論文 / 著書情報
Article / Book Information

題目(和文)	暗号文拡大率が定数である Non-Committing 暗号の構成
Title(English)	Constructions for Non-Committing Encryption with Constant Ciphertext Expansion
著者(和文)	吉田雄祐
Author(English)	Yusuke Yoshida
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第12169号, 授与年月日:2022年9月22日, 学位の種別:課程博士, 審査員:田中 圭介,伊東 利哉,尾形 わかは,鹿島 亮,安永 憲司
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第12169号, Conferred date:2022/9/22, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Constructions for Non-Committing Encryption with Constant Ciphertext Expansion

by
Yusuke YOSHIDA

Supervisor: Keisuke TANAKA
School of Computing
Tokyo Institute of Technology

August 25, 2022

Abstract

Public-key encryption is a cryptographic primitive used for establishing a secure channel between two parties. Depending on the situation and purpose, public-key encryption schemes must satisfy appropriate security requirements.

Non-committing encryption (NCE) is a public-key encryption scheme that satisfies a security notion essential to establish a secure channel in adaptively secure multi-party computation (MPC) protocols. Informally, NCE can generate a dummy ciphertext. The dummy ciphertext is indistinguishable from the real ciphertext. Moreover, we can explain the dummy ciphertext as encryption of an arbitrary message by producing consistent randomness. It has been a challenge in the theory of adaptively secure MPC to find an NCE scheme with small ciphertext expansion (required ciphertext length per bit of message). This thesis proposes the first NCE schemes with constant ciphertext expansion in the standard model (i.e., without assuming the randomness used in the cryptosystem can be securely erased or use of the random oracle). We show two instantiations of the scheme, one from the Decisional Diffie-Hellman (DDH) problem and another from the Learning with Errors (LWE) problem.

Before constructing the constant ciphertext expansion scheme, we demonstrate the new approach to constructing NCE through the construction of a simpler NCE scheme based on obliviously samplable key-encapsulation mechanism (KEM). The ciphertext expansion of this simpler scheme is $\mathcal{O}(\lambda)$ for security parameter λ. In detail, we use KEM to construct a weak NCE scheme with $\mathcal{O}(\lambda)$ ciphertext expansion. Weak NCE is NCE where its correctness and security requirements are weakened. Then weak NCE is amplified to a full-fledged NCE scheme using an information theoretical primitive called wiretap codes. This amplification increases the cipher text expansion only by a constant factor. The constant ciphertext expansion scheme is obtained by using a primitive called obliviously samplable chameleon encryption, instead of KEM in the above construction. We show instantiations of obliviously samplable chameleon encryption based on the DDH and the LWE problems.

Note on This Thesis

This thesis reorganizes two papers those construct non-committing encryption schemes[YKT19, YKXT20].

Historically, the first paper introduced obliviously samplable chameleon encryption to construct an NCE scheme with $\mathcal{O}(\log \lambda)$ ciphertext expansion. This scheme is instantiated based on the DDH problem.

The second paper improved the first paper by introducing the notion of weak NCE and the use of wiretap codes. This work added instantiation based on the LWE problem, which turned out to have a smaller public-key expansion than the DDH-based scheme. Eventually, we achieved to construct NCE schemes with a constant ciphertext expansion.

This thesis excludes several obsolete parts in the above papers. For example, we do not describe the construction of $\mathcal{O}(\log \lambda)$ ciphertext expansion scheme because the constant ciphertext expansion scheme is properly improved. Instead, we describe the construction of $\mathcal{O}(\lambda)$ ciphertext expansion scheme, that is constructed from obliviously samplable KEM. This scheme is not very well in terms of ciphertext expansion, still, it contains the essence of the proposed NCE construction. We also update several security definitions and related proofs of NCE and its building blocks such as chameleon encryption. Especially, we start to focus on the randomness used by the algorithms including the simulators which appear only in the security definition. Security notions of the building blocks are defined similarly to the definition of NCE. We hope these definitions give a new viewpoint to understanding the nature of NCE.

Acknowledgements

本学位論文は東京工業大学 情報理工学院 数理・計算科学系 田中圭介研究室に在籍している間に行なった研究をまとめたものです.

初めに，指導教員である田中圭介教授に深く感謝します．先生の研究室にめぐり合うことが無ければこの研究が始まることもなく，先生の温かい指導と支援が無ければ満足してこの研究を完了させることもなかったでしょう.

次に，本論文の審査員を引き受けてくださった伊東利哉教授，尾形わかは教授，鹿島亮准教授，安永憲司准教授に感謝いたします．発表に対するコメントを通して，本研究に対する理解をさらに深めることができました.

本論文の研究に際し，多くの助言を頂きました北川冬航氏と草川恵太氏には特段の感謝を申し上げます．彼らとの議論を通して，私だけでは見通せなかったであろう深い知見を得ることができました．とりわけ北川氏にいただいた励ましと勇気は私の大きな支えでありました.

田中研究室のメンバー，元メンバー，関係者の方々を中心に，西 8 号館 11 階で共に楽しい時間を過ごした皆様に感謝します．彼らと過ごした時間を通して，私だけでは見渡せなかったであろう広い知識と経験に触れることができました．特に，原啓祐氏と手塚真徹氏とは修士，博士課程を通しての多くのかけがえの無い経験を共有しました.

最後にこれまで理解と支援，応援をし続けてくれた家族に感謝します.

なお本研究は JSPS 科研費 JP19J22363 の助成を受けて行いました.

Contents

Chapter 1

Introduction

Public-key encryption, the most fundamental cryptographic primitive for realizing secure message transmission, has a number of security notions, which are required depending on the situations in which it is used. When it is used in adaptively secure multi-party computation, we need adaptively secure public-key encryption, also known as non-committing encryption.

1.1 Backgrounds

In secure multi-party computation (MPC) protocols, a group of parties can compute some function of their private inputs by communicating with each other. The security requirement of MPC protocols is that through the protocol, each party (possibly, some of them collude) cannot obtain information on the other party's input, except what is trivially extracted from the output of the function. To model this security requirement, we consider an algorithm called adversary. The adversary corrupts some parties and tries to extract some non-trivial information about the inputs of non-corrupted parties. Informally the protocol is secure if we can simulate what the adversary can see during the protocol (in the case of public-key encryption, all the public keys and ciphertexts, and secret keys and randomness used by the corrupted parties) only from what is trivially clear to the adversary in an idealized world (i.e. input of corrupted parties).

Depending on when the adversary determines to corrupt parties, two types of adversarial settings, called static and adaptive, have been considered for MPC. In the static 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.

We premise there are authenticated channels between each pair of parties. Furthermore, if the provided channels are private, information-theoretically secure MPC protocols

such as those proposed by Ben-Or et al. [BGW88] and Chaum et al. [CCD88] are secure against adaptive adversaries. On the other hand, for the MPC protocols relying on complexity assumption such as the one proposed by Goldreich et al. [GMW87], the security proof fails against an adaptive adversary as observed by Damgård and Nielsen [DN00].

In order to use adaptively secure protocols without private channels are provided, we have to establish private channels by using a public-key encryption scheme. For this aim, *non-committing encryption (NCE)* was introduced by Canetti et al. [CFGN96]. 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 encryption randomness that "explain" the ciphertext as an encryption of the message.

At first glance, it is weird to consider revealing a secret key to the adversary in a security definition. But due to this property, NCE can be used to establish secure communication on the adaptively secure MPC. This is because when the adaptive adversary corrupts the sender and the receiver, it can obtain the secret key and encryption randomness. To simulate such corruption, we must simulate the secret key and the randomness that is consistent with the simulated public key and ciphertext, this is where the property of NCE works. To summarize briefly, the security of NCE is not only for the confidentiality of the messages between non-corrupted parties but also for the security of the entire MPC protocol, although messages corrupted by the adversary are revealed.

Canetti et al. showed that the information-theoretically secure MPC protocols are still adaptively secure if private channels are replaced by NCE over insecure channels (assumed they are authenticated). Canetti, Lindell, Ostrovsky, and Sahai [CLOS02] also showed a slightly augmented version of NCE is useful to achieve adaptive security in the universally composable (UC) setting.

The ability to open a dummy ciphertext to any message is generally achieved at the price of efficiency. This is in contrast to the ordinary public-key encryption for which we can easily obtain schemes the size of whose ciphertext is $n + \mathsf{poly}(\lambda)$ through the hybrid encryption methodology, where n is the length of an encrypted message and λ is the security parameter. Thus, many previous works have focused on constructing efficient NCE schemes. Especially, they tried to improve *ciphertext expansion* which is the asymptotic ratio of ciphertext length and message length since ciphertext length dominates the online communication complexity of the protocol.

Indeed, a textbook on secure multiparty computation raised this as an open problem:

> "Finding a non-committing encryption scheme that can encrypt κ bits using the order of κ bits of ciphertext is an important open problem in the theory of adaptive secure multiparty computation."
>
> — Cramer et al. , Secure Multiparty Computation and Secret Sharing [CDN15]

1.2 Existing Non-Committing Encryption Schemes

Canetti et al. [CFGN96] constructed the first NCE scheme, based on common-domain trapdoor permutations which can be instantiated from the computational Diffie-Hellman (CDH) or RSA problem. Ciphertext expansion of their scheme is $\mathcal{O}(\lambda^2)$.

