE simulators to simulate $\mathsf{nce.sk}_k$ on random messages $m^{(k)}$. For more details, refer 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, j, i.e., $[i, j] := \{ i, i+1, \ldots, j \}$. For any two binary string 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. And $x[i, j]$ denotes the substring $x[i] \| x[i+1] \| \ldots \| x[j]$ when $i \leq j$ and $x[i, j] = \epsilon$ where $i > j$. 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 element, 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 $i \in [n]$.

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 $e(g_1, g_2)$, 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 (i, j)-th element is $[\mathbf{A}_{i,j}]_s$. Since, $[\cdot]_s$ is a linear functions, 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 $f : \underbrace{\mathbb{Z}_p^\ell \times \cdots \times \mathbb{Z}_p^\ell}_{n} \to \underbrace{\mathbb{Z}_p^\ell \times \cdots \times \mathbb{Z}_p^\ell}_{m}$.

Therefore, for any $\mathbb{Z}_p$-linear function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^m$, we will use the notation $f([x]_s)$[2] to denote $[f(x)]_s$. With this, we have

$$e([\mathbf{a}^\top]_1, f([\mathbf{b}_1]_2, \ldots, [\mathbf{b}_n]_2)) = f([\mathbf{a}^\top \mathbf{b}_1]_T, \ldots, [\mathbf{a}^\top \mathbf{b}_n]_T)$$
$$e(f([\mathbf{b}_1^\top]_2, \ldots, [\mathbf{b}_n^\top]_2), [\mathbf{a}]_1,) = f([\mathbf{b}_1^\top \mathbf{a}]_T, \ldots, [\mathbf{b}_n^\top \mathbf{a}]_T)$$

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 consist 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.

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 α 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. Refer Section 9 for an example of predicate encoding for the identity predicate, that is, $P(x, y) = 1 \iff x = y$.

[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.

3.2 Hardness Assumptions

3.2.1 Decisional Diffie-Hellman:

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

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

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

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

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 all 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 D_σ being an error distribution,

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

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. A batch encryption (BE) scheme 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 $M\in\{0,1\}^{n\times B}$ 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 outputs $\perp$ or a vector $m\in\{0,1\}^n$.

Correctness. For correctness, we require that for all $\lambda\in\mathbb{N}, n, B\in\mathbb{N}, \mathsf{sk}\in[B]^n, M\in\{0,1\}^{n\times B}$,

$$\Pr\Big[\mathsf{Dec}(\mathsf{sk},\mathsf{ct})=m \mid \mathsf{ct}\leftarrow\mathsf{Enc}(\mathsf{pk},M),\mathsf{pk}\leftarrow\mathsf{Setup}(\mathsf{pp},\mathsf{sk}),\mathsf{pp}\leftarrow\mathsf{Params}(1^\lambda,1^n)\Big]=1$$

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

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

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

Definition 3.2. *An 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 which 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 emphasis that the constructions given in [BLSV18] for blind batch encryptions are oblivious.

Theorem 3.3 ([BLSV18]). *Assuming the hardness of $\mathcal{C}$ where $\mathcal{C} \in \{\mathsf{CDH}, \mathsf{LWE}\}$, there exists a secure oblivious BE scheme such that the size of the public key is a 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 master public key pk, 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 $1^\lambda$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 $\lambda \in \mathbb{N}, T \in \mathbb{N}$ and $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}(1^\lambda)$, 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 b.

Simulated World:

- **Initialization Phase:** The challenger computes $(\mathsf{pk}, \mathsf{ct}^*, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(1^\lambda)$ and sends mpk 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 b.

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]). *Assuming the hardness of* LWE *and* DDH*, 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 a message (m_0, m_1), along with an 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 $\lambda \in \mathbb{N}$,*

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

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 $\frac{|m|}{|\mathsf{ct}|} = |m| - o(|m|)$.

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

Theorem 3.8 ([Dzi06]). *There exists a rate-*$\frac{1}{2}$ *incompressible SKE from one-way functions 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 $\tilde{C} : \{0,1\}^n \to \{0,1\}^m$ such that

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

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

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

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

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 input as 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 following quantity is negligible:*

$$\mathsf{Adv}^{ppriv}[\mathcal{A}] = |\Pr\left[\mathsf{Expt}_0^{\mathsf{ppriv}}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Expt}_1^{\mathsf{ppriv}}(\mathcal{A}) = 1\right]|$$

where $\mathsf{Expt}_b^{\mathsf{ppriv}}(\mathcal{A})$ *is given below:*

$\mathsf{Expt}_b^{\mathsf{ppriv}}$ is defined between a challenger and an adversary $\mathcal{A}$ for $\lambda \in \mathbb{N}$, which 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 Challenge Oracle and does not query the evaluation oracle on these points.
- A outputs a bit $b' \in \{0,1\}$, which is also output by $\mathsf{Expt}_b^{\mathsf{ppriv}}$

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

The following theorem states that privately programmable PRF can be built from IO and puncturable PRF. Puncturable PRF is a variant of 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 exits 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 input the security parameter 1^λ and length of the message 1^ℓ and outputs public parameter 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* $\lambda, \ell \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$, *input* x, *message* m *and 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 *adversary* $\mathcal{A}$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in N$, $\Pr[\mathcal{A}$ *wins in the above game*$] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

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

3.9 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme IBE for set of identity spaces $\mathcal{I} = \{\{0,1\}^n\}_{n\in\mathbb{N}}$ 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 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 $\mathsf{id} \in \{0,1\}^n$. 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 $\mathsf{id} \in \{0,1\}^n$. 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 $\lambda, n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $\mathsf{id} \in \{0,1\}^n$, $m \in \mathcal{M}$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, 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 adaptively secure IBE schemes.*

3.10 Incompressible IBE

In this section, we define the strong version of the incompressible security game for 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 $\mathcal{A}_1$ will be provided with the complete challenge ciphertext and produce a compressed version of it. The second adversary $\mathcal{A}_2$ is provided with the master public key, compressed challenge ciphertext which was created by $\mathcal{A}_1$ and certain secret keys.

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* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

[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.

Initialization Phase: $\mathcal{A}_1$ *on input* 1^λ*, outputs an upper bound on the state size* 1^S*. The challenger runs* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S; r_{\mathsf{Setup}})$ *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 messages* m_0, m_1*, an identity* id^* *along with an 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 ***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* $\lambda \in \mathbb{N}$*,*

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

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* $\lambda \in \mathbb{N}$*,*

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

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

3.11 Attribute-Based Encryption

An Attribute-Based Encryption (ABE) scheme ABE for 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.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the public parameters mpk, a message $m \in \mathcal{M}$, and an attribute $x \in \mathcal{X}$. 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* $\lambda, n \in \mathbb{N}$*, the following holds*

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

where all functions f *queried by* $\mathcal{A}$ *satisfy* $f(x^*) = 1$*.*

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* $\lambda, n \in \mathbb{N}$*, 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* $f(x^*) = 0$*.*

3.12 Incompressible ABE

In this section, we define the strong version of the incompressible security game for 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 $\mathcal{A}_1$ will be provided with the complete challenge ciphertext and produce a compressed version of it. The second adversary $\mathcal{A}_2$ is provided with the master public key, compressed challenge ciphertext which was created by $\mathcal{A}_1$ and certain secret keys.

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* $(\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ *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 messages* m_0, m_1*, an attribute* x^* *along with an 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, 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* $\lambda \in \mathbb{N}$*,*

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

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* $\lambda \in \mathbb{N}$*,*

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

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

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 in the challenge phase, the challenger returns the master secret key msk instead of the random coins used in the Setup and Enc algorithms. 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 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 id 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 id and a message 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_1.

$\mathsf{Sim}_2(\mathsf{st}_1, \mathsf{id})$: The second simulator is a stateful algorithm with an internal state st_2 that takes as input a state st_1 and an identity id and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ and updates st_2.

$\mathsf{Sim}_3(\mathsf{st}_2, \mathsf{id}^*)$: The third simulator takes as input a state st_2 and an identity id^* and outputs a ciphertext ct^* and a state st_3.

$\mathsf{Sim}_4(\mathsf{st}_3, m^*)$: The fourth simulator takes as input a state st_3 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 $\lambda \in \mathbb{N}, n \in \mathbb{N}$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, any identity id and message 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^T. The challenger computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^T)$ 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 where id^* was never queried in the pre-challenge query phase. The challenger computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$ and returns $(\mathsf{msk}, \mathsf{ct}^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

Simulated World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^T. The challenger computes $(\mathsf{mpk}, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^T)$ 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 $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, \mathsf{id})$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends m^*, id^* to the challenger where id^* was never queried in the pre-challenge query phase. The challenger computes $(\mathsf{ct}^*, \mathsf{st}_3) \leftarrow \mathsf{Sim}_3(\mathsf{st}_2, \mathsf{id}^*)$ and $\mathsf{msk} \leftarrow \mathsf{Sim}_4(\mathsf{st}_3, m^*)$. It returns $(\mathsf{msk}, \mathsf{ct}^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

It is important to note that in the challenge phase, the adversary obtains only the master secret key and the challenge ciphertext, and not the randomness used by the Setup algorithm or Enc algorithm to generate the challenge ciphertext. 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 addition takes id^* also as an additional input.

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

A receiver non-committing identity-based key-encapsulation mechanism (RNC-IB-KEM) 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 id 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 id and outputs a ciphertext ct and a session key seskey.

$\mathsf{Decap}(\mathsf{sk}, \mathsf{ct})$: 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_1.

$\mathsf{Sim}_2(\mathsf{st}_1, \mathsf{id})$: The second simulator is a stateful algorithm with an internal state st_2 that takes as input a state st_1 and an identity id and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ and updates st_2.

$\mathsf{Sim}_3(\mathsf{st}_2, \mathsf{id}^*)$: The third simulator takes as input a state st_2, an identity id^* and outputs a ciphertext ct^* and a state st_3.

