Non-Committing Encryption with Quasi-Optimal Ciphertext-Rate Based on the DDH Problem

Yusuke Yoshida[1(✉)], Fuyuki Kitagawa[2], and Keisuke Tanaka[1]

[1] Tokyo Institute of Technology, Tokyo, Japan
yoshida.y.aw@m.titech.ac.jp, keisuke@is.titech.ac.jp
[2] NTT Secure Platform Laboratories, Tokyo, Japan
fuyuki.kitagawa.yh@hco.ntt.co.jp

Abstract. Non-committing encryption (NCE) was introduced by Canetti et al. (STOC '96). Informally, an encryption scheme is non-committing if it can generate a dummy ciphertext that is indistinguishable from a real one. The dummy ciphertext can be opened to any message later by producing a secret key and an encryption random coin which "explain" the ciphertext as an encryption of the message. Canetti et al. showed that NCE is a central tool to achieve multi-party computation protocols secure in the adaptive setting. An important measure of the efficiently of NCE is the ciphertext rate, that is the ciphertext length divided by the message length, and previous works studying NCE have focused on constructing NCE schemes with better ciphertext rates.

We propose an NCE scheme satisfying the ciphertext rate $\mathcal{O}(\log \lambda)$ based on the decisional Diffie-Hellman (DDH) problem, where λ is the security parameter. The proposed construction achieves the best ciphertext rate among existing constructions proposed in the plain model, that is, the model without using common reference strings. Previously to our work, an NCE scheme with the best ciphertext rate based on the DDH problem was the one proposed by Choi et al. (ASIACRYPT '09) that has ciphertext rate $\mathcal{O}(\lambda)$. Our construction of NCE is similar in spirit to that of the recent construction of the trapdoor function proposed by Garg and Hajiabadi (CRYPTO '18).

Keywords: Non-committing encryption · Decisional Diffie-Hellman problem · Chameleon encryption

1 Introduction

1.1 Background

Secure multi-party computation (MPC) allows a set of parties to compute a function of their inputs while maintaining the privacy of each party's input. Depending on when corrupted parties are determined, two types of adversarial settings called static and adaptive have been considered for MPC. In the static

S. D. Galbraith and S. Moriai (Eds.): ASIACRYPT 2019, LNCS 11923, pp. 128–158, 2019.
https://doi.org/10.1007/978-3-030-34618-8_5

setting, an adversary is required to declare which parties it corrupts before the protocol starts. On the other hand, in the adaptive setting, an adversary can choose which parties to corrupt on the fly, and thus the corruption pattern can depend on the messages exchanged during the protocol. Security guarantee in the adaptive setting is more desirable than that in the static setting since the former naturally captures adversarial behaviors in the real world while the latter is somewhat artificial.

In this work, we study *non-committing encryption (NCE)* which is introduced by Canetti, Feige, Goldreich, and Naor [4] and known as a central tool to achieve MPC protocols secure in the adaptive setting. NCE is an encryption scheme that has a special property called non-committing property. Informally, an encryption scheme is said to be non-committing if it can generate a dummy ciphertext that is indistinguishable from real ones, but can later be opened to any message by producing a secret key and an encryption random coin that "explain" the ciphertext as an encryption of the message. Cannetti et al. [4] showed how to create adaptively secure MPC protocols by instantiating the private channels in a statically secure MPC protocol with NCE.

Previous Constructions of NCE and their Ciphertext Rate. The ability to open a dummy ciphertext to any message is generally achieved at the price of efficiency. This is in contrast to ordinary public-key encryption for which we can easily obtain schemes the size of whose ciphertext is $n + \mathsf{poly}(\lambda)$ by using hybrid encryption methodology, where n is the length of an encrypted message and λ is the security parameter. The first NCE scheme proposed by Canetti et al. [4] only needs the optimal number of rounds (that is, two rounds), but it has ciphertexts of $O(\lambda^2)$-bits for every bit of an encrypted message. In other words, the ciphertext rate of their scheme is $O(\lambda^2)$, which is far from that of ordinary public-key encryption schemes. Subsequent works have focused on building NCE schemes with better efficiency.

Beaver [1] proposed a three-round NCE scheme with the ciphertext rate $\mathcal{O}(\lambda)$ based on the decisional Diffie-Hellman (DDH) problem. Damgård and Nielsen [8] generalized Beaver's scheme and achieved a three-round NCE scheme with ciphertext rate $O(\lambda)$ based on a primitive called simulatable PKE which in turn can be based on concrete problems such as the DDH, computational Diffie-Hellman (CDH), and learning with errors (LWE) problems. Choi, Dachman-Soled, Malkin, and Wee [7] further improved these results and constructed a two-round NCE scheme with ciphertext rate $\mathcal{O}(\lambda)$ based on a weaker variant of simulatable PKE called trapdoor simulatable PKE which can be constructed the factoring problem.

The first NCE scheme achieving a sub-linear ciphertext rate was proposed by Hemenway, Ostrovsky, and Rosen [20]. Their scheme needs only two rounds and achieves the ciphertext rate $\mathcal{O}(\log n)$ based on the ϕ-hiding problem which is related to (and generally believed to be easier than) the RSA problem, where n is the length of messages. Subsequently, Hemenway, Ostrovsky, Richelson, and Rosen [19] proposed a two-round NCE scheme with the ciphertext rate $\mathsf{poly}(\log \lambda)$ based on the LWE problem. Canetti, Poburinnaya, and Raykova [5]

Table 1. Comparison of existing NCE schemes. The security parameter is denoted by λ, and the message length n. Common-domain TDP can be instantiated based on the CDH and RSA problems. Simulatable and trapdoor simulatable PKE can be instantiated based on many computational problems realizing ordinary PKE. $^{(*)}$ This scheme uses common reference strings.

	Rounds	Ciphertext rate	Assumption
Canetti et al. [4]	2	$\mathcal{O}(\lambda^2)$	Common-domain TDP
Beaver [1]	3	$\mathcal{O}(\lambda)$	DDH
Damgård and Nielsen [8]	3	$\mathcal{O}(\lambda)$	Simulatable PKE
Choi et al. [7]	2	$\mathcal{O}(\lambda)$	Trapdoor simulatable PKE
Hemenway et al. [19]	2	$\mathsf{poly}(\log \lambda)$	LWE, Ring-LWE
Hemenway et al. [20]	2	$\mathcal{O}(\log n)$	Φ-hiding
Canetti et al. [5]$^{(*)}$	2	$1 + o(1)$	Indistinguishability obfuscation
This work	2	$\mathcal{O}(\log \lambda)$	DDH

showed that by using indistinguishability obfuscation, an NCE scheme with the asymptotically optimal ciphertext rate (that is, $1 + o(1)$) can be constructed. Their scheme needs only two rounds but was proposed in the common reference string model.

Despite the many previous efforts, as far as we know, we have only a single NCE scheme satisfying a sub-linear ciphertext rate based on widely and classically used problems, that is, the scheme proposed by Hemenway et al. [19] based on the LWE problem. Since NCE is an important cryptographic tool in constructing MPC protocols secure in the adaptive setting, it is desirable to have more constructions of NCE satisfying a better ciphertext rate.

1.2 Our Contribution

We propose an NCE scheme satisfying the ciphertext rate $\mathcal{O}(\log \lambda)$ based on the DDH problem. The proposed construction achieves the best ciphertext rate among existing constructions proposed in the plain model, that is, the model without using common reference strings. The proposed construction needs only two rounds, which is the optimal number of rounds for NCE. Previously to our work, an NCE scheme with the best ciphertext rate based on the DDH problem was the one proposed by Choi et al. [7] that satisfies the ciphertext rate $\mathcal{O}(\lambda)$. We summarize previous results on NCE and our result in Table 1.

We first show an NCE scheme that we call basic construction, which satisfies the ciphertext rate $\mathsf{poly}(\log \lambda)$. Then, we give our full construction satisfying the ciphertext rate $\mathcal{O}(\log \lambda)$ by extending the basic construction using error-correcting codes. Especially, in the full construction, we use a linear-rate error-correcting code which can correct errors of weight up to a certain constant proportion of the codeword length.

Our construction of NCE utilizes a variant of *chameleon encryption*. Chameleon encryption was originally introduced by Döttling and Garg [10] as an intermediate tool for constructing an identity-based encryption scheme based

on the CDH problem. Roughly speaking, chameleon encryption is public-key encryption in which we can use a hash value of a chameleon hash function and its pre-image as a public key and a secret key, respectively. We show a variant of chameleon encryption satisfying *oblivious samplability* can be used to construct an NCE scheme with a sub-linear ciphertext rate. Informally, oblivious samplability of chameleon encryption requires that a scheme can generate a dummy hash key obliviously to the corresponding trapdoor, and sample a dummy ciphertext that is indistinguishable from a real one, without using any randomness except the dummy ciphertext itself.

Need for the DDH Assumption. A key and a ciphertext of the CDH based chameleon encryption proposed by Döttling and Garg [10] together form multiple Diffie-Hellman tuples. Thus, it seems difficult to sample them obliviously unless we prove that the knowledge of exponent assumption [2,18] is false. In order to solve this issue, we rely on the DDH assumption instead of the CDH assumption. Under the DDH assumption, a hash key and a ciphertext of our chameleon encryption are indistinguishable from independent random group elements, and thus we can perform oblivious sampling of them by sampling random group elements directly from the underlying group.

Public Key Size. As noted above, we first give the basic construction satisfying the ciphertext rate $\mathsf{poly}(\log \lambda)$, and then extend it to the full construction satisfying the ciphertext rate $\mathcal{O}(\log \lambda)$. In addition to satisfying only the ciphertext rate $\mathsf{poly}(\log \lambda)$, the basic construction also has a drawback that its public key size depends on the length of a message quadratically.

A public key of the basic construction contains ciphertexts of our obliviously samplable chameleon encryption. The size of those ciphertexts is quadratic in the length of an input to the associated chameleon hash function similarly to the construction by Döttling and Garg [10]. Since the input length of the chameleon hash function is linear in the message length of the basic construction, the public key size of the basic construction depends on the message length quadratically.

Fortunately, we can remove this quadratic dependence by a simple block-wise encryption technique. Thus, in the full construction, we utilize such a block-wise encryption technique in addition to the error-correcting code. By doing so, we reduce not only the ciphertext rate to $\mathcal{O}(\log \lambda)$, but also the public key size to linear in the length of a message as in the previous constructions of NCE.

Relation with Trapdoor Function by Garg and Hajiabadi [14]. There has been a line of remarkable results shown by using variants of chameleon encryption, starting from the one by Cho, Döttling, Garg, Gupta, Miao, and Polychroniadou [6]. This includes results on identity-based encryption [3,9–11], secure MPC [6,16], adaptive garbling schemes [15,17], and so on. Garg and Hajiabadi [14] showed how to realize trapdoor function (TDF) based on the CDH problem using a variant of chameleon encryption called one-way function with encryption.[1]

[1] Their technique is further extended by Garg, Gay, and Hajiabadi [13] and Döttling, Garg, Ishai, Malavolta, Mour, and Ostrovsky [12].

Our construction of NCE can be seen as an extension of that of TDF by Garg and Hajiabadi. Our formulation of chameleon encryption is based on that of one-way function with encryption. Concretely, we define chameleon encryption so that it has *recyclability* introduced by Garg and Hajiabadi as a key property in their work.

1.3 Paper Organization

Hereafter, in Sect. 2, we first review the definition of NCE. Then, in Sect. 3, we provide high-level ideas behind our construction of NCE. In Sect. 4, we formally define and construct obliviously samplable chameleon encryption. In Sect. 5, using obliviously samplable chameleon encryption, we construct an NCE scheme that we call the basic construction satisfying the ciphertext rate $\mathsf{poly}(\log \lambda)$. Finally, in Sect. 6, we improve the basic construction and provide the full construction that achieves the ciphertext rate $\mathcal{O}(\log \lambda)$.

2 Preliminaries

Let PPT denote probabilistic polynomial time. In this paper, λ always denotes the security parameter. For a finite set X, we denote the uniform sampling of x from X by $x \xleftarrow{\$} X$. $y \leftarrow \mathsf{A}(x; r)$ denotes that given an input x, a PPT algorithm A runs with internal randomness r, and outputs y. A function f is said to be negligible if $f(\lambda) = 2^{-\omega(\lambda)}$, and we write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that f is negligible. Let $\mathsf{Ham}(\mathsf{x})$ denotes the Hamming weight of $\mathsf{x} \in \{0,1\}^n$. $\mathbb{E}[X]$ denotes expected value of X. $[n]$ denotes $\{1, \ldots, n\}$.

Lemma 1 (Chernoff bound). *For a binomial random variable X. If $\mathbb{E}[X] \leq \mu$, then for all $\delta > 0$, $\Pr[X \geq (1+\delta)\mu)] \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ holds.*

We provide the definition of the DDH assumption and its variants used in the proof of Theorem 1. We first introduce the leftover hash lemma.

Lemma 2 (Leftover hash lemma). *Let X and Y are sets. Let $\mathcal{H} := \{\mathsf{H} : X \to Y\}$ be a universal hash family. Then, the distributions $(\mathsf{H}, \mathsf{H}(x))$ and (H, y) are $\sqrt{\frac{|Y|}{4|X|}}$-close, where $\mathsf{H} \xleftarrow{\$} \mathcal{H}$, $x \xleftarrow{\$} X$, and $y \xleftarrow{\$} Y$.*

We review some computational assumptions. Below, we let $\mathbb{G}$ be a cyclic group of order p with a generator g. We also define the function $\mathsf{dh}(\cdot, \cdot)$ as $\mathsf{dh}(g^a, g^b) := g^{ab}$ for every $a, b \in \mathbb{Z}_p$. We start with the decisional Diffie-Hellman (DDH) assumption.

Definition 1 (Decisional Diffie-Hellman Assumption). *We say that the DDH assumption holds if for any PPT adversary $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(g_1, g_2, \mathsf{dh}(g_1, g_2)) = 1] - \Pr[\mathcal{A}(g_1, g_2, g_3) = 1]| = \mathsf{negl}(\lambda)$$

holds, where $g_1, g_2, g_3 \xleftarrow{\$} \mathbb{G}$.

We introduce a lemma that is useful for the proof of oblivious samplability of our chameleon encryption. We can prove this lemma by using the self reducibility of the DDH problem.