Choi, Dachman-Soled, Malkin, and Wee [CDMW09] constructed an NCE scheme with ciphertext expansion $\mathcal{O}(\lambda)$ from trapdoor simulatable PKE. Their construction can be instantiated under many computational problems including factoring problem, since many existing (ordinary) PKE schemes satisfy trapdoor simulatability.

The first NCE scheme with sub-linear ciphertext expansion was proposed by Hemenway, Ostrovsky, and Rosen [HOR15]. They proposed an NCE scheme with ciphertext expansion $\mathcal{O}(\log n)$ for n-bit messages based on the Φ-hiding problem, which we can easily modify its ciphertext expansion to $\mathcal{O}(\log \lambda)$ by dividing long messages to λ-bit blocks. Hemenway, Ostrovsky, Richelson, and Rosen [HORR16] also showed constructions of NCE with ciphertext expansion $\mathsf{poly}(\log \lambda)$ from the learning with errors (LWE) and Ring-LWE problems.

Canetti, Poburinnaya, and Raykova [CPR17] studied the construction of NCE in the common reference strings (CRS) model. They achieved optimal ciphertext expansion $1 + o(1)$ assuming the existence of indistinguishability obfuscation (iO) and one-way function.

Yoshida, Kitagawa, and Tanaka [YKT19] constructed an NCE scheme with ciphertext expansion $\mathcal{O}(\log \lambda)$ from a primitive called chameleon encryption (CE), which additionally satisfies oblivious samplability. They showed an instantiation of obliviously samplable CE based on the decisional Diffie-Hellman (DDH) problem.

Yoshida, Kitagawa, Xagawa, and Tanaka [YKXT20] improved their previous scheme and constructed NCE scheme with constant ciphertext expansion from obliviously samplable chameleon encryption. They also showed another instantiation of obliviously samplable chameleon encryption based on the LWE problem, which reduces public-key size of the constructed NCE scheme.

Concurrently to Yoshida et al. [YKXT20], Brakerski, Branco, Döttling, Garg, and Malavolta [BBD+20] proposed NCE schemes with constant ciphertext expansion from the LWE and DDH problems. Note that a previous version of their work claimed a constant ciphertext-expansion NCE from the quadratic residuosity (QR) assumption. This result was retracted due to a bug in the QR construction. They introduced a primitive called Packed Encryption with Partial Equivocality (PEPE) as a building block to construct their NCE scheme. Their construction basically follows the framework by Hemenway et al. [HORR16], whose origin further backs to Choi et al. [CDMW09].

We show the list of existing NCE schemes in Table 1.1.

Note on NCE Schemes on Composite Order Group. When constructing an NCE scheme using a composite number $N = pq$, we should be careful in how keys are generated. In the key generation algorithm, if the composite number $N = pq$ is sampled

Reference	CT Expansion	PK Expansion	Assumption
Canetti et al. [CFGN96]	$\mathcal{O}(\lambda^2)$	$\mathcal{O}(\lambda^2)$	Common-Domain TDP
Choi et al. [CDMW09]	$\mathcal{O}(\lambda)$	$\mathcal{O}(\lambda)$	Trapdoor Simulatable PKE
Hemenway et al. [HOR15]	$\mathcal{O}(\log \lambda)$	$\lambda \cdot \mathsf{poly}(\log \lambda)$	Φ-hiding
Hemenway et al. [HORR16]	$\mathsf{poly}(\log \lambda)$	$\lambda \cdot \mathsf{poly}(\log \lambda)$	LWE
Hemenway et al. [HORR16]	$\mathsf{poly}(\log \lambda)$	$\mathsf{poly}(\log \lambda)$	Ring-LWE
Canetti et al. [CPR17] $^{(*)}$	$1 + o(1)$	$1 + o(1)$	Indistinguishability Obfuscation
Yoshida et al. [YKT19]	$\mathcal{O}(\log \lambda)$	$\mathcal{O}(\lambda^2)$	Obliviously Samplable CE (DDH)
Brakerski et al. [BBD+20]	$\mathcal{O}(1)$	$\mathcal{O}(\lambda^2)$	PEPE (DDH)
Brakerski et al. [BBD+20]	$\mathcal{O}(1)$	$\lambda \cdot \mathsf{poly}(\log \lambda)$	PEPE (LWE)
Yoshida et al. [YKXT20]	$\mathcal{O}(1)$	$\mathcal{O}(\lambda^2)$	Obliviously Samplable CE (DDH)
Yoshida et al. [YKXT20]	$\mathcal{O}(1)$	$\lambda \cdot \mathsf{poly}(\log \lambda)$	Obliviously Samplable CE (LWE)

Table 1.1: Comparison of existing (2-round) NCE schemes in terms of their ciphertext and public-key expansion. The security parameter is denoted by λ. $^{(*)}$ This scheme uses common reference strings.

straightforwardly, i.e. sampling two primes p, q and outputs its product, we cannot use hardness assumptions over the composite number in the security proof of non-committing encryption. This is because the adversary can obtain the randomness used in the key generation, which contains the factorization of N. A way to avoid this problem is to sample a Blum integer $N = pq$ without knowing its factorization, however, unfortunately, we do not know such an algorithm.

For this reason, NCE schemes constructed based on the hardness of the RSA or factorization of Blum integer [CFGN96, CDMW09] use Bach's algorithm [Bac88, Kal03] to generate a random (not necessarily Blum) integer N, together with its factorization. Since N is a random integer, we can explain to the adversary that N is sampled obliviously. However, the integer N sampled in this way may not be used as a public key for the RSA or Rabin encryption because N may be a prime number, or it may be a composite number that does not contain a large prime factor, such as a power of two. Therefore we generate a sufficiently large number of $\{N_i\}$, so that we can expect that one of them will be a Blum integer $N_i = pq$, which is a secure public key. We can use an amplification technique, that executes number of encryption for each N_i, so that if one of the ciphertexts is encrypted under secure public-key N_i, the entire scheme is secure. In this way, we can construct an NCE scheme over the composite number, but this approach incurs $\mathcal{O}(\lambda^2)$ overhead on the ciphertext and public key. This is because the probability of a random integer being a Blum number is $\Omega(1/\lambda^2)$ [RS94, GM06].

For the case of NCE schemes constructed based on the hardness of the quadratic residuosity problem [BBD+20, LCC06], it seems necessary to use the CRS model in which a trusted party generates the composite number $N = pq$ and put it to the CRS.

1.3 Related Works

Multi-Round NCE Protocols. In this thesis, we focus on NCE in a narrow sense, that is, public-key encryption that satisfies the non-committing security because it is a simplest form of non-committing encryption. However, in order to realize secure channels in adaptively secure MPC protocols, it is not necessarily for NCE to be a public-key encryption, i.e. 2-round protocol. Indeed, in some literature, the terminology NCE was also used in a broader sense to indicate multi-round adaptively secure message transmission protocols.

Beaver [Bea97] proposed a 3-round NCE scheme with ciphertext expansion $\mathcal{O}(\lambda)$ based on the decisional Diffie-Hellman (DDH) problem. Damgård and Nielsen [DN00] generalized Beaver's protocol and proposed a 3-round NCE scheme with ciphertext expansion $\mathcal{O}(\lambda)$ based on a primitive called simulatable PKE, which can be instantiated based on concrete problems such as the DDH, computational Diffie-Hellman (CDH), and learning with errors (LWE) problems. Lei, Chen, and Chen [LCC06] proposed an instantiation of 3-round NCE protocol based on the quadratic residuosity (QR) problem in order to reduce computational costs of the protocol.

Zhu, Araragi, Nishide, and Sakurai [ZANS10] proposed 4-round NCE protocol and analyzed it in the universally composable framework.

NCE in the Secure Erasure Model. Beaver and Haber [BH93] showed if honest parties are assumed to be able to erase sensitive local information completely, then adaptively secure MPC can be obtained efficiently. However, as discussed by Canetti et al. [CFGN96], such trusted erasure may be unrealistic in many scenarios.

NCE in the Random Oracle Model. Nielsen [Nie01] pointed out constructing NCE in the random oracle model is easy.

Nielsen [Nie02] show that NCE is a separation between the random oracle model and the non-programable random oracle model.

Camenisch, Lehmann, Neven, and Samelin [CLNS17] proposed a UC secure NCE scheme in the random oracle model.

1.4 This Thesis

We propose the first NCE schemes with constant ciphertext expansion without the use of iO or CRS. Along the way, we propose an alternative paradigm to construct NCE, that differs from the paradigm proposed by Choi et al. [CDMW09] and used by NCE schemes to date [HOR15, HORR16, BBD+20]. By instantiating our paradigm with obliviously samplable key-encapsulation mechanism, we obtain an NCE scheme that has similar parameter to the NCE scheme by Choi et al. . The NCE schemes with constant ciphertext

expansion are constructed by instantiating our new paradigm with obliviously samplable chameleon encryption proposed as we described in [YKT19, YKXT20].

We show that obliviously samplable CE can be realized based on the DDH problem and the LWE problem for super-polynomially large modulus. Thus, we obtain constant ciphertext expansion NCE schemes based on the DDH problem and LWE problem.