$\mathsf{Sim}_4(\mathsf{st}_3, \mathsf{seskey})$: The third simulator takes as input a state st_3, a session key seskey and outputs a ciphertext msk.

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

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

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

Real World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^T. The challenger computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^T)$ 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 where id^* was never queried in the pre-challenge query phase. The challenger computes $(\mathsf{ct}^*, \mathsf{seskey}) \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$ and returns $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

Simulated World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs 1^T. The challenger computes $(\mathsf{mpk}, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^T)$ 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 $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, \mathsf{id})$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary $\mathcal{A}$ sends id^* to the challenger where id^* was never queried in the pre-challenge query phase. The challenger computes $(\mathsf{ct}^*, \mathsf{st}_3) \leftarrow \mathsf{Sim}_3(\mathsf{st}_2, \mathsf{id}^*)$ and $\mathsf{msk} \leftarrow \mathsf{Sim}_4(\mathsf{st}_3, \mathsf{seskey})$ where seskey is randomly generated. It returns $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

It is important to note that in the challenge phase, the adversary obtains only the master secret key, the challenge ciphertext and the session key, and not the randomness used by the Setup algorithm or Encap algorithm to generate the challenge ciphertext and the session key. 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 secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,*

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

By combining an RNC-IB-KEM with *one-time pad* encryption in a hybrid encryption approach, we can obtain an RNC-IBE scheme. This is possible because the RNC-IB-KEM (and RNC-IBE) requires the disclosure of the master secret key and not the randomness used by the Setup and/or Enc algorithms. However, the ciphertext-size will be the sum of the RNC-IB-KEM ciphertext-size and the size of the session-key.

Theorem 4.4. *Assuming the existence of secure RNC-IB-KEM, there exists secure RNC-IBE schemes.*

It is immediate to see that the above theorem holds even in the ABE setting.

Theorem 4.5. *Assuming the existence of secure RNC-AB-KEM, there exists secure RNC-ABE schemes.*

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

In this section, we present an adaptive 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 $\ell := \ell(\lambda)$ 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 parameter n, n_s, n_r (for details, refer Section 3.1).

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

- 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 to write it.

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

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

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

- Parse $([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]}) \leftarrow \mathsf{pp}$ and $\{ \mathbf{k}_i \}_{i\in[\ell]} \leftarrow \mathsf{msk}$.
- Generate $s_i \leftarrow \mathbb{Z}_p, \forall i \in [\ell]$.
- Output $\mathsf{sk}_y := \{ ([s_i\mathbf{b}]_2, \mathsf{kE}(y, [\mathbf{k}_i]_2) \cdot \mathsf{rE}(y, \{ [s_i\mathbf{W}_j^\top \mathbf{b}]_2 \}_{j\in[n]}) \}_{i\in[\ell]}$[4].

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

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

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

- Parse $\{ ([\mathbf{d}_i]_2, [\mathbf{d}'_i]_2) \}_{i\in[\ell]} \leftarrow \mathsf{sk}_{\mathsf{id}}$ and $([\mathbf{c}]_1, [\mathbf{c}']_1) \leftarrow \mathsf{ct}$.
- Output $\mathsf{seskey} \leftarrow \{ \mathsf{HC}(e([\mathbf{c}]_1^\top, \mathsf{rD}(x, y, [\mathbf{d}'_i]_2)) \;/\; e(\mathsf{sD}(x, y, [\mathbf{c}']_1)^\top, [\mathbf{d}_i]_2), h) \}_{i\in[\ell]}$.

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

- Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- Generate $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_q$ and $\mathbf{W}_1, \ldots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.

[4] $[\mathbf{u}]_2 \cdot [\mathbf{v}]_2$ denotes the component-wise group multiplication.

- Generate $k_{i,1} \leftarrow \mathbb{Z}_p, \forall i \in [\ell]$ and $h \leftarrow \{0,1\}^{\log(p)}$.
- Output $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$ and $\mathsf{mpk} := (\{ [k_{i,1}]_T \}_{i\in[\ell]}, h)$ and $\mathsf{st}_1 = (\{ k_{i,1} \}_{i\in[\ell]}, \mathbf{a})$.

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

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

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

- Generate $u_1, u_2 \leftarrow \mathbb{Z}_p$ and set $\mathbf{u} = u_1\mathbf{a} + u_2\mathbf{a}^\perp$.
- Outputs $\mathsf{ct} := ([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i\mathbf{u}]_1 \}_{i\in[n]})$ and $\mathsf{st}_3 := (\mathsf{st}_2, u_1, u_2)$.

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

- Generate $x_i \in \mathbb{Z}_p$ such that $\mathsf{seskey}_i = \mathsf{HC}([x_i]_T, h)$ via rejection sampling.
- Set $k_{i,2} := \frac{x_i - u_1 k_{i,1}}{u_2}$.
- Output $\mathsf{msk} := \{ \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp \}_{i\in[\ell]}$.

Remark 5.1. The randomness used in the setup algorithm includes h, which is part of the master public key and the set $\{ k_i \}$, which constitutes the entire master secret key. Since, the challenger outputs h in the initialization phase and provides the master secret key in the challenge, the adversary effectively gains access to all the randomness used in the setup algorithm during the challenge phase.

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

Correctness. For correctly generated $\mathsf{sk}_y := \{ ([s_i\mathbf{b}]_2, \mathsf{kE}(y, [\mathbf{k}_i]_2) \cdot \mathsf{rE}(y, \{ [s_i\mathbf{W}_j^\top \mathbf{b}]_2 \}_{j\in[n]}) \}_{i\in[\ell]}$ and $\mathsf{ct} := ([r\mathbf{a}]_1, \mathsf{sE}(x, \{ [r\mathbf{W}_j\mathbf{a}]_1 \}_{j\in[n]})$ and $\mathsf{seskey} := \{ \mathsf{HC}([r\mathbf{a}^\top \mathbf{k}_i]_T, h) \}_{i\in[\ell]}$, we have

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

The first equality follows from the fact that $\mathsf{rD}(x,y,\cdot)$ is $\mathbb{Z}_p$ linear. The second equality follows due to the equality that $\mathsf{rD}(x,y,\mathsf{kE}(y,\alpha)) = \alpha$. In third equality, we use the fact that the functions $e([r\mathbf{a}]_1^\top,\cdot)$, $(e(\cdot,[s\mathbf{b}]_2))$, $\mathsf{rD}(x,y,\mathsf{rE}(y,\cdot))$ and $\mathsf{sD}(x,y,\mathsf{sE}(x,\cdot))$ commute with linear functions (refer Equation (1)). And, the last equality is due to the fact that when $P(x,y)=1$, then $\mathsf{sD}(x,y,\mathsf{sE}(x,w)) = \mathsf{rD}(x,y,\mathsf{rE}(y,w))$.

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

Proof. Consider the following experiment for an adversary $\mathcal{A}$ that makes q queries to $O_{\mathsf{UserKeyGen}}$.

Hybrid Hyb_0**:** This corresponds to the real experiment of non-committing security.

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 $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2, \forall i \in [\ell]$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{[\mathbf{W}_i\mathbf{a}]_1\}_{i\in[n]}, \{[\mathbf{W}_i^\top\mathbf{b}]_2\}_{i\in[n]})$, $\mathsf{mpk} := \{[\mathbf{a}^\top\mathbf{k}_i]_T\}_{i\in[\ell]}$, and $\mathsf{msk} := \{\mathbf{k}_i\}_{i\in[\ell]}$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(y^j)$**:** Given the j-query $y^j \in \mathcal{Y}$ as an input, it returns sk_{y^j} generated as follows.
   - Generate $s_d^j \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - Set $\mathsf{sk}_{y^j} := \{([s_d^j\mathbf{b}]_2, \mathsf{kE}(y^j,[\mathbf{k}_d]_2)\cdot \mathsf{rE}(y^j, \{[s_d^j\mathbf{W}_\alpha^\top\mathbf{b}]_2\}_{\alpha\in[n]})\}_{d\in[\ell]}$.
3. $\mathcal{A}$ outputs $x^* \in \mathcal{X}$. The challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $r \leftarrow \mathbb{Z}_p$.
   - Set $\mathsf{ct}^* := ([r\mathbf{a}]_1, \mathsf{sE}(x^*, \{[r\mathbf{W}_i\mathbf{a}]_1\}_{i\in[n]})$ and $\mathsf{seskey}^* := \{\mathsf{HC}([r\mathbf{a}^\top\mathbf{k}_i]_T, h)\}_i$.

   The challenger sends $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Using a sequence of hybrid experiments, we will prove that the experiment can be indistinguishably changed into non-committing experiment. We will denote the probability that $\mathcal{A}$ outputs 0 in the hybrid Hyb_i using $p_{\mathcal{A},H_i}$.

Below, we assume that $\mathbf{a}$ and $\mathbf{b}^\perp$ are linearly independent and $\mathbf{a}^\perp$ and $\mathbf{b}$ are also linearly independent, which hold 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

$$\mathsf{ct}^* := ([\mathbf{u}]_1, \mathsf{sE}(x^*, \{[\mathbf{W}_i\mathbf{u}]_1\}_{i\in[n]})) \text{ and } \mathsf{seskey}^* := \{[\mathbf{u}^\top\mathbf{k}_i]_T\}_i,$$

where $\mathbf{u} \leftarrow \mathbb{Z}_p^2$.

We have $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| = \mathsf{negl}(\lambda)$ 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, $|p_{\mathcal{A},H_1} - p_{\mathcal{A},H_0}| \le p_{\mathcal{B},\mathsf{DDH}}$.*

We define $\mathsf{Hyb}_{1,0,5}$ as Hyb_1.

Changing the user secret keys into semi-functional mode. We change the user secret keys into semi-functional mode using $\mathsf{Hyb}_{1,i,1}, \cdots, \mathsf{Hyb}_{1,i,5}$ for $i \in [q]$. Below, we 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.