Lemma 3. *Let n be a polynomial of λ. Let $g_{i,b} \xleftarrow{\$} \mathbb{G}$ for every $i \in [n]$ and $b \in \{0,1\}$. We set $M := (g_{i,b})_{i\in[n],b\in\{0,1\}} \in \mathbb{G}^{2\times n}$.*

Then, if the DDH assumption holds, for any PPT adversary $\mathcal{A}$, we have

$$|\Pr[\mathcal{A}(M, M^{\rho}) = 1] - \Pr[\mathcal{A}(M, R) = 1]| = \mathsf{negl}(\lambda),$$

where $M^{\rho} = (g_{i,b}^{\rho})_{i\in[n],b\in\{0,1\}} \in \mathbb{G}^{2\times n}$ and $R \leftarrow \mathbb{G}^{2\times n}$.

We next define the hashed DDH assumption which is a variant of the DDH assumption.

Definition 2 (Hashed DDH Assumption). *Let $\mathcal{H} = \{\mathsf{H}_{\mathbb{G}} : \mathbb{G} \to \{0,1\}^{\ell}\}$ be a family of hash functions. We say that the hashed DDH assumption holds with respect to $\mathcal{H}$ if for any PPT adversary $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\mathsf{H}_{\mathbb{G}}, g_1, g_2, \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathsf{H}_{\mathbb{G}}, g_1, g_2, \mathbf{e}') = 1]| = \mathsf{negl}(\lambda)$$

holds, where $\mathsf{H}_{\mathbb{G}} \xleftarrow{\$} \mathcal{H}, g_1, g_2, \xleftarrow{\$} \mathbb{G}$, $\mathbf{e} = \mathsf{H}_{\mathbb{G}}(\mathsf{dh}(g_1, g_2))$, and $\mathbf{e}' \xleftarrow{\$} \{0,1\}^{\ell}$.

In this work, we use the hashed DDH assumption with respect to a hash family $\mathcal{H}$ whose output length ℓ is small enough such as $\ell = \mathsf{poly}(\log\lambda)$ or $\mathcal{O}(\log\lambda)$. In this case, by using a family of universal hash functions $\mathcal{H}$, we can reduce the hardness of the hashed DDH problem to that of the DDH problem by relying on the leftover hash lemma. Formally, we have the following lemma.

Lemma 4. *Let $\mathcal{H} = \{\mathsf{H}_{\mathbb{G}} : \mathbb{G} \to \{0,1\}^{\ell}\}$ be a family of universal hash functions, where $\ell = \mathsf{poly}(\log\lambda)$. Then, if the DDH assumption holds, the hashed DDH assumption with respect to $\mathcal{H}$ also holds by the leftover hash lemma.*

Non-Committing Encryption. A non-committing encryption (NCE) scheme is a public-key encryption scheme that has efficient simulator algorithms $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ satisfying the following properties. The simulator Sim_1 can generate a simulated public key pk and a simulated ciphertext CT. Later Sim_2 can explain the ciphertext CT as encryption of any plaintext. Concretely, given a plaintext m, Sim_2 can output a pair of random coins for key generation r^{Gen} and encryption r^{Enc}, as if pk was generated by the key generation algorithm with the random coin r^{Gen}, and CT is encryption of m with the random coin r^{Enc}.

Some previous works proposed NCE schemes that are three-round protocols. In this work, we focus on NCE that needs only two rounds, which is also called non-committing public-key encryption, and we use the term NCE to indicate it unless stated otherwise. Below, we introduce the definition of NCE according to Hemenway et al. [19].

Definition 3 (Non-Committing Encryption). *A non-committing encryption scheme* `NCE` *consists of the following PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim}_1, \mathsf{Sim}_2)$.

- $\mathsf{Gen}\left(1^\lambda; r^{\mathsf{Gen}}\right)$: *Given the security parameter* 1^λ*, using a random coin* r^{Gen}*, it outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Enc}\left(pk, m; r^{\mathsf{Enc}}\right)$: *Given a public key* pk *and a plaintext* $m \in \{0,1\}^\mu$*, using a random coin* r^{Enc}*, it outputs a ciphertext* CT*.*
- $\mathsf{Dec}\left(sk, CT\right)$: *Given a secret key* sk *and a ciphertext* CT*, it outputs* m *or* $\perp$*.*
- $\mathsf{Sim}_1\left(1^\lambda\right)$: *Given the security parameter* 1^λ*, it outputs a simulated public key* pk*, a simulated ciphertext* CT*, and an internal state* st*.*
- $\mathsf{Sim}_2\left(m, st\right)$: *Given a plaintext* m *and a state* st*, it outputs random coins for key generation* r^{Gen} *and encryption* r^{Enc}*.*

We require NCE *to satisfy the following correctness and security.*

Correctness. NCE *is called* γ*-correct, if for any plaintext* m*,*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r^{\mathsf{Gen}}\right), CT \leftarrow \mathsf{Enc}\left(pk, m; r^{\mathsf{Enc}}\right), \\ m' = \mathsf{Dec}\left(sk, CT\right); m = m'] \geq \gamma.$$

When $\gamma = 1 - \mathsf{negl}\left(\lambda\right)$*, we call it correct. Note that* γ *cannot be equal to* 1 *in the plain model (i.e., the model without using common reference strings).*

Security. *For any stateful PPT adversary* $\mathcal{A}$*, we define two experiments as follows.*

$\mathsf{Exp}^{Real}_{\mathtt{NCE},\mathcal{A}}$	$\mathsf{Exp}^{Ideal}_{\mathtt{NCE},\mathcal{A}}$
$(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r^{\mathsf{Gen}}\right)$	$(pk, CT, st) \leftarrow \mathsf{Sim}_1\left(1^\lambda\right)$
$m \leftarrow \mathcal{A}\left(pk\right)$	$m \leftarrow \mathcal{A}\left(pk\right)$
$CT \leftarrow \mathsf{Enc}\left(pk, m; r^{\mathsf{Enc}}\right)$	$\left(r^{\mathsf{Gen}}, r^{\mathsf{Enc}}\right) \leftarrow \mathsf{Sim}_2\left(m, st\right)$
$\mathsf{out} \leftarrow \mathcal{A}\left(CT, r^{\mathsf{Gen}}, r^{\mathsf{Enc}}\right)$	$\mathsf{out} \leftarrow \mathcal{A}\left(CT, r^{\mathsf{Gen}}, r^{\mathsf{Enc}}\right)$

We say that NCE *is secure if*

$$\mathsf{Adv}_{\mathtt{NCE},\mathcal{A}}\left(\lambda\right) := \left|\Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^{Real}_{\mathtt{NCE},\mathcal{A}}\right] - \Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^{Ideal}_{\mathtt{NCE},\mathcal{A}}\right]\right| = \mathsf{negl}\left(\lambda\right)$$

holds for every PPT adversary $\mathcal{A}$*.*

3 Ideas of Our Construction

In this section, we provide high-level ideas behind our construction of NCE.

As a starting point, we review the three-round NCE protocol proposed by Beaver [1], which contains a fundamental idea to build NCE from the DDH problem. Next, we show how to extend it and construct a two-round NCE scheme whose ciphertext rate is $\mathcal{O}\left(\lambda\right)$. Then, we show how to reduce the ciphertext rate to $\mathcal{O}\left(\log \lambda\right)$, and obtain our main result. Finally, we state that our resulting construction can be described by using a variant of chameleon encryption, and it can be seen as an extension of trapdoor function proposed by Garg and Hajiabadi [14].

Sender		Receiver
Input: $m \in \{0,1\}$		
$\mathsf{z} \xleftarrow{\$} \{0,1\}$		$\mathsf{x} \xleftarrow{\$} \{0,1\}$
$\rho_{\mathsf{z}} \xleftarrow{\$} \mathbb{Z}_p$		$\alpha_{\mathsf{x}} \xleftarrow{\$} \mathbb{Z}_p$
$A_{\mathsf{z}} = g^{\rho_{\mathsf{z}}}$	(A_0, A_1)	$B_{\mathsf{x}} = g^{\alpha_{\mathsf{x}}}$
$A_{1-\mathsf{z}} \xleftarrow{\$} \mathbb{G}$	$\longrightarrow$	$B_{1-\mathsf{x}} \xleftarrow{\$} \mathbb{G}$
	$(\mathbf{e}_0, \mathbf{e}_1), (B_0, B_1)$	$\mathbf{e}_{\mathsf{x}} = {A_{\mathsf{x}}}^{\alpha_{\mathsf{x}}}$
	$\longleftarrow$	$\mathbf{e}_{1-\mathsf{x}} \xleftarrow{\$} \mathbb{G}$
if ${B_{\mathsf{z}}}^{\rho_{\mathsf{z}}} = \mathbf{e}_{\mathsf{z}}$	w	
$w := \mathsf{z} \oplus m$	$\longrightarrow$	if $w \neq \perp$
else $w := \perp$		Output: $m = w \oplus \mathsf{x}$

Fig. 1. The description of Beaver's protocol [1].

3.1 Starting Point: Beaver's Protocol

Beaver's NCE protocol essentiality executes two Diffie-Hellman key exchange protocols in parallel. This protocol can send a 1-bit message. The ciphertext rate is $\mathcal{O}(\lambda)$. We describe the protocol below and in Fig. 1.

Step1. Let $\mathbb{G}$ be a group of order p with a generator g. The sender picks a random bit $\mathsf{z} \xleftarrow{\$} \{0,1\}$ and an exponent $\rho_{\mathsf{z}} \xleftarrow{\$} \mathbb{Z}_p$, and then sets $A_{\mathsf{z}} = g^{\rho_{\mathsf{z}}}$. The sender also generates a random group element $A_{1-\mathsf{z}} \xleftarrow{\$} \mathbb{G}$ *obliviously*, i.e., without knowing the discrete log of $A_{1-\mathsf{z}}$. The sender sends (A_0, A_1) to the receiver and stores the secret $sk = (\mathsf{z}, \rho_{\mathsf{z}})$. The random coin used in this step is $(\mathsf{z}, \rho_{\mathsf{z}}, A_{1-\mathsf{z}})$.

Step2. The receiver picks a random bit $\mathsf{x} \xleftarrow{\$} \{0,1\}$ and an exponent $\alpha_{\mathsf{x}} \xleftarrow{\$} \mathbb{Z}_p$, and then sets $B_{\mathsf{x}} = g^{\alpha_{\mathsf{x}}}$. The receiver also obliviously generates $B_{1-\mathsf{x}} \xleftarrow{\$} \mathbb{G}$. Moreover, the receiver computes $\mathbf{e}_{\mathsf{x}} = {A_{\mathsf{x}}}^{\alpha_{\mathsf{x}}}$ and obliviously samples $\mathbf{e}_{1-\mathsf{x}} \xleftarrow{\$} \mathbb{G}$. The receiver sends $((B_0, B_1), (\mathbf{e}_0, \mathbf{e}_1))$ to the sender. The random coin used in this step is $(\mathsf{x}, \alpha_{\mathsf{x}}, B_{1-\mathsf{x}}, \mathbf{e}_{1-\mathsf{x}})$.

Step3. The sender checks whether $\mathsf{x} = \mathsf{z}$ holds or not, by checking if ${B_{\mathsf{z}}}^{\rho_{\mathsf{z}}} = \mathbf{e}_{\mathsf{z}}$ holds. With overwhelming probability, this equation holds if and only if $\mathsf{x} = \mathsf{z}$. If $\mathsf{x} = \mathsf{z}$, the sender sends $w := \mathsf{z} \oplus m$, and otherwise quits the protocol.

Step4. The receiver recovers the message by $w \oplus \mathsf{x}$.

We next describe the simulator for this protocol.

Simulator. The simulator simulates a transcript $(A_0, A_1), ((B_0, B_1), (\mathbf{e}_0, \mathbf{e}_1))$, and w as follows. It generates $\rho_0, \rho_1, \alpha_0, \alpha_1 \xleftarrow{\$} \mathbb{Z}_p$ and sets

$$((A_0, A_1), (B_0, B_1), (\mathbf{e}_0, \mathbf{e}_1)) = ((g^{\rho_0}, g^{\rho_1}), (g^{\alpha_0}, g^{\alpha_1}), (g^{\rho_0 \alpha_0}, g^{\rho_1 \alpha_1})).$$

The simulator also generates $w \xleftarrow{\$} \{0,1\}$.

The simulator can later open this transcript to both messages 0 and 1. In other words, for both messages, the simulator can generate consistent sender and receiver random coins. For example, when opening it to $m = 0$, the simulator sets $\mathsf{x} = \mathsf{z} = w$, and outputs (w, ρ_w, A_{1-w}) and $(w, \alpha_w, B_{1-w}, \mathbf{e}_{1-w})$ as the sender's and receiver's opened random coins, respectively.

Security. Under the DDH assumption on $\mathbb{G}$, we can prove that any PPT adversary $\mathcal{A}$ cannot distinguish the pair of transcript and opened random coins generated in the real protocol from that generated by the simulator. The only difference of them is that $\mathbf{e}_{1-\mathsf{x}}$ is generated as a random group element in the real protocol, but it is generated as ${A_{1-\mathsf{x}}}^{\alpha_{1-\mathsf{x}}} = g^{\rho_{1-\mathsf{x}}\alpha_{1-\mathsf{x}}}$ in the simulation. When the real protocol proceeds to Step. 4, we have $\mathsf{x} = \mathsf{z}$ with overwhelming probability. Then, the random coins used by the sender and receiver (and thus given to $\mathcal{A}$) does not contain exponents of $A_{1-\mathsf{x}}$ and $B_{1-\mathsf{x}}$, that is, $\rho_{1-\mathsf{x}}$ and $\alpha_{1-\mathsf{x}}$. Thus, under the DDH assumption, $\mathcal{A}$ cannot distinguish randomly generated $\mathbf{e}_{1-\mathsf{x}} \xleftarrow{\$} \mathbb{G}$ from ${A_{1-\mathsf{x}}}^{\alpha_{1-\mathsf{x}}} = g^{\rho_{1-\mathsf{x}}\alpha_{1-\mathsf{x}}}$. Thus, this protocol is a secure NCE protocol.

This protocol succeeds in transmitting a message only when $\mathsf{z} = \mathsf{x}$, and otherwise it fails. Note that even when $\mathsf{z} \neq \mathsf{x}$, the protocol can transmit a message because in Step. 3, the sender knows the receiver's secret x. However, in that case, we cannot construct a successful simulator. In order to argue the security based on the DDH assumption, we have to ensure that either one pair of exponents (ρ_0, α_0) or (ρ_1, α_1) is not known to the adversary, but when $\mathsf{z} \neq \mathsf{x}$, we cannot ensure this.

Next, we show how to extend this protocol into a (two-round) NCE scheme and obtain an NCE scheme with the ciphertext rate $\mathcal{O}(\lambda)$.