One of the disadvantage of the DDH-based NCE scheme is its relatively large public-key size. The size of public key for each message bit of the DDH-based scheme is $\mathcal{O}(\lambda^2)$. Our LWE based NCE scheme improves public-key size compared to the DDH-based scheme. The size of the public key for each message bit of our LWE based scheme is $\lambda \cdot \mathsf{poly}(\log \lambda)$. This is the same as that of NCE schemes proposed by Brakerski et al. [BBD+20] or Hemenway et al. [HORR16], which are also based on the LWE problem for super-polynomially large modulus.

Weak Non-Committing Encryption. Our starting point is the observation that by adjusting the parameters of an NCE scheme proposed in [YKT19], its ciphertext expansion can be reduced to a constant, at the cost of its perfect correctness and security.

Specifically, the scheme only satisfies *weak correctness*, which means that each bit of decrypted plaintext is flipped with constant probability. Moreover, the scheme only satisfies *weak security* that only guarantees the secrecy of some part of encrypted plaintexts. In Section 3.3, we formally define weak correctness and weak security for NCE and introduce the notion of *weak NCE* as NCE satisfying only those weak correctness and weak security. As a demonstration, we construct a weak NCE scheme from obliviously samplable KEM.

In Section 4.5, we give the description of the above scheme and its building block, obliviously samplable CE. Then we prove that the scheme is indeed a weak NCE scheme.

Amplification for Non-Committing Encryption. Next, we show that we can amplify a weak NCE scheme into a full-fledged NCE scheme in Section 3.2. As a tool of amplification, we use a coding scheme called *wiretap codes*. More specifically, we define a new security property, *conditional invertibility* for wiretap codes, which is essentially a non-committing security for the wiretap codes. We show an instantiation of wiretap codes constructed from randomness extractor and linear error-correcting codes satisfies the conditional invertibility.

This amplification increases the ciphertext expansion by only a constant factor. Thus, by applying this transformation to the weak NCE scheme shown in Section 4.5, we obtain an NCE scheme with a constant ciphertext expansion.

DDH-Based Instantiation. We propose a DDH-based instantiation of obliviously samplable CE in Section 4.3. The construction is similar to the chameleon encryption scheme based on the CDH problem, proposed by Döttling and Garg [DG17b].

A natural question would be "why we need to rely on the DDH assumption, not CDH?" This is because a hash key and a ciphertext of the chameleon encryption scheme together

form multiple Diffie-Hellman tuples. Thus, it seems difficult to sample them obliviously unless we prove that the knowledge of exponent assumption [HT98, BP04] 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 in the above sense by sampling random group elements directly from the underlying group.

Lattice-Based Instantiation. We propose a lattice-based instantiation of obliviously samplable CE in Section 4.4. The construction is a natural composition of the lattice-based hash encryption by Döttling et al. [DGHM18] and the lattice-based chameleon hash functions by Cash et al. [CHKP10].

One drawback of our construction is that we need the modulus of lattices to be super-polynomially large for the correctness of it. This seems unavoidable since the chameleon encryption [1] implies non-interactive key exchange, which is considered difficult to be realized from lattice problems for polynomially large modulus as discussed by Guo et al. [GKRS20].

1.5 Notations

In this paper, PPT denotes probabilistic polynomial time. $x \leftarrow X$ denotes an element x is sampled from uniform distribution over a set X. $y \leftarrow \mathsf{A}(x; r_{\mathsf{A}})$ denotes that probabilistic algorithm A takes x as input, outputs y using internal randomness r_{A}. $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$ denotes function $\varepsilon(\cdot)$ is negligible, that is, $\varepsilon(\lambda) = \lambda^{-\omega(1)}$ holds.

For an integer n, $[n]$ denotes a set $\{1, \ldots, n\}$. For a subset $\mathcal{I} \subset [n]$ and a vector $x = (x_i)_{1 \le i \le n} \in \{0,1\}^n$, $x_{\mathcal{I}}$ denotes $(x_i)_{i \in \mathcal{I}}$. For a matrix $M = (\boldsymbol{m}_i)_{1 \le i \le n} \in \{0,1\}^{k \times n}$, $M_{\mathcal{I}} \in \{0,1\}^{k \times |\mathcal{I}|}$ denotes the matrix composed from column vectors $\boldsymbol{m}_i$ of M for $i \in \mathcal{I}$.

$h_2(\cdot)$ denotes the binary entropy function, $h_2(p) = -p \log p - (1-p) \log (1-p)$. $H(Y|X)$ denotes the conditional entropy.

[1] At least, for the current formalization of chameleon encryption, it implies non-interactive key exchange. There leaves the possibility of constructing NCE scheme from poly-modulus LWE via introducing a relaxed version of chameleon encryption.

Chapter 2

Basics and Definitions of NCE

2.1 Overview

In this chapter, we introduce the definition of non-committing encryption. Before describing concrete definitions, we briefly and informally explain how to define semantic security for a general algorithm A, which we consider as a generalization of key generation or encryption of primitives appear in this thesis. Then we show how to define non-committing security and oblivious samplability as its extension.

Semantic Security Consider an execution of algorithm $y \leftarrow \mathsf{A}(x; r_{\mathsf{A}})$, where we want to define that the information on input x is hidden.

Semantic security (or static security as opposed to adaptive security) is defined in the following style: There exists a simulator algorithm Sim, such that for all x,

$$\mathsf{A}(x; r_{\mathsf{A}}) \stackrel{c}{\approx} \mathsf{Sim}(1^{\lambda}; r_{\mathsf{Sim}})$$

holds. The left-hand side is often called real-life execution and the right-hand side is ideal-world execution.

Oblivious Samplability We define oblivious samplability [1] in the following style: There exists an obliviously sampling algorithm $\widehat{\mathsf{A}}$ and an invert sampling algorithm $\mathsf{Inv}_{\mathsf{A}}$, such that for all x,

$$(\mathsf{A}(x; r_{\mathsf{A}}), \mathsf{Inv}_{\mathsf{A}}(r_{\mathsf{A}})) \stackrel{c}{\approx} (\widehat{\mathsf{A}}(1^{\lambda}; r_{\widehat{\mathsf{A}}}), r_{\widehat{\mathsf{A}}})$$

holds. Oblivious samplability essentially says that the output y can be sampled without using any randomness except y itself. In more detail, the real-life execution $y \leftarrow \mathsf{A}(x; r_{\mathsf{A}})$ can be simulated by oblivious sampling $\widehat{y} \leftarrow \widehat{\mathsf{A}}(1^{\lambda}; r_{\widehat{\mathsf{A}}})$, where $r_{\widehat{\mathsf{A}}}$ does not contains confidential information that affect the security, i.e. it can be revealed to the adversary. Furthermore, we also need to explain to the adversary that as if

[1] This notion is also called "trapdoor simulatability" in [CDMW09]. "simulatability" in [DN00] is a simpler notion where Inv take y as input instead of trapdoor r_{A}. Actually, we only need simulatability rather than trapdoor simulatability in this thesis.

y is obliviously sampled by showing $r'_{\widehat{\mathsf{A}}} \leftarrow \mathsf{Inv}_{\mathsf{A}}(r_{\mathsf{A}})$ that satisfy $y = \widehat{\mathsf{A}}(1^\lambda; r'_{\widehat{\mathsf{A}}})$. Note that since $\widehat{\mathsf{A}}$ plays the role of Sim of the semantic security, oblivious samplability implies semantic security.

In this thesis oblivious samplability is defined for key encapsulation mechanism and chameleon encryption.

Non-Committing Security We define non-committing security or adaptive security in the following style: There exists a simulator algorithm Sim and an opening algorithm Open, such that for all x,

$$(\mathsf{A}(x; r_{\mathsf{A}}), r_{\mathsf{A}}) \stackrel{c}{\approx} (\mathsf{Sim}(1^\lambda; r_{\mathsf{Sim}}), \mathsf{Open}(r_{\mathsf{Sim}}, x))$$

holds. Non-committing security captures a situation where the randomness r_{A} is revealed to the adversary after adaptive corruption. In the ideal world, the simulator not only need to simulate the output $y' \leftarrow \mathsf{Sim}(1^\lambda; r_{\mathsf{Sim}})$, it need to explain y' is as if the outcome of real-life execution by opening a consistent randomness $r'_{\mathsf{A}} \leftarrow \mathsf{Open}(r_{\mathsf{Sim}}, x)$ that satisfy $y' = \mathsf{A}(x; r'_{\mathsf{A}})$. Note that non-committing security also implies semantic security.

In this thesis, non-committing-style security is defined for public-key encryption (thus, it is called non-committing encryption), wiretap codes, and hash function in hash encryption (thus, it is called chameleon hash in chameleon encryption).

From the above, we can see duality between the definitions of oblivious samplability and non-committing security. In oblivious samplability, Inv fakes the real-life execution as if it is ideal-world execution by producing $r'_{\widehat{\mathsf{A}}}$. In non-committing security, Open fakes the ideal-world execution as if it is real-life execution by producing r'_{A}. In general, ideal-world execution is less structured than real-life execution since it does not need to consider functional requirements such as correctness. Since it is easier to fake structured things as less structured things than their opposite, oblivious samplability seems easier to be achieved than non-committing security. Indeed natural instantiations of public-key encryption based on the DDH or LWE satisfy oblivious samplability, while we do not know a natural non-committing encryption scheme. Thus many studies including this thesis tackle the problem of constructing non-committing encryption from obliviously samplable cryptographic primitives. Although it might be meaningless in practice, we leave the opposite, constructing obliviously samplable encryption from non-committing encryption as future work.