$O_{\mathsf{UserKeyGen}}(y^j)$**:** Given the j-th query $y^j \in \mathcal{Y}$ as an input, it behaves as follows.

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

We have $\left| \Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_{1,1,1} = 1] \right| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$.

Lemma 5.4. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that, $|p_{\mathcal{A},H_{1,1,1}} - p_{\mathcal{A},H_1}| \leq \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

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 $\left| \Pr[\mathsf{Hyb}_{1,i,1} = 1] - \Pr[\mathsf{Hyb}_{1,i,2} = 1] \right| = 0$ since the distribution of $\{ \mathbf{W}_\alpha \}_{\alpha \in [n]}$ do not change.

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

We prove that w_α's appears only in ct^* and sk_{y^i}. First, we have $[\mathbf{W}_\alpha \mathbf{a}]_1 = [\hat{\mathbf{W}}_\alpha \mathbf{a}]_1$ and $[\mathbf{W}_\alpha^\top \mathbf{b}]_2 = [\hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2$ for all $\alpha \in [n]$. For ct^*, we can write $\mathbf{u} = r\mathbf{a} + r'\mathbf{b}^\perp$, and thus we have

$$\begin{aligned}
\mathsf{ct}^* &= \left([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_\alpha \mathbf{u}]_1 \}_{\alpha \in [n]}) \right) \\
&= \left([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [(\hat{\mathbf{W}}_\alpha + w_\alpha \mathbf{W}_0)\mathbf{u}]_1 \}_{\alpha \in [n]}) \right) \\
&= \left([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\hat{\mathbf{W}}_\alpha \mathbf{u}]_1 \}_{\alpha \in [n]}) \cdot \mathsf{sE}(x^*, \{ [r' w_\alpha \mathbf{b}^\perp]_1 \}_{\alpha \in [n]}) \right)
\end{aligned}$$

Also, for sk_{y^i}, we can write $\mathbf{v}_d^i = s_d^i \mathbf{b} + t_d^i \mathbf{a}^\perp$ and thus we have

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

Moreover, for $j \neq i$, we can write sk_{y^j} as

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

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

As shown previously, w_α's are only present in the challenge ciphertext ct^* and sk_{y^i}. We want to show that

$$\Big[\mathsf{pp}, \mathsf{msk}, \mathbf{a}^\perp, \mathbf{b}^\perp, \underbrace{([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\hat{\mathbf{W}}_\alpha \mathbf{u}]_1 \}_{\alpha\in[n]}) \cdot \mathsf{sE}(x^*, \{ [r' w_\alpha \mathbf{b}^\perp]_1 \}_{\alpha\in[n]}))}_{\mathsf{ct}^*},$$
$$\underbrace{\{ ([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d]_2) \cdot \mathsf{rE}(y^i, \{ [\hat{\mathbf{W}}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]})) \cdot \mathsf{rE}(y^i, \{ [t_d^i w_\alpha \mathbf{a}^\perp]_2 \}_{\alpha\in[n]})) \}_d}_{\mathsf{sk}_{y^i} \text{ in } \mathsf{Hyb}_{1,i,2}} \Big] \approx_s$$
$$\Big[\mathsf{pp}, \mathsf{msk}, \mathbf{a}^\perp, \mathbf{b}^\perp, \underbrace{([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\hat{\mathbf{W}}_\alpha \mathbf{u}]_1 \}_{\alpha\in[n]})) \cdot \mathsf{sE}(x^*, \{ [r' w_\alpha \mathbf{b}^\perp]_1 \}_{\alpha\in[n]}))}_{\mathsf{ct}^*},$$
$$\underbrace{\{ ([\mathbf{v}_d^i]_2, \mathsf{kE}(y^i, [\mathbf{k}_d + w_d^i \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^i, \{ [\hat{\mathbf{W}}_\alpha^\top \mathbf{v}_d^i]_2 \}_{\alpha\in[n]})) \cdot \mathsf{rE}(y^i, \{ [t_d^i w_\alpha \mathbf{a}^\perp]_2 \}_{\alpha\in[n]})) \}_d}_{\mathsf{sk}_{y^i} \text{ in } \mathsf{Hyb}_{1,i,3}} \Big]$$

It is enough to show that

$$\{x^*, y^i, w_d^i, \mathsf{sE}(x^*, \{ w_\alpha \}_{\alpha\in[n]}), \mathsf{kE}(y^i, w_d^i) + \mathsf{rE}(y^i, \{ w_\alpha \}_{\alpha\in[n]})\}_{w_\alpha} \approx_s$$
$$\{x^*, y^i, w^i, \mathsf{sE}(x^*, \{ w_\alpha \}_{\alpha\in[n]}), \mathsf{rE}(y^i, \{ w_\alpha \}_{\alpha\in[n]})\}_{w_\alpha}$$

because a sample from one of the above distributions can be transformed into the corresponding distribution from the original set of distributions by sampling $\mathsf{pp}, \mathsf{msk}, \mathbf{a}^\perp, \mathbf{b}^\perp, \mathbf{u}, \hat{\mathbf{W}}_\alpha, r', t_d^i, \mathbf{v}_d^i, \mathbf{k}_d$ appropriately and applying the necessary operations.

Observe that from the linearity property of the predicate encoding and Equation (1), we have

$$\mathsf{kE}(y^i, [\mathbf{k}_d + w_d^i \mathbf{a}^\perp]_2) = [\mathsf{kE}(y^i, \mathbf{k}_d) + \mathsf{kE}(y^i, w_d^i \mathbf{a}^\perp)]_2.$$

From the α-privacy of the predicate encoding, we have for all $d \in [\ell]$,

$$\{x^*, y^i, w_d^i, \mathsf{sE}(x^*, \{ w_\alpha \}_{\alpha\in[n]}), \mathsf{kE}(y^i, w_d^i) + \mathsf{rE}(y^i, \{ w_\alpha \}_{\alpha\in[n]})\}_{w_\alpha} \approx_s$$
$$\{x^*, y^i, w^i, \mathsf{sE}(x^*, \{ w_\alpha \}_{\alpha\in[n]}), \mathsf{rE}(y^i, \{ w_\alpha \}_{\alpha\in[n]})\}_{w_\alpha}$$

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

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 $\mathsf{Hyb}_{1,i,2}$. Namely, we generate $\mathbf{W}_\alpha \leftarrow \mathbb{Z}_p^{2\times 2}, \forall \alpha \in [n]$.

We have $\big|\Pr\big[\mathsf{Hyb}_{1,i,3} = 1\big] - \Pr\big[\mathsf{Hyb}_{1,i,4}\big]\big| = 0$.

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

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

We have $\big|\Pr\big[\mathsf{Hyb}_{1,i,4} = 1\big] - \Pr\big[\mathsf{Hyb}_{1,i,5} = 1\big]\big| = \mathsf{negl}(\lambda)$ 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, $|p_{\mathcal{A},H_{1,i,5}} - p_{\mathcal{A},H_{1,i,4}}| \le \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

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

Lemma 5.9. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that, $|p_{\mathcal{A},H_{1,i+1,1}} - p_{\mathcal{A},H_{1,i,5}}| \le \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Final steps towards non-committing mode.

Hybrid Hyb_2**:** We define Hyb_2 as the same game as $\mathsf{Hyb}_{1,q,5}$. The detailed description is as follows.

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 $\mathbf{k}_d \leftarrow \mathbb{Z}_p^2, \forall d \in [\ell]$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, \{ [\mathbf{W}_i\mathbf{a}]_1 \}_{i\in[n]}, \{ [\mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})$, $\mathsf{mpk} := (\{ [\mathbf{a}^\top \mathbf{k}_d]_T \}_{d\in[\ell]}, h)$, and $\mathsf{msk} := \{ \mathbf{k}_d \}_{d\in[\ell]}$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query $y^j \in \mathcal{Y}$ as an input, it returns sk_{y^j} generated as follows.
   - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - Set $\mathsf{sk}_{y^j} := \{ ([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\mathbf{k}_d + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_i^\top \mathbf{b}]_2 \}_{i\in[n]})) \}_{d\in[\ell]}$.
3. $\mathcal{A}$ outputs $x^* \in \mathcal{X}$. The challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $\mathbf{u} \leftarrow \mathbb{Z}_p^2$.
   - Set $\mathsf{ct}^* := ([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i\mathbf{u}]_1 \}_{i\in[n]}))$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_d]_T, h) \}_{d\in[\ell]}$.

   The challenger sends $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

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

This is just conceptual change and we have $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| = 0$.

Lemma 5.10. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.*

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

We have $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = 0$ since $\frac{k_{d,2}}{|\mathbf{a}^\perp|^2} + w_d^j$ and w_d^j identically distributes for every $j \in [q]$ when w_d^j is chosen uniformly at random.

Lemma 5.11. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = 0$.*

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

This is just conceptual change and we have $|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| = 0$.

Lemma 5.12. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_5} - p_{\mathcal{A},H_4}| = 0$.*

Hybrid Hyb_6**:** This is the same as Hyb_5 except that we generate $x_d \leftarrow \mathbb{Z}_p$ and we set

$$k_{2,d} = \frac{x_d - u_1 k_{d,1}}{u_2}.$$

By this change, we have $\mathsf{seskey}^* = \{ \mathsf{HC}([x_d]_T, h) \}_{d\in[\ell]}$.

We have $|\Pr[\mathsf{Hyb}_5 = 1] - \Pr[\mathsf{Hyb}_6 = 1]| = 0$ since k_2 still distributes uniformly at random.

Hybrid Hyb_7**:** This is the same as Hyb_6 except that we defer the generation of k_2 until the challenge phase. The detailed description is as follows.