3.2 Extension to Two-Round NCE Scheme

As a first attempt, we consider an NCE scheme $\mathtt{NCE}^1_{\mathtt{lin}}$ that is a natural extension of Beaver's three-round NCE protocol. Intuitively, $\mathtt{NCE}^1_{\mathtt{lin}}$ is Beaver's protocol in which the role of the sender and receiver is reversed, and the sender sends a message even when z and x are different. Specifically, the receiver generates the public key $pk = (A_0, A_1)$ and secret key (z, ρ_z), and the sender generates the ciphertext $CT = ((B_0, B_1), (\mathbf{e}_0, \mathbf{e}_1), w)$, where $(A_0, A_1), (B_0, B_1), (\mathbf{e}_0, \mathbf{e}_1)$, and $w := \mathsf{x} \oplus m$ are generated in the same way as those in Beaver's protocol. When decrypting the CT, the receiver first recovers the value of x by checking whether $B_{\mathsf{z}}^{\rho_{\mathsf{z}}} = \mathbf{e}_{\mathsf{z}}$ holds or not, and then computes $w \oplus \mathsf{x}$.

Of course, $\mathtt{NCE}^1_{\mathtt{lin}}$ is not a secure NCE scheme in the sense that we cannot construct a successful simulator when $\mathsf{z} \neq \mathsf{x}$ for a similar reason stated above. However, we can fix this problem and construct a secure NCE scheme by running multiple instances of $\mathtt{NCE}^1_{\mathtt{lin}}$.

In $\mathtt{NCE}^1_{\mathtt{lin}}$, if z coincides with x, we can construct a simulator similarly to Beaver's protocol, which happens with probability $\frac{1}{2}$. Thus, if we run multiple instances of it, we can construct simulators successfully for some fraction of them.

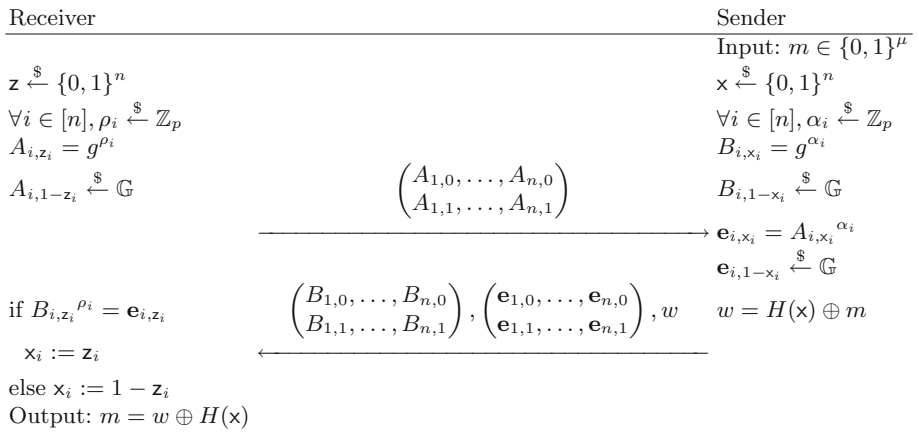

Fig. 2. The description of $\mathtt{NCE}_{\mathtt{lin}}$.

Based on this observation, we construct an NCE scheme $\mathtt{NCE}_{\mathtt{lin}}$ as follows. We also describe $\mathtt{NCE}_{\mathtt{lin}}$ in Fig. 2.

Let the length of messages be μ and $n = \mathcal{O}(\mu)$. We later specify the concrete relation of μ and n. The receiver first generates $\mathsf{z}_1 \cdots \mathsf{z}_n = \mathsf{z} \xleftarrow{\$} \{0,1\}^n$. Then, for every $i \in [n]$, the receiver generates a pubic key of $\mathtt{NCE}^1_{\mathtt{lin}}$, $(A_{i,0}, A_{i,1})$ in which the single bit randomness is z_i. We let the exponent of A_{i,z_i} be ρ_i, that is, $A_{i,\mathsf{z}_i} = g^{\rho_i}$. The receiver sends these n public keys of $\mathtt{NCE}^1_{\mathtt{lin}}$ as the public key of $\mathtt{NCE}_{\mathtt{lin}}$ to the sender. The secret key is $(\mathsf{z}, \rho_1, \dots, \rho_n)$.

When encrypting a message m, the sender first generates $\mathsf{x}_1 \cdots \mathsf{x}_n = \mathsf{x} \xleftarrow{\$} \{0,1\}^n$. Then, for every $i \in [n]$, the sender generates $((B_{i,0}, B_{i,1}), (\mathbf{e}_{i,0}, \mathbf{e}_{i,1}))$ in the same way as $\mathtt{NCE}^1_{\mathtt{lin}}$ (and thus Beaver's protocol) "encapsulates" x_i by using the i-th public key $(A_{i,0}, A_{i,1})$. We call it i-th encapsulation. Finally, the sender generates $w = m \oplus H(x)$, where H is a hash function explained later in more detail.

The resulting ciphertext is

$$\left(\begin{pmatrix} B_{1,0},\dots,B_{n,0} \\ B_{1,1},\dots,B_{n,1} \end{pmatrix}, \begin{pmatrix} \mathbf{e}_{1,0},\dots,\mathbf{e}_{n,0} \\ \mathbf{e}_{1,1},\dots,\mathbf{e}_{n,1} \end{pmatrix}, w \right).$$

Decryption is done by recovering each x_i in the same way as $\mathtt{NCE}^1_{\mathtt{lin}}$ and computing $w \oplus H(\mathsf{x})$.

The simulator for this scheme runs as follows. It first generates $\mathsf{z}_1 \cdots \mathsf{z}_n = \mathsf{z} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{x}_1 \cdots \mathsf{x}_n = \mathsf{x} \xleftarrow{\$} \{0,1\}^n$. Then, for every index $i \in [n]$ such that $\mathsf{z}_i = \mathsf{x}_i$, it simulates the i-th public key and encapsulation in the same way as the simulator for $\mathtt{NCE}^1_{\mathtt{lin}}$ (and thus Beaver's protocol). For every index $i \in [n]$ such that $\mathsf{z}_i \neq \mathsf{x}_i$, it simply generates i-th public key and encapsulation in the same way as $\mathtt{NCE}_{\mathtt{lin}}$ does in the real execution. Finally, it generates $w \xleftarrow{\$} \{0,1\}^\mu$.

Although the ciphertext generated by the simulator is not "fully non-committing" about x, it loses the information of bits of x such that $\mathsf{x}_i = \mathsf{z}_i$. Thus, if we can program the output value of the hash function H freely by programming only those bits of x, the simulator can later open the ciphertext to any message, and we see that $\mathtt{NCE}_{\mathtt{lin}}$ is a secure NCE scheme.

To realize this idea, we first set $n = 8\mu$ in order to ensure that the simulated ciphertext loses the information of at least μ-bits of x with overwhelming probability. This is guaranteed by the Chernoff bound. Moreover, as the hash function H, we use a matrix $R \in \{0,1\}^{\mu \times n}$, such that randomly picked μ out of n column vectors of length μ are linearly independent. The ciphertext rate of $\mathtt{NCE}_{\mathtt{lin}}$ is $\mathcal{O}(\lambda)$, that is already the same as the best rate based on the DDH problem achieved by the construction of Choi et al. [7].

3.3 Reduce the Ciphertext Rate

Finally, we show how to achieve the ciphertext rate $\mathcal{O}(\log \lambda)$ by compressing the ciphertext of $\mathtt{NCE}_{\mathtt{lin}}$. This is done by two steps. In the first step, we reduce the size of the first part of a ciphertext of $\mathtt{NCE}_{\mathtt{lin}}$, that is, $\{B_{i,b}\}_{i\in[n],b\in\{0,1\}}$. By this step, we compress it into just a single group element. Then, in the second step, we reduce the size of the second part of a ciphertext of $\mathtt{NCE}_{\mathtt{lin}}$, that is, $\{\mathbf{e}_{i,b}\}_{i\in[n],b\in\{0,1\}}$. In this step, we compress each $\mathbf{e}_{i,b}$ into a $\mathcal{O}(\log \lambda)$-bit string. By applying these two steps, we can achieve the ciphertext rate $\mathcal{O}(\log \lambda)$.

The second step is done by replacing each group element $\mathbf{e}_{i,b}$ with a hash value of it. In $\mathtt{NCE}_{\mathtt{lin}}$, they are used to recover the value of x_i by checking $B^{\rho_i}_{i,\mathsf{z}_i} = \mathbf{e}_{i,\mathsf{z}_i}$. We can successfully perform this recovery process with overwhelming probability even if $\mathbf{e}_{i,b}$ is hashed to a $\mathsf{poly}(\log \lambda)$-bit string. Furthermore, with the help of an *error-correcting code*, we can reduce the length of the hash value to $\mathcal{O}(\log \lambda)$-bit.

In the remaining part, we explain how to perform the first step.

Compressing a Matrix of Group Elements into a Single Group Element. We realize that we do not need all of the elements $\{B_{i,b}\}_{i\in[n],b\in\{0,1\}}$ to decrypt the ciphertext. Although the receiver gets both $B_{i,0}$ and $B_{i,1}$ for every $i \in [n]$, the receiver uses only B_{i,z_i}. Recall that the receiver recovers the value of x_i by checking whether $B^{\rho_i}_{i,\mathsf{z}_i} = \mathbf{e}_{i,\mathsf{z}_i}$ holds. This recovery of x_i can be done even if the sender sends only B_{i,x_i}, and not $B_{i,1-\mathsf{x}_i}$.

This is because, similarly to the equation $B^{\rho_i}_{i,\mathsf{z}_i} = \mathbf{e}_{i,\mathsf{z}_i}$, with overwhelming probability, the equation $B^{\rho_i}_{i,\mathsf{x}_i} = \mathbf{e}_{i,\mathsf{z}_i}$ holds if and only if $\mathsf{z}_i = \mathsf{x}_i$. For this reason, we can compress the first part of the ciphertext on $\mathtt{NCE}_{\mathtt{lin}}$ into $(B_{1,\mathsf{x}_1}, \ldots, B_{n,\mathsf{x}_n})$.

We further compress $(B_{1,\mathsf{x}_1}, \ldots, B_{n,\mathsf{x}_n})$ into a single group element generated by multiplying them, that is, $\mathsf{y} = \prod_{j\in[n]} B_{j,\mathsf{x}_j}$. In order to do so, we modify the scheme so that the receiver can recover x_i for every $i \in [n]$ using y instead of B_{i,x_i}. Concretely, for every $i \in [n]$, the sender computes $\mathbf{e}_{i,\mathsf{x}_i}$ as

$$\mathbf{e}_{i,\mathsf{x}_i} = \prod_{j\in[n]} A^{\alpha_j}_{i,\mathsf{x}_i},$$

where α_j is the exponent of B_{j,x_j} for every $j \in [n]$ generated by the sender. The sender still generates $\mathbf{e}_{i,1-\mathsf{x}_i}$ as a random group element for every $i \in [n]$. In this case, with overwhelming probability, the receiver can recover x_i by checking whether $\mathbf{e}_{i,\mathsf{z}_i} = \mathsf{y}^{\rho_i}$ holds.

However, unfortunately, it seems difficult to prove the security of this construction. In order to delete the information of x_i for indices $i \in [n]$ such that $\mathsf{z}_i = \mathsf{x}_i$ as in the proof of $\mathtt{NCE}_{\mathtt{lin}}$, we have to change the distribution of $\mathbf{e}_{i,1-\mathsf{x}_i}$ from a random group element to $\prod_{j\in[n]} A_{i,1-\mathsf{x}_i}^{\alpha_j}$ so that $\mathbf{e}_{i,0}$ and $\mathbf{e}_{i,1}$ are symmetrically generated. However, we cannot make this change by relying on the DDH assumption since all α_j are given to the adversary as a part of the sender random coin. Thus, in order to solve this problem, we further modify the scheme and construct an NCE scheme $\mathtt{NCE}$ as follows.

The Resulting NCE Scheme $\mathtt{NCE}$. In $\mathtt{NCE}$, the receiver first generates $\mathsf{z} \xleftarrow{\$} \{0,1\}^n$ and $\{A_{i,b}\}_{i\in[n],b\in\{0,1\}}$ in the same way as $\mathtt{NCE}_{\mathtt{lin}}$. Moreover, instead of the sender, the receiver *obliviously* generates $B_{i,b} = g^{\alpha_{i,b}}$ for every $i \in [n]$ and $b \in \{0,1\}$, and adds them into the public key. Moreover, for every $i \in [n]$, the receiver adds

$$\{B_{j,b}^{\rho_i} = A_{i,\mathsf{z}_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,1-\mathsf{z}_i)}$$

to the public key. In order to avoid the leakage of the information of z from the public key, for every $i \in [n]$, we have to add

$$\{A_{i,1-\mathsf{z}_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,\mathsf{z}_i)}$$

to the public key. However, the receiver cannot do it since the receiver generates $A_{i,1-\mathsf{z}_i}$ obliviously. Thus, instead, the receiver adds the same number of random group elements into the public key. At the beginning of the security proof, we can replace them with $\{A_{i,1-\mathsf{z}_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,\mathsf{z}_i)}$ by relying on the DDH assumption, and eliminate the information of z from the public key. For simplicity, below, we suppose that the public key includes $\{A_{i,1-\mathsf{z}_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,\mathsf{z}_i)}$ instead of random group elements.

When encrypting a message m by $\mathtt{NCE}$, the sender first generates $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and computes $\mathsf{y} = \prod_{j\in[n]} B_{j,\mathsf{x}_j}$. Then, for every $i \in [n]$, the sender computes $\mathbf{e}_{i,\mathsf{x}_i}$ as

$$\mathbf{e}_{i,\mathsf{x}_i} = \prod_{j\in[n]} A_{i,\mathsf{x}_i}^{\alpha_{j,\mathsf{x}_j}} = \mathsf{y}^{\rho_i}$$

just multiplying $A_{i,\mathsf{x}_i}^{\alpha_{1,\mathsf{x}_1}}, \ldots, A_{i,\mathsf{x}_i}^{\alpha_{n,\mathsf{x}_n}}$ included in the pubic key. Recall that $A_{i,\mathsf{x}_i} = g^{\rho_i}$. Note that $A_{i,\mathsf{z}_i}^{\alpha_{i,1-\mathsf{z}_i}}$ is not included in the public key, but we do not need it to compute $\mathbf{e}_{i,\mathsf{x}_i}$. The sender generates $\mathbf{e}_{i,\mathsf{x}_i}$ as a random group element for every $i \in [n]$ as before. The resulting ciphertext is

$$\left(\mathsf{y}, \begin{pmatrix}\mathbf{e}_{1,0},\ldots,\mathbf{e}_{n,0}\\ \mathbf{e}_{1,1},\ldots,\mathbf{e}_{n,1}\end{pmatrix}, R\mathsf{x} \oplus m\right).$$

The receiver can recover x_i by checking whether $\mathbf{e}_{i,\mathsf{z}_i} = \mathsf{y}^{\rho_i}$ holds, and decrypt the ciphertext.