2.2 Public-Key Encryption

We proceed to concrete definitions from here. Since non-committing encryption can be seen as public-key encryption that satisfies a non-committing security notion, we provide

the definition of public-key encryption and its static security notion known as semantic security for comparison.

Definition 2.1 (Public-Key Encryption)**.** A public-key encryption scheme consists of the following PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where Gen is the key-generation algorithm, Enc is the encryption algorithm, and Dec is the decryption algorithm. We explicitly display the randomness used in Gen and Enc as they are the target of attention in the study of non-committing encryption.

- $\mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$: Given the security parameter 1^λ, it outputs a public key pk and a secret key sk. The randomness used in this algorithm is denoted by r_{Gen} [2].
- $\mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)$: Given a public key pk and a plaintext $m \in \{0,1\}^\mu$, it outputs a ciphertext CT. The randomness used in this algorithm is denoted by r_{Enc}.
- $\mathsf{Dec}\left(sk, CT\right)$: Given a secret key sk and a ciphertext CT, it outputs m or $\perp$.

Public-Key/Ciphertext Expansion In this thesis, we measure the size of the public key and ciphertext by its asymptotic ratio to the length of the message. Public-key expansion and ciphertext expansion of a public-key encryption scheme is defined by $|pk|/|m|$ and $|CT|/|m|$, respectively for a enough long message $|m| = \mathsf{poly}(\lambda)$.

Remark 1. It is rare to focus on the above asymptotic measure of public-key expansion or ciphertext expansion if we only consider the static security of public-key encryption schemes. Since we can encrypt any polynomially long message by a single public key, its public-key expansion is trivially almost 0, or we can use the hybrid encryption technique with an efficient secret-key encryption scheme, hence its ciphertext expansion is trivially almost 1. However, in the context of non-committing encryption, we cannot reuse a single public key to encrypt multiple messages, nor does there exist an efficient *non-committing* secret-key encryption scheme. Essentially the most efficient non-committing secret-key encryption scheme is the one-time pad, so it is useless to compress the ciphertext by the hybrid encryption technique.

Definition 2.2 (Correctness)**.** The correctness of a public-key encryption scheme is defined as follows: For any message m, its ciphertext should be decrypted to the original message with overwhelming probability.

Formally, we say that a public-key encryption scheme $\mathtt{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is correct if for all message m,

$$\Pr\left[m \neq \mathsf{Dec}\left(sk, \mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)\right)\right] = \mathsf{negl}(\lambda)$$

holds, where $(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$ and the probability is taken over the choice of r_{Gen} and r_{Enc}.

[2] Since we can reproduce the secret key sk from the key-generation randomness r_{Gen}, we can use r_{Gen} as a secret key. In other words, we can understand the key-generation algorithm as $pk \leftarrow \mathsf{Gen}(1^\lambda; sk)$.

Security Informally, a public-key encryption scheme is semantically secure (statically secure) if the views of the static adversary in the real and ideal experiments are indistinguishable. In other words, the public key and ciphertext of the scheme can be simulated without knowledge of the message.

Definition 2.3 (Semantic Security)**.** For a public-key encryption scheme $\mathtt{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, consider the following PPT simulator Sim:

- $\mathsf{Sim}\left(1^\lambda; r_{\mathsf{Sim}}\right)$: Given the security parameter 1^λ, it outputs a simulated public key pk, a simulated ciphertext CT.

For a stateful PPT adversary $\mathcal{A}$, we define the following real and ideal experiments.

$\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{PKE},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{PKE},\mathcal{A}}$
$(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$	$(pk, CT) \leftarrow \mathsf{Sim}\left(1^\lambda; r_{\mathsf{Sim}}\right)$
$m \leftarrow \mathcal{A}\left(pk\right)$	$m \leftarrow \mathcal{A}\left(pk\right)$
$CT \leftarrow \mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)$	
$\mathsf{out} \leftarrow \mathcal{A}\left(CT\right)$	$\mathsf{out} \leftarrow \mathcal{A}\left(CT\right)$

We say that $\mathtt{PKE}$ is semantically secure if there exist a PPT simulator Sim such that for all PPT adversary $\mathcal{A}$,

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

holds.

Note that this definition is equivalent to the well-known indistinguishability against chosen plaintext attack (IND-CPA) security.

2.3 Non-Committing Encryption

Definition 2.4 (Non-Committing Encryption)**.** A non-committing encryption scheme is a public-key encryption scheme that satisfies the following non-committing security.

Informally, a non-committing encryption scheme is secure, or equivalently, a public-key encryption scheme satisfies non-committing security or adaptive security, if the views of the adaptive adversary upon the corruption of both sender and receiver are indistinguishable. In other words, not only the public key and ciphertext of the scheme can be simulated without knowledge of the plaintext, but also the randomness used in Gen and Enc can be simulated given the corrupted message.

Definition 2.5 (Non-Committing Security)**.** For a non-committing encryption scheme $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, consider the following PPT simulators $(\mathsf{Sim}, \mathsf{Open})$:

- $\mathsf{Sim}\left(1^\lambda; r_{\mathsf{Sim}}\right)$: Given the security parameter 1^λ, it outputs a simulated public key pk and a simulated ciphertext CT.

- $\mathsf{Open}(r_{\mathsf{Sim}}, m)$: Given the simulation randomness r_{Sim} and a message m, it outputs randomness for key generation r_{Gen} and encryption r_{Enc}.

For a stateful adversary $\mathcal{A}$, we define two experiments as follows.

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

We say that $\mathtt{NCE}$ is secure if there exist PPT simulators $(\mathsf{Sim}, \mathsf{Open})$ such that for all PPT adversary $\mathcal{A}$,

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

holds.

We also introduce a slightly weaker security notion of non-committing encryption, where the message to be encrypted is chosen independently of the public key. We use this style of security definition throughout this thesis because it is easier to define and prove such security notions, especially, in the sentence of the obliviously samplable chameleon encryption scheme constructed based on the LWE in Section 4.4.2.

Definition 2.6 (Non-Committing Security for pk-independent Messages)**.** We say that $\mathtt{NCE}$ is secure with respect to public-key independent message if there exist PPT simulators $(\mathsf{Sim}, \mathsf{Open})$ such that for all PPT adversary $\mathcal{A}$ and message m, for the following modified experiments

$\mathsf{Exp}^{\mathrm{Real}'}_{\mathtt{NCE},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Ideal}'}_{\mathtt{NCE},\mathcal{A}}$
$(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$	$(pk, CT) \leftarrow \mathsf{Sim}\left(1^\lambda; r_{\mathsf{Sim}}\right)$
$CT \leftarrow \mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)$	$(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{Sim}}, m)$
$\mathsf{out} \leftarrow \mathcal{A}\left(CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}}\right)$	$\mathsf{out} \leftarrow \mathcal{A}\left(CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}}\right)$

, the advantage $\mathsf{Adv}_{\mathtt{NCE},\mathcal{A}}\left(\lambda\right)$ is negligible.

Note that a non-committing encryption scheme $\mathtt{NCE}'$ that is secure with respect to a public-key independent message can be easily converted to $\mathtt{NCE}$ that satisfies the non-committing security (Definition 2.5) via the hybrid encryption technique with one-time pad, similarly in the sentence of semantically secure public-key encryption. In our NCE schemes, the amplification presented in section 3.4 contains the one-time pad.

Limitations of NCE schemes Due to its security requirement, non-committing encryption schemes have inherent limitations that standard public-key encryption schemes do not have.

Lemma 2.1. Let NCE be a non-committing encryption scheme for message m. Then its secret key sk, key-generation randomness r_{Gen} and encryption randomness r_{Enc} must be larger than the message length, i.e., $|sk|, |r_{\mathsf{Gen}}|, |r_{\mathsf{Enc}}| \geq |m|$.

This is because of the ability of NCE that it can open the simulated ciphertext to an arbitrary message. In detail, for any message, there exists a secret key that can decrypt the ciphertext to the message, thus the space of secret key must be larger than the message space. Similarly, there exists encryption randomness that can encrypt the message to the ciphertext, thus space of encryption randomness must be larger than the message space.

Lemma 2.2. Non-committing encryption scheme in the standard model, which does not use the random oracle nor common reference strings, cannot achieve the perfect correctness, i.e., $\Pr[m \neq \mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r_{\mathsf{Enc}}))] \neq 0$.