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 $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p$ and $\mathbf{W}_1, \ldots \mathbf{W}_n \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $k_{d,1} \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - 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 $\mathsf{mpk} := (\{ [k_{d,1}]_T \}_{d\in[\ell]}, h)$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(y^j)$**:** Given the j-query $y^j \in \mathcal{Y}$ as an input, it returns sk_{y^j} generated as follows.
   - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - Set $\mathsf{sk}_{y^j} := \{ ([s_d^j \mathbf{b}]_2, \mathsf{kE}(y^j, [\frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + w_d^j \mathbf{a}^\perp]_2) \cdot \mathsf{rE}(y^j, \{ [s_d^j \mathbf{W}_1^\top \mathbf{b}]_2 \}_{i\in[n]})) \}_{d\in[\ell]}$.
3. $\mathcal{A}$ outputs $x^* \in \mathcal{X}$. The challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $u_1, u_2, x_d \leftarrow \mathbb{Z}_p$ and set $\mathbf{u} = u_1\mathbf{a} + u_2\mathbf{a}^\perp$.
   - Set $\mathsf{ct}^* := ([\mathbf{u}]_1, \mathsf{sE}(x^*, \{ [\mathbf{W}_i\mathbf{u}]_1 \}_{i\in[n]}))$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([x_d]_T, h) \}_{d\in[\ell]}$.
   - Set $k_{d,2} = \frac{x_d - u_1 k_{d,1}}{u_2}$ and $\mathsf{msk} := \frac{k_{d,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{d,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp$.

   The challenger sends $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

It is easy to see that Hyb_7 can be simulated using the RNC-AB-KEM simulators. Using the above lemma along with triangular inequality, the theorem follows. □

Combining Theorem 4.5 and Theorem 5.2, we obtain the following.

Theorem 5.13. *Assuming hardness of* SXDH*, there exists secure RNC-ABE schemes.*

6 Receiver Non-Committing IBE from Batch Encryption

In this section, we construct an adaptive secure RNC-IBE scheme which 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.

Let d be the length of the identities such that $T = 2^d = \mathrm{poly}(\lambda)$ and n be the length of the public keys of NCE.

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

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

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

- Generate r uniformly at random.
- Generate $(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{ \mathsf{lab}_{i,b}^{(\mathsf{id})} \}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m; r))$.
- Generate $(\tilde{\mathcal{C}}^{(k)}, \{ \mathsf{lab}_{i,b}^{(k)} \}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^k; r^k))$ 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 $M[int(k) \cdot n + j, b] = \mathsf{lab}_{j,b}^{(k)}$ for all $k \in \{0,1\}^d, b \in \{0,1\}, j \in [n]$ where $int : \{0,1\}^d \to [T]_0$ is a lexicographical mapping from the set of binary string to integers.
- Compute $\mathsf{be.ct} \leftarrow \mathsf{BE.Enc}(\mathsf{mpk}, M)$.
- Output $\mathsf{ct} := (\{ \tilde{\mathcal{C}}^{(k)} \}_{k \in \{0,1\}^d}, \mathsf{be.ct})$.

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

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

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

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

$\mathsf{Sim}_2(\mathsf{st}^{(1)}, \mathsf{id})$:

- If $(\mathsf{id}, \cdot) \notin \mathsf{st}^{(2)}$, it computes $(\cdot, r^{\mathsf{id}}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_{\mathsf{id}}, m)$ where m is randomly generated and updates $\mathsf{st}^{(2)} = \mathsf{st}^{(2)} \cup \{ \mathsf{id}, r^{\mathsf{id}}_{\mathsf{NCE.Setup}} \}$.
- Compute $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; r^{(\mathsf{id})}_{\mathsf{NCE.Setup}})$.
- Output $\mathsf{sk}_{\mathsf{id}} := (\{ \mathsf{nce.pk}_i \}_{i \in \{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}})$.

$\mathsf{Sim}_3(\mathsf{st}^{(1)}, \mathsf{st}^{(2)}, \mathsf{id}, m)$:

- Compute $(\cdot, r^{\mathsf{id}}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_{\mathsf{id}}, m)$.
- For all $\mathsf{id}' \notin \{\mathsf{st}^{(2)}[0] \cup \{\mathsf{id}\}\}$, it computes $(\cdot, r^{\mathsf{id}'}_{\mathsf{NCE.Setup}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}_{\mathsf{id}'}, m_{\mathsf{id}'})$ where $m_{\mathsf{id}'}$'s are randomly generated.
- Output $r_{\mathsf{Setup}} := \{\mathsf{be.pp}, \{r^i_{\mathsf{NCE.Setup}}\}_{i\in\{0,1\}^d}\}$.

Remark 6.1. From the oblivious property of BE, the randomness used in the setup algorithm is $r_{\mathsf{Setup}} = \{\mathsf{be.pp}, \{r^{(i)}_{\mathsf{NCE.Setup}}\}_{i\in\{0,1\}^d}\}$.

Parameters. The size of a ciphertext is $\mathrm{poly}(2^d, |m|, \lambda)$, whereas the size 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 $\mathsf{sk}_{\mathsf{id}} := (\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}})$ and a ciphertext $\mathsf{ct} := (\{\tilde{\mathcal{C}}^{(k)}\}_{k\in\{0,1\}^d}, \mathsf{be.ct})$, after performing the BE decryption $d \leftarrow \mathsf{BE.Dec}(\mathsf{be.pp}, \{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{be.ct})$, we have $d[int(\mathsf{id})\cdot n + i] = \mathsf{lab}_{i,\mathsf{nce.pk}_{\mathsf{id}}[i]}$. This is due to the correctness of the BE scheme. From the correctness of 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 support polynomially many identities.*

Proof. We will show that the above scheme is an adaptive secure RNC-IBE using a sequence of hybrid arguments.

Hybrid Hyb_0 : This is the original adaptive NC-IBE game.

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. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query id^j as an input, it returns $\mathsf{sk}_{\mathsf{id}^j} = (\{\mathsf{nce.pk}_i\}_{i\in\{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}^j})$.
3. $\mathcal{A}$ outputs id^*, m^*. The challenger generates ct^* as follows.
   - Generate r uniformly at random.
   - Generate $(\tilde{\mathcal{C}}^{(\mathsf{id})}, \{\mathsf{lab}^{(\mathsf{id})}_{i,b}\}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m; r))$.
   - Generate $(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}^{(k)}_{i,b}\}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{NCE.Enc}(\cdot, m^k; r^k))$ 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 $M[int(k)\cdot n + j, b] = \mathsf{lab}^{(k)}_{j,b}$ for all $k \in \{0,1\}^d, b \in \{0,1\}, j \in [n]$.
   - Set $\mathsf{ct} := (\{\tilde{\mathcal{C}}^{(k)}\}_{k\in\{0,1\}^d}, \mathsf{be.ct})$.

   The challenger sends $(\mathsf{ct}^*, \{r^{(i)}_{\mathsf{NCE.Setup}}\})$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

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

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 sends it to the challenger who replies with be.pp (which is also the randomness used during BE.Params). It can use be.pp to generates the master public key mpk. The reduction can generate the two matrices used in Hyb_0 and Hyb_1 and sends it to the challenger. Note that these matrices differ only at indices $(int(k) \cdot n + j, 1 - \mathsf{nce.pk}_k[j])$. It will receive be.ct from the challenger and then simulate 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 $r^{\mathsf{id}}_{\mathsf{NCE.Setup}}$ and be.pp. □

Hybrid Hyb_2 : In this game, the garbled circuit $\tilde{\mathcal{C}}^{(k)}$ and labels are simulated, i.e., $(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}^{(k)}_i\}) \leftarrow \mathsf{GC.Sim}(1^\lambda, \mathsf{nce.ct}^{(k)})$ where $\mathsf{nce.ct}^{(k)} \leftarrow \mathsf{NCE.Enc}_1(\mathsf{nce.pk}_k, m^{(k)}; r^{(k)})$ for all $k \in \{0,1\}^d \setminus \{\mathsf{id}^*\}$. And, $\mathsf{nce.ct}^{(\mathsf{id}^*)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}_{\mathsf{id}^*}, m^*)$.

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 $\mathsf{id} \in \{0,1\}$. □

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 $T+1$ many intermediate hybrids $\mathsf{Hyb}_{2,i}$ where $\mathsf{Hyb}_{2,0} = \mathsf{Hyb}_2$ and $\mathsf{Hyb}_{2,T+1} = \mathsf{Hyb}_3$. 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.
- To respond to a key query, if $\mathsf{id} > i$, it returns $\mathsf{sk}_{\mathsf{id}} = (\{\mathsf{nce.pk}_i\}_{i \in \{0,1\}^d}, \mathsf{nce.sk}_{\mathsf{id}})$. Whereas, if $\mathsf{id} \leq i$, it first checks whether $(\mathsf{id}, r^{\mathsf{id}}_{\mathsf{NCE.Setup}}) \in \mathsf{st}^{(2)}$. If it exists, then it computes $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; r^{\mathsf{id}}_{\mathsf{Setup}})$ and then returns $\mathsf{sk}_{\mathsf{id}}$ accordingly. If not, it calls the second simulator of the NCE on a random message m which returns $(\cdot, r^{(\mathsf{id})}_{\mathsf{Setup}})$. It computes $(\cdot, \mathsf{nce.sk}_{\mathsf{id}}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; r^{\mathsf{id}}_{\mathsf{Setup}})$ and then returns $\mathsf{sk}_{\mathsf{id}}$ accordingly. It also updates $\mathsf{st}^{(2)} = \mathsf{st}^{(2)} \cup (\mathsf{id}, r^{(\mathsf{id})}_{\mathsf{Setup}})$.
- In the challenge phase, when it receives the challenger identity id^* and message m^*, it will generate $\mathsf{nce.ct}^{(k)}$ for all $k > i$ honestly, i.e., using NCE.Enc algorithm. The remaining $\mathsf{nce.ct}^{(k)}$ for $k \leq i$ will be the ciphertext simulated by the first simulator in the initialization phase. Then, it produces to generate $(\tilde{\mathcal{C}}^{(k)}, \{\mathsf{lab}^{(k)}_i\})$ for all k and sets up M appropriate to generate be.ct. Finally,
  - if $\mathsf{id}^* \leq i$, it will use the NCE simulator on m^* to obtain $r^{\mathsf{id}^*}_{\mathsf{Setup}}$.
  - For all $\mathsf{id} \leq i$ that was not queried and not equal to id^*, it will use the NCE simulator on a random m to obtain $r^{\mathsf{id}}_{\mathsf{Setup}}$.