By defining the simulator appropriately, the security proof of NCE proceeds in a similar way to that of $\mathtt{NCE}_{\mathtt{lin}}$. In NCE, for indices $i \in [n]$ such that $\mathsf{z}_i = \mathsf{x}_i$, we can eliminate the information of x_i. We can change $\mathbf{e}_{i,1-\mathsf{x}_i}$ from a random group element to $\prod_{j\in[n]} A_{i,1-\mathsf{x}_i}^{\alpha_{j,\mathsf{x}_j}}$ by relying on the fact that $A_{i,1-\mathsf{x}_i}^{\alpha_{i,\mathsf{x}_i}}$ is indistinguishable from a random group element by the DDH assumption. By this change, $\mathbf{e}_{i,0}$ and $\mathbf{e}_{i,1}$ become symmetric and the ciphertext loses the information of x_i. Then, the remaining part of the proof goes through in a similar way as that of $\mathtt{NCE}_{\mathtt{lin}}$ except the following point. In NCE, the first component of the ciphertext, that is, $\mathsf{y} = \prod_{j\in[n]} B_{j,\mathsf{x}_j}$ has the information of x. In order to deal with the issue, in our real construction, we replace y with $g^{\mathsf{r}} \prod_{j\in[n]} B_{j,\mathsf{x}_j}$, where $\mathsf{r} \xleftarrow{\$} \mathbb{Z}_p$. Then, y no longer leaks any information of x. Moreover, after y is fixed, for any $\mathsf{x}' \in \{0,1\}^n$, we can efficiently find r' such that $\mathsf{y} = g^{\mathsf{r}'} \prod_{j\in[n]} B_{j,\mathsf{x}'_j}$. This is important to ensure that the simulator of NCE runs in polynomial time.

3.4 Abstraction by Chameleon Encryption

We can describe NCE by using obliviously samplable chameleon encryption. If we consider $\{B_{i,b}\}_{i\in[n],b\in\{0,1\}}$ as a hash key k of chameleon hash function, the first element of the ciphertext $g^{\mathsf{r}} \prod_{j\in[n]} B_{j,\mathsf{x}_j}$ can be seen as the output of the hash $\mathsf{H}(\mathsf{k}, \mathsf{x}; \mathsf{r})$. Moreover, group elements contained in the public key are considered as ciphertexts of an chameleon encryption scheme. Oblivious samplability of chameleon encryption makes it possible to deal with the above stated issue of sampling random group elements instead of $\{A_{i,1-\mathsf{z}_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,\mathsf{z}_i)}$ for every $i \in [n]$.

Relation with Trapdoor Function of Garg and Hajiabadi. We finally remark that the construction of NCE can be seen as an extension of that of trapdoor function (TDF) proposed by Garg and Hajiabadi [14].

If we do not add the random mask g^{r} to $\mathsf{y} = \prod_{j\in[n]} B_{j,\mathsf{x}_j}$, the key encapsulation part of a ciphertext of NCE, that is,

$$\left(\mathsf{y}, \begin{pmatrix} \mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1} \end{pmatrix}\right)$$

is the same as an output of the TDF constructed by Garg and Hajiabadi. The major difference between our NCE scheme and their TDF is the secret key. A secret key of their TDF contains all discrete logs of $\{A_{i,b}\}_{i\in[n],b\in\{0,1\}}$, that is, $\{\rho_{i,b}\}_{i\in[n],b\in\{0,1\}}$. On the other hand, a secret key of our NCE scheme contains half of them corresponding to the bit representation of z, that is, $\{\rho_{i,\mathsf{z}_i}\}_{i\in[n]}$. Garg and Hajiabadi already stated that their TDF can be inverted with $\{\rho_{i,\mathsf{z}_i}\}_{i\in[n]}$ for any $\mathsf{z} \in \{0,1\}^n$, and use this fact in the security proof of a chosen ciphertext security of a public-key encryption scheme based on their TDF. By explicitly using this technique in the construction, we achieve non-committing property.

We observe that construction techniques for TDF seem to be useful for achieving NCE. Encryption schemes that can recover an encryption random coin with a message in the decryption process, such as those based on TDFs, is said to be randomness recoverable. For randomness recoverable schemes, receiver non-committing property is sufficient to achieve full (that is, both sender and receiver) non-committing property. This is because an encryption random coin can be recovered from a ciphertext by using a key generation random coin.

4 Obliviously Samplable Chameleon Encryption

Chameleon encryption was originally introduced by Döttling and Garg [10]. In this work, we introduce a variant of chameleon encryption satisfying *oblivious samplability*.

4.1 Definiton

We start with the definition of the chameleon hash function.

Definition 4 (Chameleon Hash Function). *A chameleon hash function consists of the following PPT algorithms* $\left(\mathsf{K}, \mathsf{H}, \mathsf{H}^{-1}\right)$. *Below, we let the input space and randomness space of* H *be* $\{0,1\}^n$ *and* $\mathcal{R}_{\mathsf{H}}$, *respectively, where* $n = O(\lambda)$.

- $\mathsf{K}\left(1^\lambda\right)$: *Given the security parameter* 1^λ, *it outputs a hash key* k *and a trapdoor* t.
- $\mathsf{H}\left(\mathsf{k}, \mathsf{x}; \mathsf{r}\right)$: *Given a hash key* k *and input* $\mathsf{x} \in \{0,1\}^n$, *using randomness* $\mathsf{r} \in \mathcal{R}_{\mathsf{H}}$, *it outputs a hash value* y.
- $\mathsf{H}^{-1}\left(\mathsf{t}, (\mathsf{x}, \mathsf{r}), \mathsf{x}'\right)$: *Given a trapdoor* t, *an input to the hash* x, *randomness for the hash* r *and another input to the hash* x', *it outputs randomness* r'.

A chameleon hash function is required to satisfy the following trapdoor collision property.[2]

Trapdoor Collision. *For all* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$ *and hash randomness* $\mathsf{r} \in \mathcal{R}_{\mathsf{H}}$, $\mathsf{H}\left(\mathsf{k}, \mathsf{x}; \mathsf{r}\right) = \mathsf{H}(\mathsf{k}, \mathsf{x}'; \mathsf{r}')$ *holds, where* $(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}\left(1^\lambda\right), \mathsf{r}' \leftarrow \mathsf{H}^{-1}\left(\mathsf{t}, (\mathsf{x}, \mathsf{r}), \mathsf{x}'\right)$. *Moreover, if* r *is sampled uniformly at random, then so is* r'.

Next, we define the chameleon encryption.

Definition 5 (Chameleon Encryption). *Chameleon encryption (CE) consists of a chameleon hash function* $\left(\mathsf{K}, \mathsf{H}, \mathsf{H}^{-1}\right)$ *and the following PPT algorithms* $(\mathsf{E}_1, \mathsf{E}_2, \mathsf{D})$. *Below, we let the input space and randomness space of* H *are* $\{0,1\}^n$ *and* $\mathcal{R}_{\mathsf{H}}$, *respectively, where* $n = O(\lambda)$. *We also let the randomness space of* E_1 *and* E_2 *be* $\mathcal{R}_{\mathsf{E}}$. *Moreover, we let the output space of* E_2 *be* $\{0,1\}^\ell$, *where* ℓ *be a polynomial of* λ.

[2] Usually, a chameleon hash function is required to be collision resistant, but we omit it since it is implied by the security of chameleon encryption defined later.

- $\mathsf{E}_1(\mathsf{k}, (i, b); \rho)$*: Given a hash key* k *and index* $i \in [n]$ *and* $b \in \{0,1\}$*, using a random coin* $\rho \in \mathcal{R}_{\mathsf{E}}$*, it outputs a ciphertext* ct.
- $\mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho)$*: Given a hash key* k *and a hash value* y*, using a random coin* $\rho \in \mathcal{R}_{\mathsf{E}}$*, it outputs* $\mathbf{e} \in \{0,1\}^{\ell}$.
- $\mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct})$*: Given a hash key* k*, a pre-image of the hash* (x, r) *and a ciphertext* ct*, it outputs* $\mathbf{e} \in \{0,1\}^{\ell}$ *or* $\perp$.

Chameleon encryption must satisfy the following correctness and security.

Correctness. *For all* k *output by* $\mathsf{K}(1^{\lambda}), i \in [n], \mathsf{x} \in \{0,1\}^n, \mathsf{r} \in \mathcal{R}_{\mathsf{H}}$*, and* $\rho \in \mathcal{R}_{\mathsf{E}}, \mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho) = \mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct})$ *holds, where* $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}, \mathsf{x}; \mathsf{r})$ *and* $\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{k}, (i, \mathsf{x}_i); \rho)$.

Security. *For any stateful PPT adversary* $\mathcal{A}$*, we define the following experiments.*

$\mathsf{Exp}^0_{\mathtt{CE},\mathcal{A}}$	$\mathsf{Exp}^1_{\mathtt{CE},\mathcal{A}}$
$(\mathsf{x}, \mathsf{r}, i) \leftarrow \mathcal{A}(1^{\lambda})$	$(\mathsf{x}, \mathsf{r}, i) \leftarrow \mathcal{A}(1^{\lambda})$
$(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}(1^{\lambda})$	$(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}(1^{\lambda})$
$\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{k}, (i, 1-\mathsf{x}_i); \rho)$	$\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{k}, (i, 1-\mathsf{x}_i); \rho)$
$\mathbf{e} \leftarrow \mathsf{E}_2(\mathsf{k}, \mathsf{H}(\mathsf{k}, \mathsf{x}; \mathsf{r}); \rho)$	$\mathbf{e} \xleftarrow{\$} \{0,1\}^{\ell}$
$\mathsf{out} \leftarrow \mathcal{A}(\mathsf{k}, \mathsf{ct}, \mathbf{e})$	$\mathsf{out} \leftarrow \mathcal{A}(\mathsf{k}, \mathsf{ct}, \mathbf{e})$

We say $\mathtt{CE}$ *is secure if*

$$\mathsf{Adv}_{\mathtt{CE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^0_{\mathtt{CE},\mathcal{A}}\right] - \Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^1_{\mathtt{CE},\mathcal{A}}\right]\right| = \mathsf{negl}(\lambda)$$

holds for every PPT adversary $\mathcal{A}$.

Remark 1 (On the recyclability). The above definition of chameleon encryption is slightly different from that of Döttling and Garg [10] since we define it so that it satisfies a property called recyclability introduced by Garg and Hajiabadi [14] when defining a primitive called one-way function with encryption that is similar to chameleon encryption.

More specifically, in our definition, there are two encryption algorithms E_1 and E_2. E_1 outputs only a key encapsulation part and E_2 outputs only a session key part. In the original definition by Döttling and Garg, there is a single encryption algorithm that outputs the key encapsulation part and a message masked by the session key part at once. Importantly, an output of E_1 does not depend on a hash value y. This makes possible to relate a single output of E_1 with multiple hash values. (In other words, a single output of E_1 can be recycled for multiple hash values.) We need this property in the construction of NCE and thus adopt the above definition.

We then introduce our main tool, that is, obliviously samplable chameleon encryption (obliviously samplable CE).

Definition 6 (Obliviously Samplable Chameleon Encryption). *Let* $\mathtt{CE} = (\mathsf{K}, \mathsf{H}, \mathsf{H}^{-1}, \mathsf{E}_1, \mathsf{E}_2, \mathsf{D})$ *be a chameleon encryption scheme. We define two associated PPT algorithms* $\widehat{\mathsf{K}}$ *and* $\widehat{\mathsf{E}_1}$ *as follows.*

- $\widehat{\mathsf{K}}\left(1^\lambda\right)$*: Given the security parameter* 1^λ*, it outputs only a hash key* $\widehat{\mathsf{k}}$ *without using any randomness other than* $\widehat{\mathsf{k}}$ *itself.*
- $\widehat{\mathsf{E}_1}\left(\widehat{\mathsf{k}}, (i,b)\right)$*: Given a hash key* $\widehat{\mathsf{k}}$ *and index* $i \in [n]$ *and* $b \in \{0,1\}$*, it outputs a ciphertext* $\widehat{\mathsf{ct}}$ *without using any randomness except* $\widehat{\mathsf{ct}}$ *itself.*

For any PPT adversary $\mathcal{A}$*, we also define the following experiments.*

$\mathsf{Exp}^{os\text{-}0}_{\mathtt{CE},\mathcal{A}}$	$\mathsf{Exp}^{os\text{-}1}_{\mathtt{CE},\mathcal{A}}$
$(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}\left(1^\lambda\right)$	$\widehat{\mathsf{k}} \leftarrow \widehat{\mathsf{K}}\left(1^\lambda\right)$
$\mathsf{out} \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(\mathsf{k})$	$\mathsf{out} \leftarrow \mathcal{A}^{\widehat{O}(\cdot,\cdot)}\left(\widehat{\mathsf{k}}\right)$

The oracles $O(\cdot,\cdot)$ *and* $\widehat{O}(\cdot,\cdot)$ *are defined as follows.*

- $O(i,b)$*: Given an index* $i \in [n]$ *and* $b \in \{0,1\}$*, it returns* $\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{k}, (i,b); \rho)$ *using uniformly random* ρ*.*
- $\widehat{O}(i,b)$*: Given an index* $i \in [n]$ *and* $b \in \{0,1\}$*, it returns* $\widehat{\mathsf{ct}} \leftarrow \widehat{\mathsf{E}_1}\left(\widehat{\mathsf{k}}, (i,b)\right)$*.*

We say that $\mathtt{CE}$ *is obliviously samplable if*

$$\mathsf{Adv}^{os}_{\mathtt{CE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^{os\text{-}0}_{\mathtt{CE},\mathcal{A}}\right] - \Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}^{os\text{-}1}_{\mathtt{CE},\mathcal{A}}\right]\right| = \mathsf{negl}(\lambda)$$

holds for every PPT adversary $\mathcal{A}$*.*

We define another correctness of obliviously samplable CE necessary to assure the correctness of our NCE.