Consider a simulated ciphertext is opened in two ways. Then there should exist two pairs of the secret key and encryption randomness that satisfy

$$\mathsf{Enc}(pk, m_0; r_{\mathsf{Enc}0}) = CT, \quad \mathsf{Dec}(sk_0, CT) = m_0,$$
$$\mathsf{Enc}(pk, m_1; r_{\mathsf{Enc}1}) = CT, \quad \mathsf{Dec}(sk_1, CT) = m_1.$$

This causes the case where during an honest execution of the non-committing encryption scheme, the receiver generates key pair (pk, sk_0), and the sender encrypts message m_1 with encryption randomness $r_{\mathsf{Enc}1}$. In this case, the decryption fails as

$$\mathsf{Dec}(sk_0, \mathsf{Enc}(pk, m_1; r_{\mathsf{Enc}1})) = m_0.$$

Chapter 3

NCE with $\mathcal{O}(\lambda)$ Ciphertext-Expansion

3.1 Overview

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

As a starting point, we review the three-round NCE protocol proposed by Beaver [Bea97], which contains a fundamental idea to build NCE from the DDH problem. Next, we observe that the Beaver's protocol can be transformed to a two-round NCE scheme. Although the resulting scheme is not fully secure NCE, we can regard it as a weak variant of NCE which we call weak NCE. Then we show a simple idea to transform a weak NCE scheme to a secure NCE, whose ciphertext expansion is $\mathcal{O}(\lambda)$.

3.1.1 Starting Point: Beaver's Protocol

Beaver's NCE protocol essentially executes two Diffie-Hellman key exchange protocols in parallel. This protocol can send a 1-bit message. Ciphertext expansion of this protocol is $\mathcal{O}(\lambda)$. We describe the protocol below and in Figure 3.1.

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

Step2. The receiver picks a random bit $x \leftarrow \{0,1\}$ and an exponent $\rho_x \leftarrow \mathbb{Z}_p$, and then sets $B_x = g^{\rho_x}$. The receiver also obliviously generates $B_{1-x} \leftarrow \mathbb{G}$. The receiver computes $K_x = {A_x}^{\rho_x}$ and obliviously samples $K_{1-x} \leftarrow \mathbb{G}$. The receiver sends $((B_0, B_1), (K_0, K_1))$ to the sender. The random coin used in this step is $(x, \rho_x, B_{1-x}, K_{1-x})$.

Sender		Receiver
Input: $m \in \{0,1\}$		
$z \leftarrow \{0,1\}$		$x \leftarrow \{0,1\}$
$\alpha_z \leftarrow \mathbb{Z}_p$		$\rho_x \leftarrow \mathbb{Z}_p$
$A_z = g^{\alpha_z}$	(A_0, A_1)	$B_x = g^{\rho_x}$
$A_{1-z} \leftarrow \mathbb{G}$	$\longrightarrow$	$B_{1-x} \leftarrow \mathbb{G}$
	$(K_0, K_1), (B_0, B_1)$	$K_x = {A_x}^{\rho_x}$
	$\longleftarrow$	$K_{1-x} \leftarrow \mathbb{G}$
if ${B_z}^{\alpha_z} = K_z$	w	
$w \coloneqq z \oplus m$	$\longrightarrow$	if $w \neq \perp$
else $w \coloneqq \perp$		Output: $m = w \oplus x$

Figure 3.1: The description of Beaver's protocol [Bea97].

Step3. The sender checks whether $x = z$ holds or not, by checking if ${B_z}^{\alpha_z} = K_z$ holds. With overwhelming probability, this equation holds if and only if $x = z$. If $x = z$, the sender sends $w \coloneqq z \oplus m$, otherwise quits the protocol.

Step4. The receiver recovers the message by computing $w \oplus x$.

Next, we describe the simulator for this protocol.

Simulator The simulator simulates a transcript (A_0, A_1), $((B_0, B_1), (K_0, K_1))$, and w as follows. It generates $\alpha_0, \alpha_1, \rho_0, \rho_1 \leftarrow \mathbb{Z}_p$ and sets

$$((A_0, A_1), (B_0, B_1), (K_0, K_1)) = ((g^{\alpha_0}, g^{\alpha_1}), (g^{\rho_0}, g^{\rho_1}), (g^{\alpha_0\rho_0}, g^{\alpha_1\rho_1})).$$

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

The simulator can later open this transcript to either message 0 or 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 $x = z = w$, and outputs (w, α_w, A_{1-w}) and $(w, \rho_w, B_{1-w}, K_{1-w})$ 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 K_{1-x} is generated as a random group element in the real protocol, but it is generated as ${A_{1-x}}^{\rho_{1-x}} = g^{\alpha_{1-x}\rho_{1-x}}$ in the simulation. If the real protocol proceeds to Step.4, we have $x = z$ 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-x} and B_{1-x}, that is, α_{1-x} and ρ_{1-x}. Thus, under the DDH assumption, $\mathcal{A}$ cannot distinguish randomly generated $K_{1-x} \leftarrow \mathbb{G}$ from ${A_{1-x}}^{\rho_{1-x}} = g^{\alpha_{1-x}\rho_{1-x}}$. Thus, this protocol is a secure NCE protocol.

This protocol succeeds in transmitting a message only when $z = x$, and otherwise it fails. Note that even when $z \neq x$, the protocol can transmit the message correctly because in Step.3, the sender knows the receiver's secret x. However, in this case, we cannot construct a successful simulator. In order to proof the security based on the DDH assumption, we have to ensure that either one of exponent pair (α_0, ρ_0) or (α_1, ρ_1) is not revealed to the adversary. However, when $z \neq x$, the exponents related to both 0 and 1 is corrupted, hence we cannot use the DDH assumption, and the security proof fails.

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

3.1.2 Extension to Two-Round Weak 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 $CT = ((B_0, B_1), (K_0, K_1), w)$, where (A_0, A_1), (B_0, B_1), (K_0, K_1), and $w := x \oplus m$ 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 $B_z^{\rho_z} = K_z$ holds or not, and then computes $w \oplus x$.

Of course, $\mathtt{NCE}^1_{\mathtt{lin}}$ is not a secure NCE scheme in the sense that we cannot construct a successful simulator when $z \neq x$ for the same reason explained 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. Based on this observation, we construct an NCE scheme $\mathtt{NCE}_{\mathtt{lin}}$ as follows. We also describe $\mathtt{NCE}_{\mathtt{lin}}$ in Figure 3.2.

Let the length of messages be $\mu = \mathcal{O}(SP)$ and $n = \mathcal{O}(\mu)$. We later specify the concrete relation of μ and n. The receiver first generates $z_1 \cdots z_n = z \leftarrow \{0,1\}^n$. 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, $A_{i,z_i} = g^{\rho_i}$. 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 $(z, \rho_1, \ldots, \rho_n)$.

When encrypting a message m, the sender first generates $x_1 \cdots x_n = x \leftarrow \{0,1\}^n$. Then, for every $i \in [n]$, the sender generates $((B_{i,0}, B_{i,1}), (K_{i,0}, K_{i,1}))$ in the same way as $\mathtt{NCE}^1_{\mathtt{lin}}$, where we "encapsulate" 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.

Receiver		Sender
		Input: $m \in \{0,1\}^\mu$
$z \leftarrow \{0,1\}^n$		$x \leftarrow \{0,1\}^n$
$\forall i \in [n], \alpha_i \leftarrow \mathbb{Z}_p$		$\forall i \in [n], \rho_i \leftarrow \mathbb{Z}_p$
$A_{i,z_i} = g^{\alpha_i}$		$B_{i,x_i} = g^{\rho_i}$
$A_{i,1-z_i} \leftarrow \mathbb{G}$	$\begin{pmatrix} A_{1,0},\dots,A_{n,0} \\ A_{1,1},\dots,A_{n,1} \end{pmatrix} \longrightarrow$	$B_{i,1-x_i} \leftarrow \mathbb{G}$
		$K_{i,x_i} = {A_{i,x_i}}^{\rho_i}$
		$K_{i,1-x_i} \leftarrow \mathbb{G}$
if ${B_{i,z_i}}^{\alpha_i} = K_{i,z_i}$	$\begin{pmatrix} B_{1,0},\dots,B_{n,0} \\ B_{1,1},\dots,B_{n,1} \end{pmatrix}, \begin{pmatrix} K_{1,0},\dots,K_{n,0} \\ K_{1,1},\dots,K_{n,1} \end{pmatrix}, w \longleftarrow$	$w = H(x) \oplus m$
$x_i \coloneqq z_i$		
else $x_i \coloneqq 1 - z_i$		
Output: $m = w \oplus H(x)$		

Figure 3.2: The description of $\mathtt{NCE}_{\mathtt{lin}}$.

The resulting ciphertext is

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

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

The simulator for this scheme runs as follows. It first generates $z_1 \cdots z_n = z \leftarrow \{0,1\}^n$ and $x_1 \cdots x_n = x \leftarrow \{0,1\}^n$. Then, for every index $i \in [n]$ such that $z_i = x_i$, 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 $z_i \neq x_i$, 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 $w \leftarrow \{0,1\}^\mu$.

Although the ciphertext generated by the simulator is not "fully non-committing" about x, still, it loses the information of bits of x such that $x_i = z_i$. 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 $n > 2\mu$ 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. 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. [CDMW09].

3.1.3 Compress Ciphertext

We compress the ciphertext of above scheme in two way. The first part of the ciphertext $\begin{pmatrix} B_{1,0}, \ldots, B_{n,0} \\ B_{1,1}, \ldots, B_{n,1} \end{pmatrix}$ is compressed to a single group element by using chameleon encryption. We will explain this compression in the next chapter.