  Finally, it will output $(\tilde{\mathcal{C}}^{(k)}, \mathsf{be.ct})$ and $\{r^{\mathsf{id}}_{\mathsf{Setup}}\}_{\mathsf{id}}$.

It is easy to show that an $\mathcal{A}$ 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 triangular 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 Obfuscator

In this section, we construct a selective secure receiver non-committing encryption identity-based scheme using indistinguishable obfuscators (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 scheme.
- $\mathsf{PPRF} = (\mathsf{PPRF.Setup}, \mathsf{PPRF.Eval}, \mathsf{PPRF.Program}, \mathsf{PPRF.PEval})$ be a secure programmable PRF[5].
- iO be a secure indistinguishable obfuscator.
- $\mathsf{NCE} = (\mathsf{NCE.Setup}, \mathsf{NCE.Enc}, \mathsf{NCE.Dec})$ be a non-committing encryption scheme.

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 $1 \leq i \leq j \leq n$. For any $i > j$, $\mathsf{id}[i,j] = \epsilon$. We first describe two algorithms - Node and Leaf.

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

- 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 $x^{(v)} := \mathsf{ss.vk}^{(v\|0)}\|\mathsf{ss.vk}^{(v\|1)}$.
- Computes $\mathsf{ss}.\sigma^{(v)} \leftarrow \mathsf{SS.Sign}(\mathsf{pp}, \mathsf{ss.sk}^{(v)}, x^{(v)})$.
- 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 $(\mathsf{nce.pk}^{(v\|0)}, \mathsf{nce.sk}^{(v\|0)}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.Eval}(s, v\|0))$.
- Generate $(\mathsf{nce.pk}^{(v\|1)}, \mathsf{nce.sk}^{(v\|1)}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.Eval}(s, v\|1))$.
- 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 will assume that the size of the NCE public keys and OTSE verification keys are of size λ.

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

- Generate $\mathsf{pprf.msk} \leftarrow \mathsf{PPRF.Setup}(1^\lambda, 1^{n+1})$.

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

- Compute $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{0^i\}_{i\in[n]_0}, \{t_i\}_{i\in[n]_0})$ where t_i are generated uniformly and independently at random.
- Generate $\mathsf{pp} \leftarrow \mathsf{SS.Gen}(1^\lambda)$.
- Generate $(\mathsf{ss.vk}^{(\epsilon)}, \cdot, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$.
- Set $\mathsf{mpk} := (\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)})$ and $\mathsf{msk} := (\mathsf{iO}(\mathsf{Node}(\cdot, s)), \mathsf{iO}(\mathsf{Leaf}(\cdot, s)))$.

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

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

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

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

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

- Parse $(\{(\sigma^{(v)}, x^{(v)})\}_v, \mathsf{nce.sk}^{(\mathsf{id})}) \leftarrow \mathsf{sk}_{\mathsf{id}}$ where $v \in \{\mathsf{id}[1,j]\}_{j\in[n-1]_0}$.
- Set $\mathsf{vk}^{(0)} := \mathsf{ss.vk}^{(\epsilon)}$.
- For $k = 0$ to $n-1$,
  1. Compute $\{\mathsf{ss.ct}_{i,j}\} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{Q}}^{(k)}, \{\mathsf{lab}_i^{(k)}\}_{i\in[\lambda]})$.
  2. Set $\mathsf{vk}^{(k+1)} := x^{(k)}[\mathsf{id}[k+1]]$ where $x^{(k)}$ is a 2 block of λ length strings.
  3. Computes $\mathsf{lab}_i^{(k+1)} \leftarrow \mathsf{SS.Dec}(\mathsf{pp}, \mathsf{vk}^{(k)}, (x^{(k)}, \sigma^{(k)}), \mathsf{ss.ct}_{i,\mathsf{vk}^{(k+1)}[i]})$, for all $i \in [\lambda]$.
- Compute $\mathsf{ct} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{T}}, \{\mathsf{lab}_i^{(n)}\}_{i\in[\lambda]})$.
- Return $m \leftarrow \mathsf{NCE.Dec}(\mathsf{nce.sk}^{(\mathsf{id})}, \mathsf{ct})$.

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

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

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

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

- Parse $(\mathsf{mpk}, s, \mathsf{nce.st}^{(\mathsf{id}^*)}) \leftarrow \mathsf{st}_1$.
- For $j = 0$ to $n-2$, it compute $(\cdot, x^{(\mathsf{id}[1,j])}, \sigma^{(\mathsf{id}[1,j])}) \leftarrow \mathsf{Node}(\mathsf{id}[1,j], s)$.
- If id^* is not a sibling of id, compute $(\cdot, x^{(\mathsf{id}[1,n-1])}, \sigma^{(\mathsf{id}[1,n-1])}, \{\, \mathsf{nce.sk}^{(\mathsf{id}[1,n-1] \| b)} \,\}_{b \in \{0,1\}}) = \mathsf{Leaf}(\mathsf{id}[1, n-1], s)$.
- Else, generate $(\mathsf{ss.vk}^{(\mathsf{id}[1,n-1])}, \mathsf{ss.sk}^{(\mathsf{id}[1,n-1])}) \leftarrow \mathsf{SS.Setup}(1^\lambda; r^{(\mathsf{id}[1,n-1])})$ and $(\mathsf{nce.pk}^{(\mathsf{id})}, \mathsf{nce.sk}^{(\mathsf{id})}) \leftarrow \mathsf{NCE.Setup}(1^\lambda; \mathsf{PPRF.Eval}(s, \mathsf{id}))$ and sets $x^{(\mathsf{id}[1,n-1])}$ using $\mathsf{nce.pk}^{(\mathsf{id})}$ and $\mathsf{nce.pk}^{(\mathsf{id}^*)}$ and generates $\sigma^{(\mathsf{id}[1,n-1])}$.
- Return $(\{(\sigma^{(v)}, x^{(v)})\}_v, \mathsf{nce.sk}^{(\mathsf{id})})$ where $v \in \{\, \mathsf{id}[1,j] \,\}_{j \in [n-1]_0}$.
- Set $\mathsf{st}_2 = \mathsf{st}_1$.

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

- Parse $(\mathsf{mpk}, s, \mathsf{nce.st}^{(\mathsf{id}^*)}) \leftarrow \mathsf{st}_1$.
- Compute $(\mathsf{nce}.r_{\mathsf{Setup}}, \mathsf{nce}.r_{\mathsf{Enc}}) \leftarrow \mathsf{NCE.Sim}_2(\mathsf{nce.st}^{(\mathsf{id}^*)}, m^*)$.
- Set $r^{(\mathsf{id}^*)} = \mathsf{nce}.r_{\mathsf{Setup}}$.
- $s^* \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\, \mathsf{id}^*[1,k] \,\}_{k \in [n]_0}, \{\, r^{(\mathsf{id}^*[1,k])} \,\}_{k \in [n]_0})$.
- Return $\mathsf{msk} = (\mathsf{iO}(\mathsf{Node}(\cdot, s^*)), \mathsf{iO}(\mathsf{Leaf}(\cdot, s^*)))$.

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 size of the master secret key and secret keys are of the form $\mathrm{poly}(n, \lambda)$.

Correctness. For a well-formed ciphertext $\mathsf{ct} := (\{\, \mathsf{lab}^{(0)}_{i, \mathsf{mpk}[i]} \,\}, \{\, \tilde{\mathcal{Q}}^{(k)} \,\}_{k \in [n-1]_0}, \tilde{\mathcal{T}})$ and secret key $(\{(\sigma^{(v)}, x^{(v)})\}_v, \mathsf{nce.sk}^{(\mathsf{id})})$, we have

$$\tilde{\mathcal{Q}}^{(0)}(\{\, \mathsf{lab}^{(0)}_{i, \mathsf{ss.vk}^{(\epsilon)}[i]} \,\}_i) \to \{\mathsf{ss.ct}_{i,b} = \mathsf{SS.Enc}(\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)}, \mathsf{id}[1] \cdot \lambda + i, b, \mathsf{lab}^{(1)}_{i,b})\}_{i \in [\lambda], b \in \{0,1\}}$$

due to the correctness of the garbling scheme. From the correctness of the OTSE scheme, we have

$$\mathsf{lab}^{(1)}_{i, \mathsf{ss.vk}^{(1)}[i]} = \mathsf{SS.Dec}(\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)}, x^{(\epsilon)}, \mathsf{ss.ct}_{i, \mathsf{ss.vk}^{(1)}[i]})$$

Repeating this process, we obtain the labels $\{\, \mathsf{lab}^{(n)}_{i, \mathsf{nce.pk}^{(\mathsf{id})}[i]} \,\}_i$ for the garbled circuit $\tilde{\mathcal{T}}$. Again, due to correctness of the garbling scheme, we have

$$\tilde{\mathcal{T}}(\{\, \mathsf{lab}^{(n)}_{i, \mathsf{nce.pk}^{(\mathsf{id})}[i]} \,\}_i) \to \mathsf{nce.ct} = \mathsf{NCE.Enc}(\mathsf{nce.pk}^{(\mathsf{id})}, m)$$

and due to the correctness of the NCE scheme, we get $m = \mathsf{NCE.Dec}(\mathsf{nce.sk}^{(\mathsf{id})}, \mathsf{nce.ct})$.

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

Proof. We now proceed to sketch the proof that the above scheme is a secret NC-IBE scheme. Let $p_{\mathcal{A},\mathsf{Hyb}_i}$ denote probability of $\mathcal{A}$ outputting $b' = 0$ in Hybrid Hyb_i. We will show that this probability is almost the same in every hybrid game.