Definition 7 (Correctness under Obliviously Sampled Keys). *An obliviously samplable CE* $(\mathtt{CE}, \widehat{\mathsf{K}}, \widehat{\mathsf{E}_1})$ *is correct under obliviously sampled keys if for all* $\widehat{\mathsf{k}}$ *output by* $\widehat{\mathsf{K}}, i \in [n], \mathsf{x} \in \{0,1\}^n, \mathsf{r} \in \mathcal{R}_{\mathsf{H}}$*, and* $\rho \in \mathcal{R}_{\mathsf{E}}, \mathsf{E}_2\left(\widehat{\mathsf{k}}, (i,b); \rho\right) = \mathsf{D}\left(\widehat{\mathsf{k}}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}\right)$ *holds, where* $\mathsf{y} \leftarrow \mathsf{H}\left(\widehat{\mathsf{k}}, \mathsf{x}; \mathsf{r}\right)$ *and* $\mathsf{ct} \leftarrow \mathsf{E}_1\left(\widehat{\mathsf{k}}, (i, \mathsf{x}_i); \rho\right)$*.*

4.2 Construction

We construct an obliviously samplable CE $\mathtt{CE} = \left(\mathsf{K}, \mathsf{H}, \mathsf{H}^{-1}, \mathsf{E}_1, \mathsf{E}_2, \mathsf{D}, \widehat{\mathsf{K}}, \widehat{\mathsf{E}_1}\right)$ based on the hardness of the DDH problem.

Let $\mathbb{G}$ be a cyclic group of order p with a generator g. In the construction, we use a universal hash family $\mathcal{H} = \{\mathsf{H}_{\mathbb{G}} : \mathbb{G} \to \{0,1\}^\ell\}$. Below, let $\mathsf{H}_{\mathbb{G}}$ be a hash function sampled from $\mathcal{H}$ uniformly at random, and it is given to all the algorithms implicitly.

$\mathsf{K}\left(1^{\lambda}\right)$:
- For all $i \in [n], b \in \{0,1\}$, sample $\alpha_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and set $g_{i,b} := g^{\alpha_{i,b}}$.
- Output

$$\mathsf{k} := \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right) \text{ and } \mathsf{t} := \begin{pmatrix} \alpha_{1,0}, \ldots, \alpha_{n,0} \\ \alpha_{1,1}, \ldots, \alpha_{n,1} \end{pmatrix}. \tag{1}$$

$\mathsf{H}\left(\mathsf{k}, \mathsf{x}; \mathsf{r}\right)$:
- Sample $\mathsf{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{H}} = \mathbb{Z}_p$ and output $\mathsf{y} = g^{\mathsf{r}} \prod_{i \in [n]} g_{i,\mathsf{x}_i}$.

$\mathsf{H}^{-1}\left(\mathsf{t}, (\mathsf{x}, \mathsf{r}), \mathsf{x}'\right)$:
- Parse t as in Eq. 1.
- Output $\mathsf{r}' := \mathsf{r} + \sum_{i \in [n]} \left(\alpha_{i,\mathsf{x}_i} - \alpha_{i,\mathsf{x}'_i}\right)$.

$\mathsf{E}_1\left(\mathsf{k}, (i, b); \rho\right)$:
- Parse k as in Eq. 1.
- Sample $\rho \xleftarrow{\$} \mathcal{R}_{\mathsf{E}} = \mathbb{Z}_p$ and compute $c := g^{\rho}$.
- Compute $c_{i,b} := (g_{i,b})^{\rho}$ and $c_{i,1-b} := \perp$.
- For all $j \in [n]$ such that $j \neq i$, compute $c_{j,0} := (g_{j,0})^{\rho}$ and $c_{j,1} := (g_{j,1})^{\rho}$
- Output

$$\mathsf{ct} := \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right). \tag{2}$$

$\mathsf{E}_2\left(\mathsf{k}, \mathsf{y}; \rho\right)$:
- Output $\mathbf{e} \leftarrow \mathsf{H}_{\mathbb{G}}\left(\mathsf{y}^{\rho}\right)$.

$\mathsf{D}\left(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}\right)$:
- Parse ct as in Eq. 2.
- Output $\mathbf{e} \leftarrow \mathsf{H}_{\mathbb{G}}\left(c^{\mathsf{r}} \prod_{i \in [n]} c_{i,\mathsf{x}_i}\right)$.

$\widehat{\mathsf{K}}\left(1^{\lambda}\right)$:
- For all $i \in [n]$ and $b \in \{0,1\}$, sample $g_{i,b} \xleftarrow{\$} \mathbb{G}$.
- Output $\widehat{\mathsf{k}} := \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right)$.

$\widehat{\mathsf{E}_1}\left(\widehat{\mathsf{k}}, (i, b)\right)$:
- Set $\widehat{c}_{i,1-b} := \perp$, and sample $\widehat{c} \xleftarrow{\$} \mathbb{G}$ and $\widehat{c}_{i,b} \xleftarrow{\$} \mathbb{G}$.
- For all $j \in [n]$ such that $j \neq i$, sample $\widehat{c}_{j,0} \xleftarrow{\$} \mathbb{G}$ and $\widehat{c}_{j,1} \xleftarrow{\$} \mathbb{G}$.
- Output $\widehat{\mathsf{ct}} := \left(\widehat{c}, \begin{pmatrix} \widehat{c}_{1,0}, \ldots, \widehat{c}_{n,0} \\ \widehat{c}_{1,1}, \ldots, \widehat{c}_{n,1} \end{pmatrix}\right)$.

Theorem 1. CE *is an obliviously samplable CE scheme assuming the hardness of the DDH problem.*

The trapdoor collision property, correctness, and correctness under obliviously sampled keys of CE directly follow from the construction of CE. Below, we first prove the security of CE under the hashed DDH assumption with respect to $\mathcal{H}$. We then prove the oblivious samplability of CE under the DDH assumption.

Security. Let $\mathcal{A}$ be an adversary against the security of $\mathtt{CE}$. We construct a reduction algorithm $\mathcal{A}'$ which solves the hashed DDH problem using $\mathcal{A}$.

Given $(\mathsf{H}_{\mathbb{G}}, g_1, g_2, \mathbf{e})$, $\mathcal{A}'$ first runs $(\mathsf{x}, \mathsf{r}, i) \leftarrow \mathcal{A}(1^\lambda)$, and generates k as follows. For all $(j, b) \in [n] \times \{0,1\}$ such that $(j, b) \neq (i, \mathsf{x}_i)$, $\mathcal{A}'$ samples $\alpha_{j,b} \xleftarrow{\$} \mathbb{Z}_p$ and sets $g_{j,b} := g^{\alpha_{j,b}}$, $g_{i,\mathsf{x}_i} := g_1 / \left(g^{\mathsf{r}} \prod_{j \neq i} g_{j,\mathsf{x}_j}\right)$ and

$$\mathsf{k} := \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right).$$

Next, $\mathcal{A}'$ generates ct as follows. $\mathcal{A}'$ first sets $c := g_2$ and $c_{i,\mathsf{x}_i} := \perp$. Then for all $(j, b) \in [n] \times \{0,1\}$ such that $(j, b) \neq (i, \mathsf{x}_i)$, $\mathcal{A}'$ sets $c_{j,b} := {g_2}^{\alpha_{j,b}}$. $\mathcal{A}'$ sets the ciphertext to

$$\mathsf{ct} := \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right).$$

Finally, $\mathcal{A}'$ outputs what $\mathcal{A}(\mathsf{k}, \mathsf{ct}, \mathbf{e})$ does.

k and ct generated by $\mathcal{A}'$ distribute identically to those output by $\mathsf{K}\left(1^\lambda\right)$ and $\mathsf{E}_1(\mathsf{k}, (i, 1 - \mathsf{x}_i); \rho)$, respectively. $\mathcal{A}'$ perfectly simulates $\mathsf{Exp}^0_{\mathtt{CE},\mathcal{A}}$ to $\mathcal{A}$ if $\mathbf{e} = \mathsf{H}_{\mathbb{G}}(\mathsf{dh}(g_1, g_2))$ because we have

$$\mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho) = \mathsf{H}_{\mathbb{G}}\left(\mathsf{dh}\left(g^{\mathsf{r}} \prod_{i \in [n]} g_{i,\mathsf{x}_i}, c\right)\right) = \mathsf{H}_{\mathbb{G}}(\mathsf{dh}(g_1, g_2)) = \mathbf{e}.$$

On the other hand, if $\mathbf{e} \xleftarrow{\$} \{0,1\}^\ell$, $\mathcal{A}'$ perfectly simulates $\mathsf{Exp}^1_{\mathtt{CE},\mathcal{A}}$ to the adversary. Thus, it holds that $\mathsf{Adv}_{\mathtt{CE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ under the hash DDH assumption with respect to $\mathcal{H}$.

This completes the security proof of $\mathtt{CE}$.

Oblivious Samplability. Let $\mathcal{A}$ be an PPT adversary that attacks oblivious samplability of $\mathtt{CE}$ and makes q queries to its oracle. We prove that the probability that $\mathcal{A}$ outputs 1 in $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$ is negligibly close to that in $\mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$. The detailed description of these experiments is as follows.

$\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$**:** $\mathcal{A}$ is given a hash key k output by K and can access to the oracle $O(i, b) = \mathsf{E}_1(\mathsf{k}, (i, b); \rho)$, where $i \in [n]$, $b \in \{0,1\}$, and $\rho \leftarrow \mathbb{Z}_p$. Concretely, $O(i, b)$ behaves as follows.

- Sample ρ uniformly from $\mathbb{Z}_p$, and let $c := g^\rho$. For all $j \neq i$, let $c_{j,0} := (g_{j,0})^\rho$ and $c_{j,1} := (g_{j,1})^\rho$, and let $c_{i,b} := (g_{i,b})^\rho$ and $c_{i,1-b} := \perp$. Return $\mathsf{ct} := \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.

$\mathsf{Exp}^{\text{os-1}}_{\mathtt{C}E,\mathcal{A}}$: $\mathcal{A}$ is given a hash key $\widehat{\mathsf{k}}$ output by $\widehat{\mathsf{K}}$ and can access to the oracle $\widehat{O}(i, b) = \widehat{\mathsf{E}_1}\left(\widehat{\mathsf{k}}, (i, b)\right)$, where $i \in [n]$ and $b \in \{0,1\}$. Concretely, $\widehat{O}(i, b)$ behaves as follows.

– Let $\widehat{c}_{i,1-b} := \bot$, and sample $\widehat{c}, \widehat{c}_{i,b}$, and $\widehat{c}_{j,0}$ and $\widehat{c}_{j,1}$ for all $j \neq i$ uniformly from $\mathbb{G}$. Return $\widehat{\mathsf{ct}} := \left(\widehat{c}, \begin{pmatrix} \widehat{c}_{1,0}, \ldots, \widehat{c}_{n,0} \\ \widehat{c}_{1,1}, \ldots, \widehat{c}_{n,1} \end{pmatrix}\right)$.

We define $\mathsf{Exp}\ j$ for every $j \in \{0, \ldots, q\}$ that are intermediate experiments between $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$ and $\mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$ as follows. Below, for two experiments $\mathsf{Exp}\ X$ and $\mathsf{Exp}\ Y$, we write $\mathsf{Exp}\ X \approx \mathsf{Exp}\ Y$ to denote that the probability that $\mathcal{A}$ outputs 1 in $\mathsf{Exp}\ X$ is negligibly close to that in $\mathsf{Exp}\ Y$.

$\mathsf{Exp}\ j$**:** This experiment is exactly the same as $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$ except how queries made by $\mathcal{A}$ are answered. For the j'-th query $(i, b) \in [n] \times \{0, 1\}$ made by $\mathcal{A}$, the experiment returns $\mathsf{E}_1(\mathsf{k}, (i, b); \rho)$ if $j < j'$, and $\widehat{\mathsf{E}_1}(\mathsf{k}, (i, b))$ otherwise.

We see that $\mathsf{Exp}\ 0$ and $\mathsf{Exp}\ q$ are exactly the same experiment as $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$ and $\mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$, respectively. Note that $\mathcal{A}$ is given k output by $\mathsf{K}\left(1^\lambda\right)$ and can access to the oracle $\widehat{\mathsf{E}_1}(\mathsf{k}, (i, b))$ in $\mathsf{Exp}\ q$, but on the other hand, $\mathcal{A}$ is given $\widehat{\mathsf{k}}$ output by $\widehat{\mathsf{K}}\left(1^\lambda\right)$ and can access to the oracle $\widehat{\mathsf{E}_1}\left(\widehat{\mathsf{k}}, (i, b)\right)$ in $\mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$. However, this is not a problem since k output by $\mathsf{K}\left(1^\lambda\right)$ and $\widehat{\mathsf{k}}$ output by $\widehat{\mathsf{K}}\left(1^\lambda\right)$ distribute identically in our construction. For every $j \in [q]$, $\mathsf{Exp}\ j - 1 \approx \mathsf{Exp}\ j$ directly follows from Lemma 3. Therefore, we have $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}} \approx \mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$ under the DDH assumption. From the above arguments, $\mathtt{CE}$ satisfies oblivious samplability under the DDH assumption.

This completes the proof of Theorem 1.

5 Basic Construction of Proposed NCE

In this section, we present our NCE scheme with ciphertext rate $\mathsf{poly}(\log \lambda)$ from an obliviously samplable CE. We call this construction basic construction. In Sect. 6, improving the basic construction, we describe our full construction of NCE which achieves ciphertext rate $\mathcal{O}(\log \lambda)$.

5.1 Construction

We use three parameters μ, n, and ℓ, all of which are polynomials of λ and concretely determined later.

Let $\mathtt{CE} = \left(\mathsf{K}, \mathsf{H}, \mathsf{H}^{-1}, \mathsf{E}_1, \mathsf{E}_2, \mathsf{D}, \widehat{\mathsf{K}}, \widehat{\mathsf{E}_1}\right)$ be an obliviously samplable CE scheme. We let the input length of H be n and let the output length of E_2 (and thus D) be ℓ. We also let the randomness spaces of H and E_1 be $\mathcal{R}_\mathsf{H}$ and $\mathcal{R}_\mathsf{E}$, respectively. Below, using $\mathtt{CE}$, we construct an NCE scheme $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ whose message space is $\{0, 1\}^\mu$.

In the construction, we use a matrix $R \in \{0, 1\}^{\mu \times n}$, such that randomly picked μ out of n column vectors of length μ are linearly independent. A random matrix satisfies such property except for negligible probability [21].

We first describe $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and show the correctness of $\mathtt{NCE}$ below. We also describe a protocol when using $\mathtt{NCE}$ in Fig. 3.