In this section we focus on the compression of the second part of the ciphertext $\begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}$. Recall that each $K_{i,b}$ is a group element of size λ. However its property as a group element is not used in the scheme as only in the decryption algorithm, $K_{i,b}$ is compared with ${B_{i,z_i}}^{\alpha_i}$. We replace this comparison of group elements to the comparison of hash value of them, i.e., $\mathsf{H}_{\mathbb{G}}(K_{i,z_i})$ vs. $\mathsf{H}_{\mathbb{G}}({B_{i,z_i}}^{\alpha_i})$. Henceforth $K_{i,b}$ denotes the hash value of length ℓ, which we can choose arbitrary.

We want ℓ to be as small as possible as it directly reflects to the ciphertext size. However if we choose too much small ℓ, the probability of $K_{i,1-z_i} = \mathsf{H}_{\mathbb{G}}({B_{i,z_i}}^{\alpha_i})$, hence the probability of decryption error becomes large. We can fix this decryption error using error correcting codes, but as the error probability increases, the rate of the codes decreases, resulting in large ciphertext. Therefor we must choose appropriate constant ℓ that balances ciphertext size and error probability. We also choose appropriate wiretap codes, whose rate is as high as possible, to amplify the two problem of this weak NCE scheme: weak security due to its construction, and weak correctness caused by compressing ciphertext.

3.1.4 Related Works on Amplification for Public-Key Encryption

Studies on security amplification have asked and answered the question:

> "How far can we weaken a security definition so that schemes satisfying the definition can still be transformed into those satisfying full-fledged security?"

Dwork, Naor, and Reingold [DNR04] first studied the amplification of public-key encryption. They showed that a public-key encryption scheme that satisfies weak forms of one-wayness and correctness can be transformed into one satisfies the ordinary correctness and IND-CPA security. Holenstein and Renner [HR05] showed a more efficient amplification method, starting from a scheme satisfying weak forms of IND-CPA security and correctness. Lin and Tessaro [LT13] provided an amplification method for schemes with IND-CCA security.

In this work, we show an amplification method for NCE, which can be seen as one of this line of research.

3.2 Wiretap Channel and Amplification

In this section we introduce the wiretap channel model, wiretap codes, and its instantiation.

3.2.1 Channel Model

When a sender transmits a message $x \in \{0,1\}^n$ through a channel ChR, the receiver gets a noisy version of the message $\tilde{x} \in \{0,1,\perp\}^n$. We define the procedure of such channels as probabilistic functions, $\tilde{x} \leftarrow \mathsf{ChR}(x; r_{\mathsf{ch}})$. We review two channel models, Binary Erasure Channel (BEC) and Binary Symmetric Channel (BSC).

Let $\mathcal{B}_p^n$ be the n-bit Bernoulli distribution with parameter p. In other words, $r_{\mathsf{ch}} \leftarrow \mathcal{B}_p^n$ is an n-bit string where for each $i \in [n]$, $\Pr[r_{\mathsf{ch}i} = 1] = p$ and $\Pr[r_{\mathsf{ch}i} = 0] = 1 - p$.

Definition 3.1 (Binary Erasure Channel (BEC))**.** Through a binary erasure channel BEC_p, each bit of input $x \in \{0,1\}^n$ is erased with probability p.

$\mathsf{BEC}_p(x; r_{\mathsf{ch}})$ samples randomness $r_{\mathsf{ch}} \leftarrow \mathcal{B}_p^n$. Output of the channel is $\tilde{x}$ where $\tilde{x}_i = \perp$ if $r_{\mathsf{ch}i} = 1$ and $\tilde{x}_i = x_i$ if $r_{\mathsf{ch}i} = 0$.

We also denote the output of BEC by $x_{\mathcal{I}} \leftarrow \mathsf{BEC}_p(x; r_{\mathsf{ch}})$ where $\mathcal{I} = \{i \in [n] \mid r_{\mathsf{ch}i} = 0\}$ is the set of non-erased indices.

Definition 3.2 (Binary Symmetric Channel (BSC))**.** Through a binary symmetric channel BSC_p, each bit of input $x \in \{0,1\}^n$ is flipped with probability p.

BSC_p samples randomness $r_{\mathsf{ch}} \leftarrow \mathcal{B}_p^n$. Output of the channel is $\tilde{x} = x \oplus r_{\mathsf{ch}}$.

We denote by $\mathsf{BEC}_{\leq p}$, a binary symmetric channel with parameter $p' \leq p$.

3.2.2 Wiretap Codes

When weak NCE is used to communicate, roughly speaking, the receiver gets a noisy version of the transmitted message x, and the adversary can see some partial information of x. In fact, such a situation is very natural and studied as physical layer security in the Information and Coding (I&C) community since the wiretap channel model was proposed by Wyner [Wyn75]. Based on this observation, in this section, we show how to amplify a weak NCE scheme into a full-fledged one by using *wiretap codes*. [1]

Wiretap Codes As described in Figure 3.3, when the sender transmits a message x over the wiretap channel, on one hand, the receiver gets the message affected by noise over receiver channel $\mathsf{ChR}(x)$. On the other hand, an adversary can interrupt the transmission and gets a noisier version of the message $\mathsf{ChA}(x)$.

[1] In literature, wiretap codes sometimes appeared in the name of "encryption" or "one-way secret-key agreement". It can be also interpreted as a kind of secret sharing scheme.

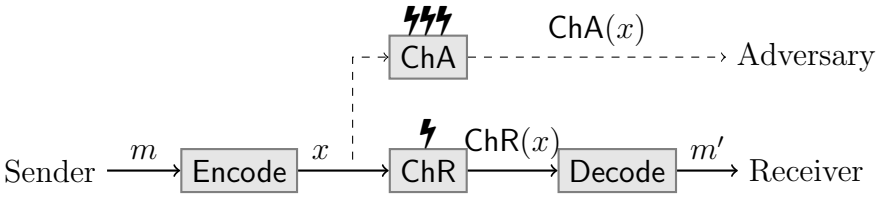

Figure 3.3: Wiretap channel model.

In such a model, using the difference in the amount of noise the receiver and the adversary are affected, wiretap codes $\mathtt{WC}$ enable us to transmit a message m correctly to the receiver while keeping it information-theoretically secure against the adversary.

Wiretap codes have an encoding and a decoding algorithm similar to error-correcting codes. Wiretap codes satisfy two properties. One is correctness, which ensures that the receiver can decode codewords even if they are affected by some amount of noise. The other is security, which guarantees that the adversary can get no information about the message given some part of the codeword. It is known that the encoding algorithm must use randomness to satisfy security.

Originally in the I&C community, the security of wiretap codes was defined by mutual information. Bellare et al. [BTV12b, BT12, BTV12a] proposed several equivalent definitions in a cryptographic manner. Among them, we recall one adopting the distinguishing style of security below. Then we proposed a new security property, *conditional invertibility* for wiretap codes, which we need in the security proof of our amplification for NCE.

Note that the following definition adopts the seeded version of wiretap codes also proposed by Bellare et al. [BTV12b]. In the seeded wiretap channel, the sender, receiver, and an adversary can see a public random seed. We adopt the seeded wiretap codes to give a simple construction of the codes. The seed can be removed without increasing the rate of the codes by a transformation shown in [BT12]. In this work, we put the seed into a part of the public key when constructing NCE.

Definition 3.3 (Wiretap Codes)**.** (Seeded) wiretap codes $\mathtt{WC}$ consist of the following PPT algorithms $(\mathsf{WC.Setup}, \mathsf{WC.Encode}, \mathsf{WC.Decode})$.

- $\mathsf{WC.Setup}(1^\lambda)$: Given the security parameter 1^λ, it samples a public seed p.
- $\mathsf{WC.Encode}(p, m; s)$: It encodes a message $m \in \{0,1\}^\mu$ with a public seed p and randomness $s \leftarrow \mathbf{S}$, and outputs a codeword $x \in \{0,1\}^n$.
- $\mathsf{WC.Decode}(p, x)$: On input a noisy codeword $x \in \{0,1\}^n$ and a public seed p, it outputs a message m.

Rate of Wiretap Codes. The rate of $\mathtt{WC}$ is the length of messages over the length of codewords $\mu/n \in (0,1)$. The rate of $\mathtt{WC}$ is at most the secrecy capacity of the wiretap channel. The secrecy capacity of wiretap channel, defined with symmetric channels ChR

and ChA, is equal to $H(U|\mathsf{ChA}(U)) - H(U|\mathsf{ChR}(U))$ for a uniformly random bit U [Leu77], where $H(Y|X)$ denotes the conditional entropy.

Usually, wiretap codes are required to satisfy the following correctness and security.

As a security property, we present a definition of distinguishing security adopted for seeded wiretap codes. This is a natural extension of the distinguishing security for seedless wiretap codes proposed by Bellare et al. [BTV12b].

Correctness: $\mathtt{WC}$ is correct over the receiver's channel ChR if for all message $m \in \{0,1\}^\mu$ and public seed p, we have

$$\Pr[\mathsf{WC.Decode}(p, \mathsf{ChR}(\mathsf{WC.Encode}(p, m))) \neq m] = \mathsf{negl}(\lambda) \quad .$$

Note that correctness holds when the message length $|m| = \mu$ is enough large, concretely, $|m| = \Omega(\lambda)$ is enough.