Hybrid Hyb_0 : This is the original NC-IBE game. This hybrid will also be denoted as $H_{1,-1,4}$.

1. $\mathcal{A}$ sends target identity id^* of length n.
2. The challenger generates $\mathsf{mpk}, \mathsf{msk}$ as follows.
   - Generate $\mathsf{pprf.msk} \leftarrow \mathsf{PPRF.Setup}(1^\lambda, 1^{n+1})$.
   - Compute $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{0^i\}_{i\in[n]_0}, \{t_i\}_{i\in[n]_0})$ where t_i are generated uniformly and independently at random.
   - Generate $\mathsf{pp} \leftarrow \mathsf{SS.Gen}(1^\lambda)$.
   - Generate $(\mathsf{ss.vk}^{(\epsilon)}, \cdot, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$.
   - Set $\mathsf{msk} := (\mathsf{iO}(\mathsf{Node}(\cdot, s)), \mathsf{iO}(\mathsf{Leaf}(\cdot, s)))$ and $\mathsf{mpk} := (\mathsf{pp}, \mathsf{ss.vk}^{(\epsilon)})$.

   The challenger sends mpk to $\mathcal{A}$.
3. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query id^j as an input, it returns $\mathsf{sk}_{\mathsf{id}^j}$ as follows.
   - Parse $(C_{\mathsf{Node}}, C_{\mathsf{Leaf}}) \leftarrow \mathsf{msk}$.
   - For $j = 0$ to $n-2$, compute $(\cdot, x^{(\mathsf{id}[1,j])}, \sigma^{(\mathsf{id}[1,j])}) = C_{\mathsf{Node}}(\mathsf{id}[1,j])$.
   - Compute $(\cdot, x^{(\mathsf{id}[1,n-1])}, \sigma^{(\mathsf{id}[1,n-1])}, \{\mathsf{nce.sk}^{(\mathsf{id}[1,n-1]\|b)}\}_{b\in\{0,1\}}) = C_{\mathsf{Leaf}}(\mathsf{id}[1,n-1])$.
   - Set $\mathsf{sk}_{\mathsf{id}} := (\{(\sigma^{(v)}, x^{(v)})\}_v, \mathsf{nce.sk}^{(\mathsf{id})})$ where $v \in \{\mathsf{id}[1,j]\}_{j\in[n-1]_0}$.
4. $\mathcal{A}$ outputs id^*, m^*. The challenger generates ct^* as follows.
   - Generate $(\tilde{\mathcal{T}}, \{\mathsf{lab}_{i,j}^{(n)}\}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, T_{m^*})$.
   - For $k = n-1$ to 0, generate $(\tilde{\mathcal{Q}}^{(k)}, \{\mathsf{lab}_{i,j}^{(k)}\}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, Q_{\mathsf{id}^*[k+1],\{\mathsf{lab}_{i,j}^{(k+1)}\}})$.
   - Set $\mathsf{ct}^* := (\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}\}, \{\tilde{\mathcal{Q}}^{(k)}\}_{k\in[n-1]_0}, \tilde{\mathcal{T}})$.

   The challenger sends $(\mathsf{ct}^*, \mathsf{msk})$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

For $k = 0$ to $n-2$,

Hybrid $\mathsf{Hyb}_{1,k,0}$: In this hybrid, $(\tilde{\mathcal{Q}}^{(k)}, \{\mathsf{lab}_i^{(k)}\})$ are generated by the simulation algorithm on $Q_{\mathsf{id}^*[k+1],\{\mathsf{lab}_{i,j}^{(k+1)}\}}(\mathsf{ss.vk}^{(\mathsf{id}^*[1,k])})$.

We have $|p_{\mathcal{A},\mathsf{Hyb}_{1,1,0}} - p_{\mathcal{A},\mathsf{Hyb}_0}| = \mathsf{negl}(\lambda)$ due the security of the garbling scheme. This is because only $\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}^{(\epsilon)}[i]}\}$ is present in the ciphertext.

Lemma 7.2. *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* $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{1,k,0}} - p_{\mathcal{A},H_{1,k-1,4}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,1}$: In this hybrid, instead of generating $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[k-1]_0} \cup \{0^j\}_{j\in[k,n]}, \{t_i\}_{i\in[n]_0})$, it generates the key $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[k]_0} \cup \{0^j\}_{j\in[k+1,n]}, \{t_i\}_{i\in[n]_0})$.

We have $|p_{\mathcal{A},H_{1,k,1}} - p_{\mathcal{A},H_{1,k,0}}| \leq \mathsf{negl}(\lambda)$ due to the privacy security of the PPRF.

Lemma 7.3. *Assume that* PPRF *is a secure programmable PRF scheme, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{1,k,1}} - p_{\mathcal{A},H_{1,k,0}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,2}$: In this hybrid, msk is changed from $\mathsf{iO}(\mathsf{Node}(\cdot, s))$ to the following program P. P has a randomly generated $(\mathsf{vk}, \mathsf{sk})$ and $\mathsf{Node}(\cdot, s)$ hardcoded. When $v \neq \mathsf{id}^*[1,k]$, it runs $\mathsf{Node}(\cdot, s)$. Else, it will won't use s to get the keys rather use $(\mathsf{vk}, \mathsf{sk})$ directly.

To show that this hybrid is indistinguishable from the previous hybrid, we need to demonstrate that the circuits in both hybrids are functionally the same. In $\mathsf{Hyb}_{1,k,1}$, the circuit is $\mathsf{Node}(\cdot, s)$, where s at position $\mathsf{id}^*[1,j]$ is programmed with a truly random value. As a result, the OTSE keys are generated using this truly random value. In $\mathsf{Hyb}_{1,k,2}$, the circuit also uses $\mathsf{Node}(\cdot, s)$, but at $\mathsf{id}^*[1,j]$ it employs hard-coded values $(\mathsf{vk}, \mathsf{sk})$, which are themselves are generated using truly random coins. Therefore, in both cases, the keys related to $\mathsf{id}^*[1,j]$ are generated from truly random values, making the circuits equivalent.

Lemma 7.4. *Assume that* iO *is a secure indistinguishable obfuscation, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{1,k,2}} - p_{\mathcal{A},H_{1,k,1}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,3}$: In this hybrid, $\mathsf{lab}^{(k+1)}_{i,1-\mathsf{vk}^{k+1}[i]}$ is replaced $\mathsf{lab}^{(k+1)}_{i,\mathsf{vk}^{k+1}[i]}$.

We have $|p_{\mathcal{A},H_{1,k,3}} - p_{\mathcal{A},H_{1,k,2}}| \leq \mathsf{negl}(\lambda)$ due to the OTSE security. Note that the program P must have vk and sk hardcoded. While vk is provided by the OTSE challenger, sk is not. However, sk is only required when P is called on $\mathsf{id}^*[1:k]$ to generate $\sigma^{(\mathsf{id}^*[1:j])}$. This signature can be obtained from the OTSE challenger and hardcoded into P, eliminating the need to hardcode sk.

Lemma 7.5. *Assume that* SS *is a secure one-time signature encryption scheme, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{1,k,3}} - p_{\mathcal{A},H_{1,k,2}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,k,4}$: In this hybrid, msk is revert back to $\mathsf{iO}(\mathsf{Node}(\cdot, s))$.

Lemma 7.6. *Assume that* iO *is a secure indistinguishable obfuscation, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{1,k,4}} - p_{\mathcal{A},H_{1,k,3}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{1,n-1,0}$: In this hybrid, $(\tilde{\mathcal{Q}}^{(n-1)}, \{\, \mathsf{lab}^{(n-1)}_i \,\})$ are generated by the simulation algorithm on $Q_{\mathsf{id}^*[n],\{\, \mathsf{lab}^{(n)}_{i,j} \,\}}($ $\mathsf{ss.vk}^{(\mathsf{id}^*[1,n-1])})$.

Hybrid $\mathsf{Hyb}_{1,n-1,1}$: In this hybrid, instead of generating $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[n-2]_0} \cup \{\, 0^j \,\}_{j\in[n-1,n]}$, $\{\, t_i \,\}_{i\in[n]_0})$, it generates $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[n-1]_0} \cup \{\, 0^n \,\}, \{\, t_i \,\}_{i\in[n]_0})$.

Hybrid $\mathsf{Hyb}_{1,n-1,2}$: In this hybrid, msk is changed from $\mathsf{iO}(\mathsf{Leaf}(\cdot, s))$ to the following program P. P has a randomly generated $(\mathsf{vk}, \mathsf{sk})$ and $\mathsf{Leaf}(\cdot, s)$ hardcoded. When $v \neq \mathsf{id}^*[1, n-1]$, it runs $\mathsf{Leaf}(\cdot, s)$. Else, it will won't use s to get the keys rather use $(\mathsf{vk}, \mathsf{sk})$ directly.

Hybrid $\mathsf{Hyb}_{1,n-1,3}$: In this hybrid, $\mathsf{lab}^{(n)}_{i,1-\mathsf{nce.pk}^{\mathsf{id}^*}[i]}$ is replaced $\mathsf{lab}^{(n)}_{i,\mathsf{nce.pk}^{\mathsf{id}^*}[i]}$.

Hybrid $\mathsf{Hyb}_{1,n-1,4}$: In this hybrid, msk is revert back to $\mathsf{iO}(\mathsf{Leaf}(\cdot, s))$.

Hybrid $\mathsf{Hyb}_{2,0}$: In this hybrid, $(\tilde{\mathcal{T}}, \{\, \mathsf{lab}_{i,\mathsf{nce.pk}_i^{(\mathsf{id}^*)}} \,\})$ are generated by the simulation algorithm on $T_m(\mathsf{nce.pk}^{(\mathsf{id}^*)})$.