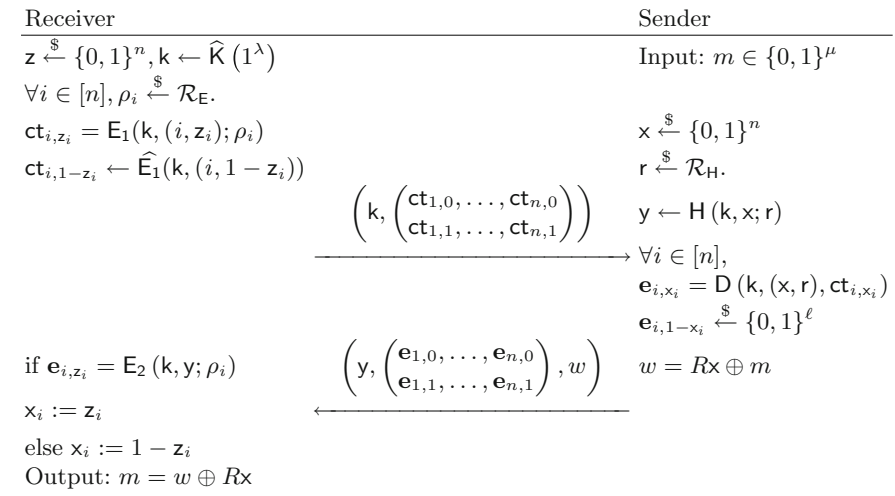

Fig. 3. The description of NCE.

$\mathsf{Gen}\left(1^\lambda; r^{\mathsf{Gen}}\right)$**:**

- Sample $\mathsf{k} \leftarrow \widehat{\mathsf{K}}\left(1^\lambda\right)$ and $\mathsf{z} \xleftarrow{\$} \{0,1\}^n$.
- For all $i \in [n]$, sample $\rho_i \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute

$$\mathsf{ct}_{i,b} \leftarrow \begin{cases} \mathsf{E}_1(\mathsf{k}, (i,b); \rho_i) & (b = \mathsf{z}_i) \\ \widehat{\mathsf{E}_1}(\mathsf{k}, (i,b)) & (b \neq \mathsf{z}_i) \end{cases}.$$

- Output

$$pk := \left(\mathsf{k}, \begin{pmatrix}\mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1}\end{pmatrix}\right) \quad \text{and} \quad sk := (\mathsf{z}, (\rho_1, \ldots, \rho_n)). \tag{3}$$

The random coin r^{Gen} used in Gen is $\left(\mathsf{k}, \mathsf{z}, \{\rho_i\}_{i\in[n]}, \{\mathsf{ct}_{i,1-\mathsf{z}_i}\}_{i\in[n]}\right)$.

$\mathsf{Enc}\left(pk, m; r^{\mathsf{Enc}}\right)$**:**

- Sample $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{H}}$.
- Compute $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}, \mathsf{x}; \mathsf{r})$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute

$$\mathbf{e}_{i,b} \leftarrow \begin{cases} \mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}_{i,b}) & (b = \mathsf{x}_i) \\ \{0,1\}^\ell & (b \neq \mathsf{x}_i) \end{cases}.$$

- Compute $w \leftarrow R\mathsf{x} \oplus m$.

– Output

$$CT := \left(\mathsf{y}, \begin{pmatrix} \mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1} \end{pmatrix}, w\right). \tag{4}$$

The random coin r^{Enc} used in Enc is $\left(\mathsf{x}, \mathsf{r}, \{\mathbf{e}_{i,1-\mathsf{x}_i}\}_{i\in[n]}\right)$.

$\mathsf{Dec}\,(sk, CT)$**:**

– Parse sk and CT as the Eqs. 3 and 4, respectively.

– For all $i \in [n]$, set

$$\mathsf{x}_i := \begin{cases} \mathsf{z}_i & (\mathbf{e}_{i,\mathsf{z}_i} = \mathsf{E}_2\,(\mathsf{k}, \mathsf{y}; \rho_i)) \\ 1 - \mathsf{z}_i & (\text{otherwise}) \end{cases}.$$

– Output $m := R\mathsf{x} \oplus w$.

By setting $\ell = \mathsf{poly}(\log \lambda)$, NCE is correct. Formally, we have the following theorem.

Theorem 2. *Let $\ell = \mathsf{poly}(\log \lambda)$. If* CE *is correct under obliviously sampled keys, then* NCE *is correct.*

Proof. Due to the correctness under obliviously sampled keys of CE, the recovery of x_i fails only when $\mathsf{z}_i \neq \mathsf{x}_i$ happens and $\mathbf{e}_{i,1-\mathsf{x}_i} \xleftarrow{\$} \{0,1\}^\ell$ coincides with $\mathsf{E}_2\,(\mathsf{k}, \mathsf{y}; \rho_i)$. Thus, the probability of decryption failure is bounded by

$$\begin{aligned} &\Pr\left[m \neq \mathsf{Dec}\,(sk, CT)\right] \\ \leq &\Pr\left[\exists i \in [n], \mathbf{e}_{i,1-\mathsf{x}_i} \xleftarrow{\$} \{0,1\}^\ell; \mathbf{e}_{i,1-\mathsf{x}_i} = \mathsf{E}_2\,(\mathsf{k}, \mathsf{y}; \rho_i)\right] \leq \frac{n}{2^\ell}. \end{aligned}$$

Note that at the last step, we used the union bound. Since $n = \mathcal{O}(\lambda)$, the probability is negligible by setting $\ell = \mathsf{poly}(\log \lambda)$. Therefore NCE is correct.

Intuition for the Simulators and Security Proof. The description of the simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ of NCE is somewhat complex. Thus, we give an overview of the security proof for NCE before describing them. We think this will help readers understand the construction of simulators.

In the proof, we start from the real experiment $\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{NCE},\mathcal{A}}$, where $\mathcal{A}$ is an PPT adversary attacking the security of NCE. We then change the experiment step by step so that, in the final experiment, we can generate the ciphertext CT given to $\mathcal{A}$ without the message m chosen by $\mathcal{A}$, which can later be opened to any message. The simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ are defined so that they simulate the final experiment.

In $\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{NCE},\mathcal{A}}$, CT is of the form

$$CT := \left(\mathsf{y}, \begin{pmatrix} \mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1} \end{pmatrix}, R\mathsf{x} \oplus m\right).$$

Informally, $\left(\mathsf{y}, \begin{pmatrix}\mathbf{e}_{1,0},\dots,\mathbf{e}_{n,0}\\ \mathbf{e}_{1,1},\dots,\mathbf{e}_{n,1}\end{pmatrix}\right)$ encapsulates $\mathsf{x} \in \{0,1\}^n$, and $R\mathsf{x} \oplus m$ is a one-time encryption of $m \in \{0,1\}^\mu$ by x. If we can eliminate the information of x from the encapsulation part, CT becomes statistically independent of m. Thus, if we can do that, the security proof is almost complete since in that case, CT can be simulated without m and later be opened to any message. While we cannot eliminate the entire information of x from the encapsulation part, we can eliminate the information of μ out of n bits of x from the encapsulation part, and it is enough to make CT statistically independent of m. Below, we briefly explain how to do it.

We first change $\begin{pmatrix}\mathsf{ct}_{1,0},\dots,\mathsf{ct}_{n,0}\\ \mathsf{ct}_{1,1},\dots,\mathsf{ct}_{n,1}\end{pmatrix}$ contained in pk so that every $\mathsf{ct}_{i,b}$ is generated as $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1(\mathsf{k}, (i,b); \rho_{i,b})$, and set $\rho_i := \rho_{i,\mathsf{z}_i}$, where $\mathsf{z} \in \{0,1\}^n$ is a random string generated in Gen. We can make this change by the oblivious samplability of $\mathtt{CE}$.

Next, by using the security of $\mathtt{CE}$, we try to change the experiment so that for every $i \in [n]$, $\mathbf{e}_{i,0}$ and $\mathbf{e}_{i,1}$ contained in CT are symmetrically generated in order to eliminate the information of x_i from the encapsulation part. Concretely, for every $i \in [n]$, we try to change $\mathbf{e}_{i,1-\mathsf{x}_i}$ from a random string to

$$\mathbf{e}_{i,b} \leftarrow \mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}_{i,1-\mathsf{x}_i}) = \mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho_{i,1-\mathsf{x}_i}).$$

Unfortunately, we cannot change the distribution of every $\mathbf{e}_{i,1-\mathsf{x}_i}$ because some of $\rho_{i,1-\mathsf{x}_i}$ is given to $\mathcal{A}$ as a part of r^{Gen}. Concretely, for $i \in [n]$ such that $\mathsf{z}_i \neq \mathsf{x}_i$, $\rho_i = \rho_{i,\mathsf{z}_i} = \rho_{i,1-\mathsf{x}_i}$ is given to $\mathcal{A}$ and we cannot change the distribution of $\mathbf{e}_{i,1-\mathsf{x}_i}$. On the other hand, for $i \in [n]$ such that $\mathsf{z}_i = \mathsf{x}_i$, we can change the distribution of $\mathbf{e}_{i,1-\mathsf{x}_i}$.

In order to make clear which index $i \in [n]$ we can change the distribution of $\mathbf{e}_{i,1-\mathsf{x}_i}$, in the proof, we replace z with $\mathsf{z}' = \mathsf{x} \oplus \mathsf{z}$. Then, we can say that for $i \in [n]$ such that $\mathsf{z}_i = 0$, we can change the distribution of $\mathbf{e}_{i,1-\mathsf{x}_i}$. Since z is chosen uniformly at random, due to the Chernoff bound, we can ensure that the number of such indices is greater than μ with overwhelming probability by setting n and μ appropriately. Namely, we can eliminate the information of μ out of n bits of x from CT. At this point, CT becomes statistically independent of m, and we almost complete the security proof. Note that y itself does not have any information of x. To make this fact clear, in the proof, we add another step using the trapdoor collision property of $\mathtt{CE}$ after using the security of $\mathtt{CE}$.

To complete the proof formally, we have to ensure that CT can later be opened to any message *efficiently (i.e., in polynomial time)*. This is possible by using a matrix $R \in \{0,1\}^{\mu \times n}$, such that randomly picked μ out of n column vectors of length μ are linearly independent. For more details, see the formal security proof in Sect. 5.2.

We now show the simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$.

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

– Sample $(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}(1^\lambda)$.

- For all $i \in [n]$ and $b \in \{0,1\}$, sample $\rho_{i,b} \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$ and compute $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1(\mathsf{k}, (i,b); \rho_{i,b})$.
- Sample $\mathsf{z} \xleftarrow{\$} \{0,1\}^n, \mathsf{x} \xleftarrow{\$} \{0,1\}^n$ [3], and $\mathsf{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{H}}$.
- Compute $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}, 0^n; \mathsf{r})$ and sample $w \xleftarrow{\$} \{0,1\}^{\mu}$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute

$$\mathbf{e}_{i,b} \leftarrow \begin{cases} \mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho_{i,b}) & (b = \mathsf{x}_i \vee \mathsf{z}_i = 0) \\ \{0,1\}^{\ell} & (b \neq \mathsf{x}_i \wedge \mathsf{z}_i = 1) \end{cases}.$$

- Output

$$pk := \left(\mathsf{k}, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right), \quad CT := \left(\mathsf{y}, \begin{pmatrix} \mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1} \end{pmatrix}, w\right),$$

and $st := (\mathsf{t}, \mathsf{z}, \mathsf{x}, \mathsf{r})$.

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

- Sample x' at random from $\{0,1\}^n$ under the condition that $R\mathsf{x}' = m \oplus w$ and $\mathsf{x}_i = \mathsf{x}'_i$ hold for every $i \in [n]$ such that $\mathsf{z}_i = 1$.
- Compute $\mathsf{r}' \leftarrow \mathsf{H}^{-1}(\mathsf{t}, (0^n, \mathsf{r}), \mathsf{x}')$ and $\mathsf{z}' := \mathsf{z} \oplus \mathsf{x}'$.
- Output

$$r^{\mathsf{Gen}} := \left(\mathsf{k}, \mathsf{z}', \{\rho_{i,\mathsf{z}'_i}\}_{i\in[n]}, \{\mathsf{ct}_{i,1-\mathsf{z}'_i}\}_{i\in[n]}\right) \quad \text{and} \quad r^{\mathsf{Enc}} := \left(\mathsf{x}', \mathsf{r}', \{\mathbf{e}_{i,1-\mathsf{x}'_i}\}_{i\in[n]}\right).$$

5.2 Security Proof

In this section, we prove the security of NCE. Formally, we prove the following theorem.

Theorem 3. *Let $\mu = \mathcal{O}(\lambda)$ and $n = 8\mu$. If* CE *is an obliviously samplable CE, then* NCE *is secure.*

Proof. Let $\mathcal{A}$ is a PPT adversary attacking the security of NCE. We define a sequence of experiments $\mathsf{Exp}\ 0, ..., \mathsf{Exp}\ 6$. Below, for two experiments $\mathsf{Exp}\ X$ and $\mathsf{Exp}\ Y$, we write $\mathsf{Exp}\ X \approx \mathsf{Exp}\ Y$ (resp. $\mathsf{Exp}\ X \equiv \mathsf{Exp}\ Y$) to denote that the probability that $\mathcal{A}$ outputs 1 in $\mathsf{Exp}\ X$ is negligibly close to (resp. the same as) that in $\mathsf{Exp}\ Y$.

$\mathsf{Exp}\ 0$**:** This experiment is exactly the same as $\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{NCE},\mathcal{A}}$. The detailed description is as follows.

1. The experiment first samples $\mathsf{k} \leftarrow \widehat{\mathsf{K}}(1^{\lambda})$ and $\mathsf{z} \xleftarrow{\$} \{0,1\}^n$. Then, for all $i \in [n]$, it samples $\rho_i \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$. Next, for all $i \in [n]$ and $b \in \{0,1\}$, it computes

$$\mathsf{ct}_{i,b} \leftarrow \begin{cases} \mathsf{E}_1(\mathsf{k}, (i,b); \rho_i) & (b = \mathsf{z}_i) \\ \widehat{\mathsf{E}_1}(\mathsf{k}, (i,b)) & (b \neq \mathsf{z}_i) \end{cases}.$$

[3] Sim_1 and Sim_2 do not use x_i for i such that $\mathsf{z}_i = 0$, but for simplicity, we generate whole x.

It sets

$$pk := \left(\mathsf{k}, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right) \quad \text{and} \quad r^{\mathsf{Gen}} := \left(\mathsf{k}, \mathsf{z}, \{\rho_i\}_{i\in[n]}, \{\mathsf{ct}_{i,1-\mathsf{z}_i}\}_{i\in[n]}\right).$$

Finally, it runs $m \leftarrow \mathcal{A}(pk)$. Note that r^{Gen} is used in the next step.

2. The experiment samples $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{H}}$. It then computes $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}, \mathsf{x}; \mathsf{r})$. For all $i \in [n]$ and $b \in \{0,1\}$, it also computes

$$\mathbf{e}_{i,b} \leftarrow \begin{cases} \mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}_{i,b}) & (b = \mathsf{x}_i) \\ \{0,1\}^\ell & (b \neq \mathsf{x}_i) \end{cases}.$$

It sets

$$CT := \left(\mathsf{y}, \begin{pmatrix} \mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1} \end{pmatrix}, R\mathsf{x} \oplus m\right) \quad \text{and} \quad r^{\mathsf{Enc}} = \left(\mathsf{x}, \mathsf{r}, \{\mathbf{e}_{i,1-\mathsf{x}_i}\}_{i\in[n]}\right).$$

Finally, it outputs $\mathsf{out} \leftarrow \mathcal{A}\left(CT, r^{\mathsf{Gen}}, r^{\mathsf{Enc}}\right)$.