Security: $\mathtt{WC}$ is DS-secure against adversary's channel ChA if for any unbounded stateful adversary $\mathcal{A}$, we have

$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{l} p \leftarrow \mathsf{WC.Setup}(1^\lambda), (m_0, m_1) = \mathcal{A}(p), \\ b \leftarrow \{0,1\}, x \leftarrow \mathsf{WC.Encode}(p, m_b), \\ \tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}}), \\ b' = \mathcal{A}(\tilde{x}) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda) \quad .$$

Next, we introduce a new security property for wiretap codes, *conditional invertibility*.

Intuitively, this security notion states that after the adversary sees the partial information $\tilde{x} \leftarrow \mathsf{ChA}(x)$ resulted from the codeword x of a message m', we can efficiently explain that $\tilde{x}$ has resulted from another message m. The security definition involves a PPT inversion algorithm $\mathsf{WC.Invert}$, which on inputs seed p, a condition $\tilde{x}$, and a message m, outputs randomness s' and $r_{\mathsf{ch}}{}'$ such that $\mathsf{ChA}(\mathsf{WC.Encode}(p, m; s'); r_{\mathsf{ch}}{}')$ is equal to the condition $\tilde{x}$.

Conditional invertibility implies the ordinary distinguishing security. It can be seen as non-committing security for wiretap codes. Note that wiretap codes are inherently non-committing in the sense that they usually required to statistically lose the information of messages. Thus, the only point conditional invertibility additionally requires is that the inversion can be computed efficiently.

Definition 3.4 (Conditional Invertibility)**.** For an unbounded stateful adversary $\mathcal{A}$ and a PPT algorithm $\mathsf{WC.Invert}$, define two experiments as follows:

$\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{WC},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{WC},\mathcal{A}}$
$p \leftarrow \mathsf{WC.Setup}(1^\lambda)$	$p \leftarrow \mathsf{WC.Setup}(1^\lambda)$
$(m, m') = \mathcal{A}(p)$	$(m, m') = \mathcal{A}(p)$
$x \leftarrow \mathsf{WC.Encode}(p, m; s)$	$x \leftarrow \mathsf{WC.Encode}(p, m'; s)$
$\tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}})$	$\tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}})$
	$(s', r_{\mathsf{ch}}{}') \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m)$
$\mathsf{out} = \mathcal{A}(\tilde{x}, s, r_{\mathsf{ch}})$	$\mathsf{out} = \mathcal{A}(\tilde{x}, s', r_{\mathsf{ch}}{}')$

We say that $\mathtt{WC}$ is invertible conditioned on ChA if there exists a PPT inverter $\mathsf{WC.Invert}$ such that for any unbounded adversary $\mathcal{A}$,

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

holds.

3.2.3 Instantiation of Wiretap Codes

Overview. We recall a modular construction of wiretap codes proposed by Bellare et al. [BTV12b] called Invert-then-Encode construction. The building blocks are error-correcting codes and invertible extractors. This idea of composing error-correcting codes and extractors can be found also in the construction of a linear secret sharing scheme proposed by Cramer et al. [CDD+15].

Consider an seeded extractor $\mathsf{Ext} : \{0,1\}^k \to \{0,1\}^\mu$ which on inputs $X \in \{0,1\}^k$ and a seed p, outputs $m \in \{0,1\}^\mu$. The extractor is *invertible* if there is an efficient inverter Inv, which on inputs $m \in \{0,1\}^\mu$ and seed p, samples a preimage $X \in \{0,1\}^k$ using randomness s. The Invert-then-Encode construction takes input m with seed p, first inverts the extractor $X \leftarrow \mathsf{Inv}(m, p; s)$, then encodes X by the error-correcting code as $x = \mathsf{Encode}(X)$.

For a concrete instantiation, Bellare et al. suggested to use the polar codes [Ari09] as error-correcting codes to achieve the optimal rate. Note that we can compute the encoding of input m by mG where G is a generator matrix of the linear error-correcting code. Invertible extractors can be instantiated using multiplication over $\mathrm{GF}(2^k)$. Concretely, the extractor takes inputs $x \in \{0,1\}^k$ and seed $p \in \mathrm{GF}(2^k)$, and outputs the first μ bit of $x \odot p$, where $\odot$ denotes multiplication over $\mathrm{GF}(2^k)$. The inverter Inv for this extractor is obtained by $\mathsf{Inv}(m, p; s) = (m \| s) \odot p^{-1}$.

Construction. We describe the construction of wiretap codes for $\mu = \mathcal{O}(\lambda)$ bit messages. For a longer message, we can encode it by first dividing it into blocks of μ bit and then encoding each block by the following codes (see [BT12]).

Let $\mu, k, n = \mathcal{O}(\lambda)$. Let $G \in \{0,1\}^{k \times n}$ be a generator matrix of a linear error-correcting code, and $\mathsf{ECC.Decode}$ a corresponding decoding algorithm. Choose a constant $\epsilon > 0$ such that the error-correcting code can be correct over $\mathsf{ChR} = \mathsf{BSC}_{\le\epsilon}$. We construct wiretap codes which is correct over $\mathsf{ChR} = \mathsf{BSC}_{\le\epsilon}$ and invertible conditioned on $\mathsf{ChA} = \mathsf{BEC}_{0.5}$. Thus, in this construction, the wiretap decoding algorithm takes as input $x' \leftarrow \mathsf{BSC}_\epsilon(x)$, and the wiretap inverter algorithm takes as input $x_{\mathcal{I}} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}})$ where $\mathcal{I} \in [n]$ is the set of non-erased indices determined by a uniformly random n-bit string r_{ch}.

- $\mathsf{WC.Setup}(1^\lambda)$: Sample and output $p \leftarrow \mathrm{GF}(2^k) \setminus \{0\}$.
- $\mathsf{WC.Encode}(p, m; s)$: For input $m \in \{0,1\}^\mu$, sample $s \leftarrow \{0,1\}^{k-\mu}$, output $x = ((m \| s) \odot p)G \in \{0,1\}^n$.

- $\mathsf{WC.Decode}(p, x')$: Output the first μ bits of $\mathsf{ECC.Decode}(x') \odot p^{-1}$.
- $\mathsf{WC.Invert}(p, x_{\mathcal{I}}, m)$: On input a condition $x_{\mathcal{I}} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}})$, sample and output s' which satisfies $x_{\mathcal{I}} = ((m \| s') \odot p)G_{\mathcal{I}}$.

  Concretely, let $\sum_i z_i c_i + c_0$ $(c_i \in \{0,1\}^k, z_i \in \{0,1\})$ be the general solution of linear equation $x_{\mathcal{I}} = yG_{\mathcal{I}}$. Then, uniformly sample a solution $\{z_i\}_i$ of linear equation $m = \sum_i z_i (c_i \odot p^{-1})_{\{1,\ldots,\mu\}} + (c_0 \odot p^{-1})_{\{1,\ldots,\mu\}}$. Finally, output $s' = \sum_i z_i (c_i \odot p^{-1})_{\{\mu+1,\ldots,k\}} + (c_0 \odot p^{-1})_{\{\mu+1,\ldots,k\}}$.

  It also outputs randomness for the channel $r_{\mathsf{ch}}{}' = r_{\mathsf{ch}}$, which is a uniformly random n-bit string representing the non-erased indices $\mathcal{I}$.

Rate of the Scheme. The rate μ/n of the scheme can be set to a constant smaller than $(\frac{k}{n} - \frac{1}{2})$. If the rate k/n of the error-correcting codes is close to its capacity $1 - h_2(\epsilon)$, the rate of $\mathtt{WC}$ can be close to its secrecy capacity $1/2 - h_2(\epsilon)$, which is the optimal rate of wiretap codes.

Correctness. The correctness of the wiretap codes directly follows from the correctness of the underlying error-correcting codes.

Conditional Invertibility. To show the invertibility conditioned on $\mathsf{BEC}_{0.5}$, we need to show that distributions of $(\tilde{x}, s, r_{\mathsf{ch}})$ are statistically indistinguishable in the real and ideal experiments of the definition. We introduce the hybrid experiment defined as follows:

$\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{WC},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Hybrid}}_{\mathtt{WC},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{WC},\mathcal{A}}$
$p \leftarrow \mathsf{WC.Setup}(1^\lambda)$	$p \leftarrow \mathsf{WC.Setup}(1^\lambda)$	$p \leftarrow \mathsf{WC.Setup}(1^\lambda)$
$(m, m') = \mathcal{A}(p)$	$(m, m') = \mathcal{A}(p)$	$(m, m') = \mathcal{A}(p)$
$x \leftarrow \mathsf{WC.Encode}(p, m; s)$	$x \leftarrow \mathsf{WC.Encode}(p, m; s')$	$x \leftarrow \mathsf{WC.Encode}(p, m'; s)$
$\tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}})$	$\tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}})$	$\tilde{x} \leftarrow \mathsf{ChA}(x; r_{\mathsf{ch}})$
	$(s', r_{\mathsf{ch}}{}') \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m)$	$(s', r_{\mathsf{ch}}{}') \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m)$
$\mathsf{out} = \mathcal{A}(\tilde{x}, s, r_{\mathsf{ch}})$	$\mathsf{out} = \mathcal{A}(\tilde{x}, s', r_{\mathsf{ch}}{}')$	$\mathsf{out} = \mathcal{A}(\tilde{x}, s', r_{\mathsf{ch}}{}')$

Before the proof we recall useful lemma, the Chernoff bound and the leftover hash lemma.