Lemma 7.7. *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* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_{2,0}} - p_{\mathcal{A},H_{1,n-1,4}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{2,1}$: In this hybrid, instead of generating $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[n-1]_0} \cup \{0^n\}, \{t_i\}_{i\in[n]_0})$, it generates $s \leftarrow \mathsf{PPRF.Program}(\mathsf{pprf.msk}, \{\mathsf{id}^*[1,j]\}_{j\in[n]_0}, \{t_i\}_{i\in[n]_0})$.

Lemma 7.8. *Assume that* PPRF *is a secure programmable PRF scheme, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_{2,1}} - p_{\mathcal{A},H_{2,0}}| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathsf{Hyb}_{2,2}$: In this hybrid, $(\mathsf{nce.pk}^{(\mathsf{id}^*)}, \mathsf{nce.sk}^{(\mathsf{id}^*)}, \mathsf{nce.ct}^{(\mathsf{id}^*)})$ are all simulated using the NCE simulators. And msk is generated at the end after obtaining the setup randomness from the NCE simulator and programming pprf.msk appropriately.

Lemma 7.9. *Assume that* NCE *is a secure non-committing encryption scheme, then for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_{2,2}} - p_{\mathcal{A},H_{2,1}}| \leq \mathsf{negl}(\lambda)$.

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_0} - p_{\mathcal{A},H_{2,2}}| \leq \mathsf{negl}(\lambda)$.

□

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

8 Rate-1 Strongly Incompressible ABE/IBE from RNC-AB/IB-KEM

In this section, we show a construction for strongly incompressible IBE using an RNC-IB-KEM and an incompressible SKE scheme. Since the construction follows a similar approach in the ABE setting, we omit it here. 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 IncSKE.Setup outputs a truly random string.

$\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm takes as input the security parameter λ and the upper bound for the state bound S. It computes $(\mathsf{ibkem.mpk}, \mathsf{ibkem.msk}) \leftarrow \mathsf{IBKEM.Setup}(1^\lambda, 1^{|\mathsf{inc.sk}|})$ 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 message m. It generates $(\mathsf{ibkem.ct}, \mathsf{seskey}) \leftarrow \mathsf{IBKEM.Encap}(\mathsf{ibkem.mpk}, \mathsf{id})$ and sets $\mathsf{inc.sk} := \mathsf{seskey}$. It then 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 $\mathsf{sk} = \mathsf{ibkem.sk}$ and a ciphertext $\mathsf{ct} = (\mathsf{ibkem.ct}, \mathsf{inc.ct})$. It first computes $\mathsf{inc.sk} \leftarrow \mathsf{IBKEM.Decap}(\mathsf{ibkem.sk}, \mathsf{ibkem.ct})$. Then, it computes $m \leftarrow \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})$. It returns m.

Correctness. The correctness is straight-forward from the correctness of NC-IBE scheme and 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 NC-IBE scheme generates ciphertext whose size is independent of the session key, we have $|\mathsf{ibkem.ct}| = \mathrm{poly}(\lambda)$. Therefore, the rate of the incompressible IBE scheme is $1 - o(1)$.

Theorem 8.1. *Assuming that* IBKEM *is a secure RNC-IB-KEM scheme and* IncSKE *is a incompressible SKE scheme, the above construction is a secure strongly incompressible IBE scheme.*

Proof sketch. We will show that the construction is secure using a sequence of hybrid arguments.

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

Hybrid H_1: In this game, the IBKEM scheme is changed to simulation mode. To be precise, the first simulator is used generate the master public key $\mathsf{ibkem.mpk}$. The second simulator will be used to handle the key queries. The third simulator will be invoked on id^* to produce the simulated ciphertext $\mathsf{ibkem.ct}$. Finally, for a randomly generated $\mathsf{inc.sk}$, the fourth simulator on input $\mathsf{inc.sk}$ will produce the master secret key $\mathsf{ibkem.msk}$. The indistinguishability of H_0 from H_1 follows directly from the security of the RNC-IBE.

We now argue that there is no PPT adversary that can win in H_1 with non-negligible probability. This follows from the security of the incompressible SKE scheme. This is because an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that wins in H_1 can be used to build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the security of the underlying incompressible SKE as follows. $\mathcal{B}$ can simulate H_1 using the first two IBKEM simulators upto the challenge phase. On receiving (m_0, m_1) from $\mathcal{A}_1$, it will forward it to the SKE challenger and receives $\mathsf{inc.ct}^*$. It will relay $(\mathsf{ibkem.ct}, \mathsf{inc.ct}^*)$ where $\mathsf{ibkem.ct}$ is generated by the third IBKEM simulator. In the second phase, it will receive $\mathsf{inc.sk}$ from the SKE challenger and will use the fourth IBKEM simulator will generate $\mathsf{ibkem.msk}$. □

Combining Theorem 8.1, Theorem 3.7, Theorem 10.2, we obtain Theorem 1.2.

Remark 8.2. We highlight that a similar approach can be employed to construct incompressible IBE from RNC-IBE and incompressible SKE. 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 $\mathsf{inc.sk}$. Therefore, combining Theorem 1.3 and Theorem 3.8, we get Theorem 1.4.

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, 5, 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. 5

[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, 11

[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, 5

[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, 5

[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, 9, 21, 43

[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. 5

[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. 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, 5

[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, 14

[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. 5

[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. 15

[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

[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, 5

[HOR15] Brett Hemenway, Rafail Ostrovsky, and Alon Rosen. Non-committing encryption from ϕ-hiding. In *Theory of Cryptography Conference*, pages 591–608. Springer, 2015. 5

[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, 5

[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. 5

[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, 21, 43

[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, 9, 21

[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, 5

[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. 5, 12

9 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 a predicate such that $P(x,y) = 1 \iff x = y$. 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 $x = y$, $\mathsf{sE}(x, (\mathbf{w}_1, \mathbf{w}_2)) = \mathsf{rE}(y, (\mathbf{w}_1, \mathbf{w}_2))$. With regards to α-privacy, when $x \neq y$, $(\mathbf{w}_1 + x\mathbf{w}_2, \mathbf{w}_1 + y\mathbf{w}_2)$ are pairwise independent.

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

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

10.1 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 $\ell := \ell(\lambda)$ 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 to write it.

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

- Parse $([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2) \leftarrow \mathsf{pp}$.

- Generate $h \leftarrow \{0,1\}^{\log(p)}$.
- Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2, \forall i \in [\ell]$.
- Output $\mathsf{mpk} := (\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_{i\in[\ell]}, h)$ and $\mathsf{msk} := \{ \mathbf{k}_i \}_{i\in[\ell]}$.

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

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

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

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

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

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

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

- Generate $s_i, w_i \leftarrow \mathbb{Z}_p, \forall i \in [\ell]$.
- Output $\mathsf{sk}_{\mathsf{id}} := \{ ([s_i\mathbf{b}]_2, [\frac{k_1}{|\mathbf{a}|^2}\cdot\mathbf{a} + s_i(\mathbf{W}_1 + \mathsf{id}\cdot\mathbf{W}_2)^\top\mathbf{b} + w_i\mathbf{a}^\perp]_2) \}_{i\in[\ell]}$ and $\mathsf{st}_2 := \mathsf{st}_1$.

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

- Generate $u_1, u_2 \leftarrow \mathbb{Z}_p$ and set $\mathbf{u} = u_1\mathbf{a} + u_2\mathbf{a}^\perp$.
- Outputs $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^*\cdot\mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{st}_3 := (\mathsf{st}_2, u_1, u_2)$.

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

- Generate $x_i \in \mathbb{Z}_p$ such that $\mathsf{seskey}_i = \mathsf{HC}([x_i]_T, h)$ via rejection sampling.
- Set $k_{i,2} := \frac{x_i - u_1 k_{i,1}}{u_2}$.
- Output $\mathsf{msk} := \{ \frac{k_{i,1}}{|\mathbf{a}|^2}\cdot\mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2}\cdot\mathbf{a}^\perp \}_{i\in[\ell]}$.

Remark 10.1. The randomness used in the setup algorithm includes h, which is part of the master public key and the set $\{ k_i \}$, which constitutes the entire master secret key. Since, the challenger outputs h in the initialization phase and provides the master secret key in the challenge, the adversary effectively gains access to all the randomness used in the setup algorithm during the challenge phase.

Parameters. The size of a ciphertext is $\mathrm{poly}(\lambda)$, i.e., it is independent of ℓ. Whereas, the size of the master public key, master secret key and 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 a deterministic function, the correctness follows immediately.

Theorem 10.2. *Assuming the hardness of* SXDH*, the above scheme is a secure receiver non-committing identity-based key encapsulation mechanism.*

Proof. Consider the following experiment for an adversary $\mathcal{A}$ that makes q queries to $O_{\mathsf{UserKeyGen}}$.

Hyb_0**:** This corresponds to the real experiment of non-committing security.

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

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle that provides access to $\mathsf{Keygen}(\mathsf{pp}, \mathsf{msk}, \cdot)$.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query $\mathsf{id}^j \in \mathbb{Z}_p$ as an input, it returns $\mathsf{sk}_{\mathsf{id}^j}$ generated as follows.
   - Generate $s_i^j \leftarrow \mathbb{Z}_p$, $\forall i \in [\ell]$.
   - Set $\mathsf{sk}_{\mathsf{id}^j} := \{\, ([s_i^j\mathbf{b}]_2, [\mathbf{k}_i + s_i^j(\mathbf{W}_1 + \mathsf{id}^j\cdot\mathbf{W}_2)^\top\mathbf{b}]_2)\,\}_i$.
3. $\mathcal{A}$ outputs $\mathsf{id}^* \in \mathbb{Z}_p$. 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 $\mathsf{seskey}^* := \{\, \mathsf{HC}([r\mathbf{a}^\top\mathbf{k}_i]_T, h)\,\}_i$.

   The challenger sends $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Using a sequence of hybrid experiments, we will prove that the experiment can be indistinguishably changed into non-committing experiment. We will denote the probability that $\mathcal{A}$ outputs 0 in the hybrid Hyb_i using $p_{\mathcal{A},H_i}$.