Exp 1**:** This experiment is the same as Exp 0 except the followings. First, pk is generated together with a trapdoor of the chameleon hash function t as $(\mathsf{k}, \mathsf{t}) \leftarrow \mathsf{K}\left(1^\lambda\right)$ instead of $\mathsf{k} \leftarrow \widehat{\mathsf{K}}\left(1^\lambda\right)$. Moreover, all ciphertexts of chameleon encryption $\mathsf{ct}_{i,b}$ are computed by E_1, instead of $\widehat{\mathsf{E}_1}$. Specifically, for every $i \in [n]$ and $b \in \{0,1\}$, the experiment samples $\rho_{i,b} \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$ and compute $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1(\mathsf{k}, (i,b); \rho_{i,b})$. Also, it sets $r^{\mathsf{Gen}} = (\mathsf{k}, \mathsf{z}, \{\rho_{i,\mathsf{z}_i}\}_{i\in[n]}, \{\mathsf{ct}_{i,1-z_i}\}_{i\in[n]})$.

Lemma 5. *Assuming the oblivious samplability of* $\mathtt{CE}$*,* $\mathsf{Exp}\ 0 \approx \mathsf{Exp}\ 1$ *holds.*

Proof. Using $\mathcal{A}$, we construct a reduction algorithm $\mathcal{A}'^{O^*(\cdot,\cdot)}$ that attacks the oblivious samplability of $\mathtt{CE}$ and makes n oracle queries.

1. On receiving a hash key k^*, $\mathcal{A}'$ generates $\rho_i \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$ for every $i \in [n]$ and sets the public key as $pk = \left(\mathsf{k}^*, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right)$, where

$$\mathsf{ct}_{i,b} \leftarrow \begin{cases} \mathsf{E}_1(\mathsf{k}^*, (i,b); \rho_i) & (b = \mathsf{z}_i) \\ O^*(i,b) & (b \neq \mathsf{z}_i) \end{cases}.$$

   $\mathcal{A}'^{O^*(\cdot,\cdot)}$ also sets $r^{\mathsf{Gen}} = \left(\mathsf{k}, \mathsf{z}, \{\rho_i\}_{i\in[n]}, \{\mathsf{ct}_{i,1-\mathsf{z}_i}\}_{i\in[n]}\right)$. Then, $\mathcal{A}'^{O^*(\cdot,\cdot)}$ runs $\mathcal{A}(pk)$ and obtains m.
2. $\mathcal{A}'^{O^*(\cdot,\cdot)}$ simulates the step 2. of Exp 0 and Exp 1, and outputs what $\mathcal{A}$ does. Note that the step 2. of Exp 0 is exactly the same as that of Exp 1.

When playing $\mathsf{Exp}^{\text{os-0}}_{\mathtt{CE},\mathcal{A}}$ and $\mathsf{Exp}^{\text{os-1}}_{\mathtt{CE},\mathcal{A}}$, $\mathcal{A}'$ perfectly simulates Exp 0 and Exp 1 for $\mathcal{A}$, respectively. By the oblivious samplability of $\mathtt{CE}$,

$$\left|\Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 0] - \Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 1]\right| = \mathsf{Adv}^{\text{os}}_{\mathtt{CE},\mathcal{A}'}(\lambda) = \mathsf{negl}(\lambda)$$

holds. This proves $\mathsf{Exp}\ 0 \approx \mathsf{Exp}\ 1$.

$\mathsf{Exp}\ 2$**:** This experiment is the same as $\mathsf{Exp}\ 1$, except that we replace z contained in r^{Gen} by $\mathsf{z}' := \mathsf{z} \oplus \mathsf{x}$.

Because z distributes uniformly at random, so does z'. Therefore, the distribution of the inputs to $\mathcal{A}$ does not change between $\mathsf{Exp}\ 1$ and $\mathsf{Exp}\ 2$, and thus $\mathsf{Exp}\ 1 \equiv \mathsf{Exp}\ 2$ holds.

$\mathsf{Exp}\ 3$**:** The essential difference from $\mathsf{Exp}\ 2$ in this experiment is that when $\mathsf{z}_i = 0, \mathbf{e}_{i,1-\mathsf{x}_i}$ is computed by $\mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho_{i,1-\mathsf{x}_i})$ instead of uniformly sampled from $\{0,1\}^\ell$.
Additionally, each $\mathbf{e}_{i,\mathsf{x}_i}$ is replaced to $\mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho_{i,\mathsf{x}_i})$ from $\mathsf{D}(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}_{i,\mathsf{x}_i})$, though this does not change the distribution due to the correctness of $\mathtt{CE}$. After all, for every $i \in [n]$ and $b \in \{0,1\}$, the experiment computes

$$\mathbf{e}_{i,b} \leftarrow \begin{cases} \mathsf{E}_2(\mathsf{k}, \mathsf{y}; \rho_{i,b}) & (b = \mathsf{x}_i \vee \mathsf{z}_i = 0) \\ \{0,1\}^\ell & (b \neq \mathsf{x}_i \wedge \mathsf{z}_i = 1) \end{cases}.$$

Lemma 6. *If* $\mathtt{CE}$ *is correct and secure,* $\mathsf{Exp}\ 2 \approx \mathsf{Exp}\ 3$ *holds.*

Proof. This proof is done by hybrid arguments. We define $\mathsf{Exp}\ 2_j$ for every $j \in \{0, \ldots, n\}$ that are intermediate experiments between $\mathsf{Exp}\ 2$ and $\mathsf{Exp}\ 3$ as follows.

$\mathsf{Exp}\ 2_j$**:** This experiment is exactly the same as $\mathsf{Exp}\ 2$ except how $\mathbf{e}_{i,b}$ is generated for every $i \in [n]$. For $j < i \leq n$, $\mathbf{e}_{i,b}$ is generated as in $\mathsf{Exp}\ 2$. For $1 \leq i \leq j$, $\mathbf{e}_{i,b}$ is generated as in $\mathsf{Exp}\ 3$.

$\mathsf{Exp}\ 2_0$ is equal to $\mathsf{Exp}\ 2$, and $\mathsf{Exp}\ 2_n$ is equal to $\mathsf{Exp}\ 3$. In the following, we show $\mathsf{Exp}\ 2_{j-1} \approx \mathsf{Exp}\ 2_j$ for all $j \in [n]$.

In the case of $\mathsf{z}_j = 1$, except negligible probability, $\mathbf{e}_{j,\mathsf{x}_j}$ distributes identically in $\mathsf{Exp}\ 2_{j-1}$ and $\mathsf{Exp}\ 2_j$ because $\mathsf{E}_2\left(\mathsf{k}, \mathsf{y}; \rho_{j,\mathsf{x}_j}\right) = \mathsf{D}\left(\mathsf{k}, (\mathsf{x}, \mathsf{r}), \mathsf{ct}_{j,\mathsf{x}_j}\right)$ holds with overwhelming probability due to the correctness of $\mathtt{CE}$. Moreover, $\mathbf{e}_{j,1-\mathsf{x}_j}$ is generated in the same way in both experiments. Thus $\mathsf{Exp}\ 2_{j-1} \approx \mathsf{Exp}\ 2_j$ holds.

In the case of $\mathsf{z}_j = 0$, we show $\mathsf{Exp}\ 2_{j-1} \approx \mathsf{Exp}\ 2_j$ by constructing a reduction algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ and attacks the security of $\mathtt{CE}$. The description of $\mathcal{A}'$ is as follows.

1. $\mathcal{A}'$ samples $\mathsf{x} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{H}}$, outputs $(\mathsf{x}, \mathsf{r}, j)$, and receives $(\mathsf{k}^*, \mathsf{ct}^*, \mathbf{e}^*)$. Then, $\mathcal{A}'$ generates pk as follows. $\mathcal{A}'$ first samples $\mathsf{z} \xleftarrow{\$} \{0,1\}^n$ and sets $\mathsf{z}' = \mathsf{x} \oplus \mathsf{z}$. For every $(i,b) \in [n] \times \{0,1\}$ such that $(i,b) \neq (j, 1-\mathsf{x}_j)$, $\mathcal{A}'$ samples $\rho_{i,b} \xleftarrow{\$} \mathcal{R}_{\mathsf{E}}$ and computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1(\mathsf{k}, (i,b); \rho_{i,b})$. $\mathcal{A}'$ sets $\mathsf{ct}_{j,1-\mathsf{x}_j} := \mathsf{ct}^*$,

$$pk := \left(\mathsf{k}^*, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right) \quad \text{and} \quad r^{\mathsf{Gen}} = \left(\mathsf{k}^*, \mathsf{z}', \{\rho_{i,\mathsf{z}'_i}\}_{i \in [n]}, \{\mathsf{ct}_{i,1-\mathsf{z}'_i}\}_{i \in [n]}\right).$$

Finally, $\mathcal{A}'$ runs $m \leftarrow \mathcal{A}(pk)$. Note that $\rho_{j,\mathsf{z}'_i} = \rho_{j,\mathsf{x}_j \oplus \mathsf{z}_j} = \rho_{j,\mathsf{x}_j}$ since we consider the case of $\mathsf{z}_j = 0$, and thus $\mathcal{A}'$ generates ρ_{i,z'_i} by itself for every $i \in [n]$.

2. $\mathcal{A}'$ computes $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}^*, \mathsf{x}; \mathsf{r})$. For $j < i \leq n$, $\mathcal{A}'$ computes $\mathbf{e}_{i,b}$ as in Exp 2, and for $1 \leq i < j$, it does as in Exp 3. For $i = j$, $\mathcal{A}'$ computes $\mathbf{e}_{j,\mathsf{x}_j} \leftarrow \mathsf{E}_2\left(\mathsf{k}, \mathsf{y}; \rho_{j,\mathsf{x}_j}\right)$ and sets $\mathbf{e}_{j,1-\mathsf{x}_j} := \mathbf{e}^*$. Finally, $\mathcal{A}'$ sets

$$CT := \left(\mathsf{y}, \begin{pmatrix}\mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1}\end{pmatrix}, R\mathsf{x} \oplus m\right) \quad \text{and} \quad r^{\mathsf{Enc}} = \left(\mathsf{x}, \mathsf{r}, \{\mathbf{e}_{i,1-\mathsf{x}_i}\}_{i\in[n]}\right),$$

and outputs $\mathsf{out} \leftarrow \mathcal{A}\left(CT, r^{\mathsf{Gen}}, r^{\mathsf{Enc}}\right)$.

When playing $\mathsf{Exp}^1_{\mathtt{CE},\mathcal{A}'}$, $\mathcal{A}'$ simulates Exp 2_{j-1} for $\mathcal{A}$. Also, when playing $\mathsf{Exp}^0_{\mathtt{CE},\mathcal{A}'}$, $\mathcal{A}'$ simulates Exp 2_j for $\mathcal{A}$. By the security of $\mathtt{CE}$,

$$|\Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 2_{j-1}] - \Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 2_j]| = \mathsf{Adv}_{\mathtt{CE},\mathcal{A}'}(\lambda) = \mathsf{negl}(\lambda)$$

holds. From the above, we have

$$\begin{aligned} &|\Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 2] - \Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 3]| \\ \leq &\sum_{j\in[n]} |\Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 2_{j-1}] - \Pr[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 2_j]| = \mathsf{negl}(\lambda). \end{aligned}$$

We can conclude Exp 2 $\approx$ Exp 3.

Exp 4**:** This experiment is the same as Exp 3 except how y and r are computed. In this experiment, y is computed as $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{k}, 0^n; \mathsf{r})$. Moreover, the randomness r contained in r^{Enc} is replaced with $\mathsf{r}' \leftarrow \mathsf{H}^{-1}(\mathsf{t}, (0^n, \mathsf{r}), \mathsf{x})$.

Due to the trapdoor collision property of $\mathtt{CE}$, the view of $\mathcal{A}$ does not change between Exp 3 and Exp 4. Thus, Exp 3 $\equiv$ Exp 4 holds.

Exp 5**:** This experiment is the same as Exp 4, except that $R\mathsf{x}$ is replaced with $w \xleftarrow{\$} \{0,1\}^\mu$. Moreover, the experiment computes r' as $\mathsf{r}' \leftarrow \mathsf{H}^{-1}(\mathsf{t}, (0^n, \mathsf{r}), \mathsf{x}')$, and replaces x in r^{Enc} with x', where x' is a uniformly random string sampled from $\{0,1\}^n$ under the following two conditions:
- $R\mathsf{x}' = w$ holds.
- $\mathsf{x}'_i = \mathsf{x}_i$ holds for every $i \in [n]$ such that $\mathsf{z}_i = 1$.

Before showing Exp 4 $\approx$ Exp 5, we review a basic lemma on inversion sampling.

Lemma 7. *For a function $f : \mathcal{X} \to \mathcal{Y}$, we define two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ as $\mathcal{D}_1 = \left\{(x, y) \mid x \xleftarrow{\$} \mathcal{X}, y = f(x)\right\}$ and $\mathcal{D}_2 = \left\{(x', y) \mid x \xleftarrow{\$} \mathcal{X}, \right\}$ $y = f(x), x' \xleftarrow{\$} f^{-1}(y)$, where $f^{-1}(y)$ denotes the set of pre-images of y. Then, $\mathcal{D}_1$ and $\mathcal{D}_2$ are identical.*

Furthermore, we define a distribution $\mathcal{D}_3$ as $\mathcal{D}_3 = \left\{(x', y) \mid y \xleftarrow{\$} \mathcal{Y}, \right\}$ $x' \xleftarrow{\$} f^{-1}(y)$. If f has a property that $f(x)$ distributes uniformly at random over $\mathcal{Y}$ if the input x distributes uniformly at random over $\mathcal{X}$, $\mathcal{D}_1$ and $\mathcal{D}_3$ are identical.

Lemma 8. $\mathsf{Exp}\ 4 \approx \mathsf{Exp}\ 5$ *holds.*

Proof. According to the Chernoff bound on z,

$$\Pr\left[\mathsf{Ham}\left(\mathsf{z}\right) \geq (1+\delta)\frac{n}{2}\right] \leq e^{-\frac{\delta^2}{2+\delta}\frac{n}{2}}$$

holds for any $\delta > 0$. By taking $\delta = 1 - \frac{2\mu}{n}$, we have

$$\Pr\left[\mathsf{Ham}\left(\mathsf{z}\right) \geq n - \mu\right] \leq 2^{-\lambda} = \mathsf{negl}\left(\lambda\right).$$

Below, we show that $(\mathsf{x}, R\mathsf{x})$ in $\mathsf{Exp}\ 4$ has the same distribution as (x', w) in $\mathsf{Exp}\ 5$ in the case of $\mathsf{Ham}\left(\mathsf{z}\right) < n - \mu$, and complete the proof of this lemma.