Below, we assume that $\mathbf{a}$ and $\mathbf{b}^\perp$ are linearly independent and $\mathbf{a}^\perp$ and $\mathbf{b}$ are also linearly independent, which hold 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 $(\mathsf{ct}^*, \mathsf{seskey}^*)$ is generated as $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^*\cdot\mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{\, \mathsf{HC}([\mathbf{u}^\top\mathbf{k}_i]_T, h)\,\}_i$, 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 10.3. *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 $\mathsf{Hyb}_{1,0,5}$ as Hyb_1.

Changing the user secret keys into semi-functional mode. We change the user secret keys into semi-functional mode using $\mathsf{Hyb}_{1,i,1}, \cdots, \mathsf{Hyb}_{1,i,5}$ for $i \in [q]$. Below, we 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.

$O_{\mathsf{UserKeyGen}}(\mathsf{id})$**:** Given the j-th query $\mathsf{id}^j \in \mathbb{Z}_p$ as an input, it behaves as follows.

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

We have $|p_{\mathcal{A},H_{1,1,1}} - p_{\mathcal{A},H_1}| = \mathsf{negl}(\lambda)$ from the DDH assumption on $\mathbb{G}_2$.

Lemma 10.4. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that, $|p_{\mathcal{A},H_{1,1,1}} - p_{\mathcal{A},H_1}| \leq \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

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 distribution of $\mathbf{W}_1, \mathbf{W}_2$ do not change.

Lemma 10.5. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_{1,i,2}} - p_{\mathcal{A},H_{1,i,1}}| = 0$.*

We prove that w_1 and w_2 appears only in ct^* in the form of $w_1 + \mathsf{id}^* \cdot w_2$ and in $\mathsf{sk}_{\mathsf{id}^i}$ in the form of $w_1 + \mathsf{id}^i \cdot w_2$. First, we have $[\mathbf{W}_\alpha \mathbf{a}]_1 = [\hat{\mathbf{W}}_\alpha \mathbf{a}]_1$ and $[\mathbf{W}_\alpha^\top \mathbf{b}]_2 = [\hat{\mathbf{W}}_\alpha^\top \mathbf{b}]_2$ for $\alpha \in \{1,2\}$. For ct^*, we can write $\mathbf{u} = r\mathbf{a} + r'\mathbf{b}^\perp$, and thus we have

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

Also, for $\mathsf{sk}_{\mathsf{id}^i}$, we can write $\mathbf{v}_d^i = s_d^i \mathbf{b} + t_d^i \mathbf{a}^\perp$ and thus we have

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

Moreover, for $j \neq i$, we can write $\mathsf{sk}_{\mathsf{id}^j}$ as

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

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

It is important to note that only the secret key $\mathsf{sk}_{\mathsf{id}^i}$ contains information about w_1 and w_2, while the remaining secret keys do not. Therefore, we have $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$, since $\mathsf{id}^* \neq \mathsf{id}^i$, which implies

$$\{ w_1 + \mathsf{id}^* \cdot w_2, w_1 + \mathsf{id}^i \cdot w_2 \}_{w_1,w_2} \equiv \{ w_1 + \mathsf{id}^* \cdot w_2, u \}_{w_1,w_2,u}$$

and with probability $\frac{p-1}{p}$, a randomly chosen t_d^i is invertible.

Lemma 10.6. *For all PPT adversaries $\mathcal{A}$ and $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_{1,i,3}} - p_{\mathcal{A},H_{1,i,2}}| = \mathsf{negl}(\lambda)$.*

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 $\mathsf{Hyb}_{1,i,2}$. Namely, 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 10.7. *For all PPT adversaries $\mathcal{A}$, $|p_{\mathcal{A},H_{1,i,4}} - p_{\mathcal{A},H_{1,i,3}}| = 0$.*

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

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

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

Lemma 10.9. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that, $|p_{\mathcal{A},H_{1,i+1,1}} - p_{\mathcal{A},H_{1,i,5}}| \leq \ell \cdot p_{\mathcal{B},\mathsf{DDH}}$.*

Final steps towards non-committing mode.

Hybrid Hyb_2: We define Hyb_2 as the same game as $\mathsf{Hyb}_{1,q,5}$. The detailed description is as follows.

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 $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_q$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $\mathbf{k}_i \leftarrow \mathbb{Z}_p^2, \forall i \in [\ell]$ and $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$ and $\mathsf{mpk} := (\{ [\mathbf{a}^\top \mathbf{k}_i]_T \}_i, h)$ and $\mathsf{msk} := \{ \mathbf{k}_i \}_i$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query $\mathsf{id}^j \in \mathbb{Z}_p$ as an input, it returns $\mathsf{sk}_{\mathsf{id}^j}$ generated as follows.
   - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - Set $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\mathbf{k}_d + s_d^j (\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_d$.
3. $\mathcal{A}$ outputs $\mathsf{id}^* \in \mathbb{Z}_p$. The challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $\mathbf{u} \leftarrow \mathbb{Z}^2$.
   - Set $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([\mathbf{u}^\top \mathbf{k}_i]_T, h) \}_i$.

   The challenger sends $(\mathsf{msk}, \mathsf{ct}^*, \mathsf{seskey}^*)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Hybrid Hyb_3**:** This is the same as Hyb_2 except that we generate $k_{i,1}, k_{i,2} \leftarrow \mathbb{Z}_p$ and set $\mathbf{k}_i \leftarrow \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp$. By this change, we have $\mathsf{mpk} = \{ [k_{i,1}]_T \}_i$.

This is just conceptual change and we have $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.

Lemma 10.10. *For all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_2}| = 0$.*

Hybrid Hyb_4**:** This is the same as Hyb_3 except that for the any query id^j, $O_{\mathsf{UserKeyGen}}$ returns

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

, where $s_d^j, w_d^j \leftarrow \mathbb{Z}_p$.

We have $|p_{\mathcal{A},H_4} - p_{\mathcal{A},3}| = 0$ since $\frac{k_2}{|\mathbf{a}^\perp|^2} + w_d^j$ and w_d^j identically distributes for every $j \in [q], d \in [\ell]$ when w_d^j is chosen uniformly at random.

Lemma 10.11. *For all PPT adversaries $\mathcal{A}$ and for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_4} - p_{\mathcal{A},3}| = 0$.*

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

This is just conceptual change and we have $|p_{\mathcal{A},H_5} - p_{\mathcal{A},4}| = 0$.

Lemma 10.12. *For all PPT adversaries $\mathcal{A}$ and for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_5} - p_{\mathcal{A},4}| = 0$.*

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

We have $|p_{\mathcal{A},H_6} - p_{\mathcal{A},5}| \leq \mathsf{negl}(\lambda)$ since k_2 still distributes uniformly at random when u_2 is invertible which occurs with high probability.

Lemma 10.13. *For all PPT adversaries $\mathcal{A}$ and $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_6} - p_{\mathcal{A},5}| \leq \mathsf{negl}(\lambda)$.*

Hybrid Hyb_7**:** This is the same as Hyb_6 except that we defer the generation of k_2 until the challenge phase. The detailed description is as follows.

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 $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_q$ and $\mathbf{W}_1, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{2\times 2}$.
   - Generate $k_{i,1} \leftarrow \mathbb{Z}_p, \forall i \in [\ell]$ and $h \leftarrow \{0,1\}^{\log(p)}$.
   - Set $\mathsf{pp} := ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{W}_1\mathbf{a}]_1, [\mathbf{W}_2\mathbf{a}]_1, [\mathbf{W}_1^\top \mathbf{b}]_2, [\mathbf{W}_2^\top \mathbf{b}]_2)$ and $\mathsf{mpk} := (\{ [k_{i,1}]_T \}_i, h)$.

   The challenger sends pp and mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{UserKeyGen}}(\mathsf{id}^j)$**:** Given the j-query $\mathsf{id}^j \in \mathbb{Z}_p$ as an input, it returns $\mathsf{sk}_{\mathsf{id}^j}$ generated as follows.
   - Generate $s_d^j, w_d^j \leftarrow \mathbb{Z}_p, \forall d \in [\ell]$.
   - Set $\mathsf{sk}_{\mathsf{id}^j} := \{ ([s_d^j \mathbf{b}]_2, [\frac{k_1}{|\mathbf{a}|^2} \cdot \mathbf{a} + s_d^j (\mathbf{W}_1 + \mathsf{id}^j \cdot \mathbf{W}_2)^\top \mathbf{b} + w_d^j \mathbf{a}^\perp]_2) \}_d$.
3. $\mathcal{A}$ outputs $\mathsf{id}^* \in \mathbb{Z}_p$. The challenger generates $(\mathsf{ct}^*, \mathsf{seskey}^*)$ as follows.
   - Generate $u_1, u_2 \leftarrow \mathbb{Z}_p$ and set $\mathbf{u} = u_1 \mathbf{a} + u_2 \mathbf{a}^\perp$.
   - Generate $x_i \leftarrow \mathbb{Z}_p, \forall i \in [\ell]$

- Set $\mathsf{ct}^* := ([\mathbf{u}]_1, [(\mathbf{W}_1 + \mathsf{id}^* \cdot \mathbf{W}_2)\mathbf{u}]_1)$ and $\mathsf{seskey}^* := \{ \mathsf{HC}([x_i]_T, h) \}_i$.
- Set $k_{i,2} = \frac{x_i - u_1 k_{i,1}}{u_2}$ and $\mathsf{msk} := \{ \frac{k_{i,1}}{|\mathbf{a}|^2} \cdot \mathbf{a} + \frac{k_{i,2}}{|\mathbf{a}^\perp|^2} \cdot \mathbf{a}^\perp \}_i$.

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

4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

It is easy to see that Hyb_7 can be simulated using the RNC-IB-KEM simulators. Using the above lemma along with triangular inequality, the theorem follows. □

Combining Theorem 4.4 and Theorem 10.2, we obtain the following.

Theorem 10.14. *Assuming hardness of* SXDH*, there exists secure RNC-IBE schemes.*