We first introduce some notations. For an integer ordered set $\mathcal{I} \subset [n]$, we define $R_{\mathcal{I}}$ as the restriction of R to $\mathcal{I}$, that is $R_{\mathcal{I}} = (\mathbf{r}_1|\cdots|\mathbf{r}_{|\mathcal{I}|})$, where $R = (\mathbf{r}_1|\cdots|\mathbf{r}_n)$. We define $\mathsf{x}_{\mathcal{I}}$ in a similar way.

Fix any z which satisfies $\mathsf{Ham}\left(\mathsf{z}\right) < n - \mu$ and set $\mathcal{I} = \{i_k \in [n] \mid \mathsf{z}_{i_k} = 0\}$. Because $|\mathcal{I}| \geq \mu$, $R_{\mathcal{I}}$ is full rank due to the choice of R. Hence, $R_{\mathcal{I}} \cdot u$ is uniformly random over $\{0,1\}^{\mu}$ if u is uniformly random over $\{0,1\}^{|\mathcal{I}|}$.

Then, from Lemma 7 when setting $\mathcal{X} := \{0,1\}^{|\mathcal{I}|}, \mathcal{Y} := \{0,1\}^{\mu}$, and $f(u) = R_{\mathcal{I}} \cdot u$, the distribution of $(\mathsf{x}_{\mathcal{I}}, R_{\mathcal{I}} \cdot \mathsf{x}_{\mathcal{I}})$ and (u, w) are the same, where $\mathsf{x} \xleftarrow{\$} \{0,1\}^n, u \xleftarrow{\$} f^{-1}(w) = \left\{u' \in \{0,1\}^{|\mathcal{I}|} \mid R_{\mathcal{I}} \cdot u' = w\right\}$, and $w \xleftarrow{\$} \{0,1\}^{\mu}$. Moreover, we have $R\mathsf{x} = R_{\mathcal{I}} \cdot \mathsf{x}_{\mathcal{I}} \oplus R_{[n]\setminus\mathcal{I}} \cdot \mathsf{x}_{[n]\setminus\mathcal{I}}$. Since x' sampled in $\mathsf{Exp}\ 5$ is a bit string generated by replacing i_k-th bit of x with k-th bit of u for every $k \in [|\mathcal{I}|]$, we see that $(\mathsf{x}, R\mathsf{x})$ has the same distribution as $(\mathsf{x}', w \oplus R_{[n]\setminus\mathcal{I}} \cdot \mathsf{x}_{[n]\setminus\mathcal{I}})$. $(\mathsf{x}', w \oplus R_{[n]\setminus\mathcal{I}} \cdot \mathsf{x}_{[n]\setminus\mathcal{I}})$ also has the same distribution as (x', w) because w is sampled uniformly at random, and thus $(\mathsf{x}, R\mathsf{x})$ has the same distribution as (x', w). This completes the proof of Lemma 8.

Note that we can sample the above u in polynomial time, by computing a particular solution $v \in \{0,1\}^{|\mathcal{I}|}$ of $R_{\mathcal{I}} \cdot v = w$, and add a vector sampled uniformly at random from the kernel of $R_{\mathcal{I}}$.

$\mathsf{Exp}\ 6$**:** This experiment is the same as $\mathsf{Exp}\ 6$ except that w is replaced with $w \oplus m$. By this change, CT is of the form

$$CT := \left(\mathsf{y}, \begin{pmatrix}\mathbf{e}_{1,0}, \ldots, \mathbf{e}_{n,0} \\ \mathbf{e}_{1,1}, \ldots, \mathbf{e}_{n,1}\end{pmatrix}, w\right).$$

Moreover, x' contained in r^{Enc} is sampled so that $R\mathsf{x}' = m \oplus w$ holds.

Since w is uniformly at random, so is $w \oplus m$. Thus, $\mathsf{Exp}\ 5 \equiv \mathsf{Exp}\ 6$ holds.

We see that $\mathsf{Exp}\ 6$ is the same as $\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{NCE},\mathcal{A}}$. Put all the above arguments together, we have

$$\mathsf{Adv}_{\mathtt{NCE},\mathcal{A}}\left(\lambda\right) \leq \left|\Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 0\right] - \Pr\left[\mathsf{out} = 1 \text{ in } \mathsf{Exp}\ 6\right]\right| = \mathsf{negl}\left(\lambda\right).$$

Hence $\mathtt{NCE}$ is secure. This completes the proof of Theorem 3.

5.3 Ciphertext Rate

Finally, we evaluate the ciphertext rate of $\mathtt{NCE}$. From Theorem 2, in order to make $\mathtt{NCE}$ correct, it is sufficient to set $\ell = \mathsf{poly}(\log \lambda)$. Moreover, from Theorem 3, in order to make $\mathtt{NCE}$ secure, it is sufficient to set $\mu = \mathcal{O}(\lambda)$ and $n = 8\mu$. In this setting, the ciphertext length of $\mathtt{NCE}$ is $|CT| = \lambda + 2n\ell + \mu$. Note that we assume a group element of $\mathbb{G}$ is described as a λ-bit string. Then, the ciphertext rate of $\mathtt{NCE}$ is evaluated as

$$\frac{|CT|}{\mu} = \frac{\lambda + 2n\ell + \mu}{\mu} = \mathcal{O}(\ell) = \mathsf{poly}(\log \lambda).$$

6 Full Construction of Proposed NCE

In the basic construction, we construct an NCE scheme with correctness $\gamma = 1 - \mathsf{negl}(\lambda)$, by setting $\ell = \mathsf{poly}(\log \lambda)$ which is the output length of E_2 (and thus D) of the underlying $\mathtt{CE}$. Of course, if we set ℓ to $\mathcal{O}(\log \lambda)$, we can make the ciphertext rate of the resulting NCE scheme $\mathcal{O}(\log \lambda)$. However, this modification also affects the correctness of the resulting NCE scheme. γ is no longer $= 1 - \mathsf{negl}(\lambda)$, and is at most $1 - 1/\mathsf{poly}(\lambda)$.

Fortunately, we can amplify the correctness of the scheme to $1 - \mathsf{negl}(\lambda)$ from enough large constant without changing the ciphertext rate. For that purpose, we use a constant-rate error-correcting code which can correct errors up to some constant fraction. Concretely, we modify the scheme as follows. In the encryption, we first encode the plaintext by the error-correcting code and parse it into N blocks of length μ. Then, we encrypt each block by the γ-correct NCE scheme for a constant γ using different public keys. The decryption is done naturally, i.e., decrypt each ciphertext, concatenate them, and decode it. The ciphertext rate is still $\mathcal{O}(\log \lambda)$ because the rate of error-correcting code is constant.

This block-wise encryption technique not only amplifies the correctness but also reduces the public key size. In the basic construction, the size of a public key depends on the length of a message quadratically. However, by applying the block-wise encryption technique, it becomes linear in the length of a message.

The description of the full construction is as follows. Let $\mathtt{ECC} = (\mathsf{Encode}, \mathsf{Decode})$ be a constant-rate error-correcting code which can correct errors up to ϵ-fraction of the codeword where $\epsilon > 0$ is some constant.

Specifically, given a message $m \in \{0,1\}^{\mu M}$, Encode outputs a codeword $\overrightarrow{CW} \in \{0,1\}^{\mu N}$. If $\mathsf{Ham}\left(\overrightarrow{CW} - \overrightarrow{CW'}\right) \leq \epsilon \mu N$, $\mathsf{Decode}\left(\overrightarrow{CW'}\right) = m$. The rate of $\mathtt{ECC}$ is some constant N/M.

Let $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ be an NCE scheme whose message space is $\{0,1\}^{\mu}$, ciphertext rate is $\mathcal{O}(\log \lambda)$, and correctness is $\gamma = 1 - \frac{\epsilon}{2}$. We construct $\overrightarrow{\mathtt{NCE}} = (\overrightarrow{\mathsf{Gen}}, \overrightarrow{\mathsf{Enc}}, \overrightarrow{\mathsf{Dec}}, \overrightarrow{\mathsf{Sim}_1}, \overrightarrow{\mathsf{Sim}_2})$ as follows. The message space of $\overrightarrow{\mathtt{NCE}}$ is $\{0,1\}^{\mu M}$.

$\overrightarrow{\mathsf{Gen}}\left(1^\lambda; \overrightarrow{r^{\mathsf{Gen}}}\right)$:

- Parse the given random coin to $\overrightarrow{r^{\mathsf{Gen}}} = \left(r_1^{\mathsf{Gen}}, \ldots, r_N^{\mathsf{Gen}}\right)$.
- For all $i \in [N]$, generate key pairs $(pk_i, sk_i) \leftarrow \mathsf{Gen}\left(1^\lambda; r_i^{\mathsf{Gen}}\right)$.
- Output $\overrightarrow{pk} := (pk_1, \ldots, pk_N)$ and $\overrightarrow{sk} := (sk_1, \ldots, sk_N)$.

$\overrightarrow{\mathsf{Enc}}\left(\overrightarrow{pk}, m; \overrightarrow{r^{\mathsf{Enc}}}\right)$:

- Parse $\overrightarrow{r^{\mathsf{Enc}}} = \left(r_1^{\mathsf{Enc}}, \ldots, r_N^{\mathsf{Enc}}\right)$.
- Compute $\overrightarrow{CW} \leftarrow \mathsf{Encode}(m)$ and parse $\overrightarrow{CW} = (CW_1, \ldots, CW_N)$.
- For all $i \in [N]$, compute $CT_i \leftarrow \mathsf{Enc}\left(pk_i, CW_i; r_i^{\mathsf{Enc}}\right)$.
- Output $\overrightarrow{CT} := (CT_1, \ldots, CT_N)$.

$\overrightarrow{\mathsf{Dec}}\left(\overrightarrow{sk}, \overrightarrow{CT}\right)$:

- For all $i \in [N]$, Compute $CW_i' \leftarrow \mathsf{Dec}(sk_i, CT_i)$.
- Concatenate them as $\overrightarrow{CW'} := (CW_1', \ldots, CW_N')$.
- Output $m \leftarrow \mathsf{Decode}\left(\overrightarrow{CW'}\right)$.

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

- For all $i \in [N]$, compute $(pk_i, CT_i, st_i) \leftarrow \mathsf{Sim}_1\left(1^\lambda\right)$,
- Output $\overrightarrow{pk} := (pk_1, \ldots, pk_N)$, $\overrightarrow{CT} := (CT_1, \ldots, CT_N)$, and $\overrightarrow{st} := (st_1, \ldots, st_N)$.

$\overrightarrow{\mathsf{Sim}_2}\left(m, \overrightarrow{st}\right)$:

- Compute $\overrightarrow{CW} \leftarrow \mathsf{Encode}(m)$ and parse $(CW_1, \ldots, CW_N) \leftarrow \overrightarrow{CW}$.
- For all $i \in [N]$, compute $\left(r_i^{\mathsf{Gen}}, r_i^{\mathsf{Enc}}\right) \leftarrow \mathsf{Sim}_2(CW_i, st_i)$.
- Output $\overrightarrow{r^{\mathsf{Gen}}} := \left(r_1^{\mathsf{Gen}}, \ldots, r_N^{\mathsf{Gen}}\right)$ and $\overrightarrow{r^{\mathsf{Enc}}} := \left(r_1^{\mathsf{Enc}}, \ldots, r_N^{\mathsf{Enc}}\right)$.

Correctness. We can prove the correctness of $\overrightarrow{\mathtt{NCE}}$ by the Chernoff bound. Formally, we have the following theorem. See the full version for the proof.

Theorem 4. *Let* ECC *be an constant-rate error-correcting code which can correct errors up to ϵ-fraction of a codeword. Let* NCE *be a γ-correct NCE scheme, where $\gamma = 1 - \frac{\epsilon}{2}$. If the number of parsed codeword $N \geq \mathsf{poly}(\log \lambda)$, the above $\overrightarrow{\mathtt{NCE}}$ is correct.*

Security. For the security of $\overrightarrow{\mathtt{NCE}}$, we have the following theorem. Since we can prove it via a straightforward hybrid argument, we omit it.

Theorem 5. *If* NCE *is an secure NCE scheme, then $\overrightarrow{\mathtt{NCE}}$ is also secure.*

Ciphertext Rate. Since rate of the error-correcting code N/M is constant, the ciphertext rate of $\overrightarrow{\mathtt{NCE}}$ is $\frac{N|CT|}{\mu M} = \mathcal{O}(\ell) = \mathcal{O}(\log \lambda)$.

Acknowledgements. A part of this work was supported by NTT Secure Platform Laboratories, JST OPERA JPMJOP1612, JST CREST JPMJCR14D6, JSPS KAKENHI JP16H01705, JP17H01695, JP19J22363.

References

1. Beaver, D.: Plug and play encryption. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052228
2. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_17
3. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 535–564. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_20
4. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC, pp. 639–648 (1996)
5. Canetti, R., Poburinnaya, O., Raykova, M.: Optimal-rate non-committing encryption. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 212–241. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_8
6. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 33–65. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_2
7. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_17
8. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_27
9. Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 372–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_13
10. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
11. Döttling, N., Garg, S., Hajiabadi, M., Masny, D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 3–31. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_1
12. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_1
13. Garg, S., Gay, R., Hajiabadi, M.: New techniques for efficient trapdoor functions and applications. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 33–63. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_2

14. Garg, S., Hajiabadi, M.: Trapdoor functions from the computational Diffie-Hellman assumption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 362–391. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_13
15. Garg, S., Ostrovsky, R., Srinivasan, A.: Adaptive garbled RAM from laconic oblivious transfer. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 515–544. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_18
16. Garg, S., Srinivasan, A.: Adaptively secure garbling with near optimal online complexity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 535–565. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_18
17. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_16
18. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055744
19. Hemenway, B., Ostrovsky, R., Richelson, S., Rosen, A.: Adaptive security with quasi-optimal rate. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016, Part I. LNCS, vol. 9562, pp. 525–541. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_22
20. Hemenway, B., Ostrovsky, R., Rosen, A.: Non-committing encryption from ϕ-hiding. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 591–608. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_24
21. Tao, T., Vu, V.: On the singularity probability of random Bernoulli matrices. J. Am. Math. Soc. **20**(3), 603–628 (2007)