Lemma 3.1 (Chernoff Bound)**.** Let X be a binomial random variable. 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.

Lemma 3.2 (Leftover Hash Lemma)**.** Let $\mathcal{H} := \{h : \{0,1\}^n \to \{0,1\}^\ell\}$ be a universal hash family. If $\ell \leq \mathbf{H}_\infty(x) - \omega(\log \lambda)$, $(h, h(x))$ and (h, u) are statistically indistinguishable where $u \leftarrow \{0,1\}^\ell$.

Claim 3.3. The distribution of output in the real and hybrid experiments are same.

Proof. In general, for a function $f : \mathcal{X} \to \mathcal{Y}$,

$$\{(x, y) \mid x \leftarrow \mathcal{X}, y = f(x)\} \equiv \left\{(x', y) \mid x \leftarrow \mathcal{X}, y = f(x), x' \leftarrow f^{-1}(y)\right\}$$

holds, where $f^{-1}(y)$ denotes the set of preimages of y.

By applying the above fact to $f_{p,m}(s, r_{\mathsf{ch}}) = \mathsf{ChA}(\mathsf{WC.Encode}(p, m; s); r_{\mathsf{ch}})$, what we need to show is that $\mathsf{WC.Invert}$ implements sampling $(s', {r_{\mathsf{ch}}}') \leftarrow f_{p,m}^{-1}(\tilde{x})$.

Since we consider $\mathsf{ChA} = \mathsf{BEC}_{0.5}$, $\mathsf{WC.Invert}$ can uniquely determine ${r_{\mathsf{ch}}}' = r_{\mathsf{ch}}$ from the representation of $\tilde{x} = x_{\mathcal{I}}$. Recall that $\mathsf{WC.Invert}$ samples s' satisfying $x_{\mathcal{I}} = ((m \| s') \odot p)G_{\mathcal{I}} = \mathsf{BEC}_{0.5}(\mathsf{WC.Encode}(p, m; s'); r_{\mathsf{ch}})$ uniformly at random. Hence, the claim follows. □

Claim 3.4. The hybrid and ideal experiments are statistically close if the wiretap codes are secure in the ordinarily sense.

Proof. Consider the adversary $\mathcal{A}$ that distinguished the two experiments. We can construct another adversary $\mathcal{A}'$ against the security of the wiretap codes as follows: Given p, run $\mathcal{A}'$ on p and obtain m, m'; send them to its challenger and receive $\tilde{x}$; compute $(s, r_{\mathsf{ch}}) \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m)$; run $\mathcal{A}'$ on $\tilde{x}, s, r_{\mathsf{ch}}$ and receive out; output out. The claim is proven, since the simulation by $\mathcal{A}$ is perfect. □

Claim 3.5. The wiretap codes are secure in the ordinarily sense.

Bellare et al. [BTV12b] show a detailed security proof of the wiretap codes for general ChA. Below, we show a specific security proof for $\mathsf{ChA} = \mathsf{BEC}_{0.5}$.

Proof. Recall that the parameter is selected to satisfy $\mu/n < (k/n - 1/2)$. Let $2\delta := ((k - \mu)/n - 1/2) > 0$ be a constant.

Since $\mathsf{ChA} = \mathsf{BEC}_{0.5}$, the input for the adversary is $x_{\mathcal{I}} = ((m \| s) \odot p)G_{\mathcal{I}}$. By the Chernoff bound, $|\mathcal{I}| < (\frac{1}{2} + \delta)n$ holds except negligible probability.

Let us decompose the submatrix of the generator $G_{\mathcal{I}} = PDQ$, where $P \in \{0,1\}^{k \times k}$ and $Q \in \{0,1\}^{|\mathcal{I}| \times |\mathcal{I}|}$ are invertible. Furthermore $D = (d_{i,j}) \in \{0,1\}^{k \times |\mathcal{I}|}$ satisfies $d_{i,i} = 1$ for $1 \leq i \leq r := \mathrm{Rank}(G_{\mathcal{I}})$ and $d_{i,j} = 0$ for other elements. We interpret the multiplication by D as getting the first r bits and concatenating $0^{|\mathcal{I}|-r}$. Thus $x_{\mathcal{I}} = ((((m \| s) \odot p)P)_{[r]} \| 0^{|\mathcal{I}|-r})Q$.

For input $m \| s$ and seed p, $h_p(m \| s) := ((m \| s \odot p)P)_{[r]}$ forms a universal hash family. Note that the input has min-entropy $\mathbf{H}_\infty(m \| s) = k - \mu$.

Since $r \leq |\mathcal{I}| \leq (\frac{1}{2} + \delta)n \leq k - \mu - \delta n < \mathbf{H}_\infty(m \| s) - \omega(\log \lambda)$ holds, by the left over hash lemma, $(p, h_p(m \| s))$ is statistically indistinguishable from (p, u) where $u \leftarrow \{0,1\}^r$. Therefore $x_{\mathcal{I}}$ is statistically indistinguishable from $(u \| 0^{|\mathcal{I}|-r})Q$, which is independent of m. Thus, the claim is proven. □

By combining the above three claims, conditional invertibility of the wiretap codes follows.

3.3 Weak Non-Committing Encryption

In this section, we formalize the weak correctness and weak security for a non-committing encryption scheme. Then, we show a simple construction of weak non-committing encryption scheme that has $\mathcal{O}(\lambda)$ public-key and ciphertext expansion.

3.3.1 Definition of Weak Non-Committing Encryption

Informally, we say that a public-key encryption scheme is weakly correct if it has decryption error for each message bit.

Definition 3.5 (Weak Correctness)**.** We say that an NCE scheme $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is weakly correct if it has non-negligible decryption error for each message bit. Specifically, we say that $\mathtt{NCE}$ has ϵ-decryption error if for all message $m \in \{0,1\}^\mu$ and index $i \in [\mu]$,

$$\Pr\left[m_i \neq \mathsf{Dec}\left(sk, \mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)\right)_i\right] \leq \epsilon$$

holds, where $(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$ and the probability is taken over the choice of r_{Gen} and r_{Enc}.

Note that the procedure of encryption and decryption of $\mathtt{NCE}$ works as the binary symmetric channel

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, \cdot\ )) = \mathsf{BSC}_{\leq\epsilon}(\cdot).$$

Furthermore, we say that $\mathtt{NCE}$ satisfies correctness If $\epsilon = \mathsf{negl}(\lambda)$, the weak correctness corresponds to the correctness, Definition 2.2.

Definition 3.6 (Weak Non-Committing Security)**.** Weak security allows an adversary to learn some partial information of a plaintext $\mathsf{Leak}(m)$. Still, it guarantees that other information of m remains hidden.

For an NCE scheme $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a probabilistic function Leak, consider the following PPT simulators $(\mathsf{SimGen}, \mathsf{SimEnc}, \mathsf{Open})$:

- $\mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$: Given the security parameter 1^λ, it outputs a simulated public key pk.
- $\mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{m} \leftarrow \mathsf{Leak}(m; r); r_{\mathsf{SimEnc}})$: Given the simulation randomness r_{SimGen} and a partial information of a plaintext $\tilde{m}$ which is computed by the probabilistic function Leak with randomness r, it outputs a simulated ciphertext CT.
- $\mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, m, r)$: Given the simulation randomness r_{SimGen}, r_{SimEnc}, a message m and the randomness r used by Leak, it outputs randomness for key generation r_{Gen} and encryption r_{Enc}.

For an adversary $\mathcal{A}$ and a message m, define two experiments as follows.

$\mathsf{Exp}^{\text{Weak Real}}_{\mathtt{NCE},\mathcal{A}}$	$\mathsf{Exp}^{\text{Weak Ideal}}_{\mathtt{NCE},\mathcal{A}}$
$(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$	$pk \leftarrow \mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$
$CT \leftarrow \mathsf{Enc}\left(pk, m; r_{\mathsf{Enc}}\right)$	$CT \leftarrow \mathsf{SimEnc}(r_{\mathsf{SimGen}}, \mathsf{Leak}(m; r); r_{\mathsf{SimEnc}})$
	$(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, m, r)$
$\mathsf{out} \leftarrow \mathcal{A}\left(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}}\right)$	$\mathsf{out} \leftarrow \mathcal{A}\left(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}}\right)$

We say that $\mathtt{NCE}$ is weakly secure with respect to Leak if there exist PPT simulators $(\mathsf{SimGen}, \mathsf{SimEnc}, \mathsf{Open})$ such that for any PPT adversary $\mathcal{A}$ and any message m,

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

holds.

Remind that in this definition, following the style of Definition 2.6, the challenge message is fixed in advance independently of the public key.

Weak security with respect to $\mathsf{Leak} = \bot$ in which the target message is chosen by the adversary is exactly the same notion as the full-fledged security for NCE which we recall below.

Definition 3.7 (Weak Non-Committing Encryption)**.** We say $\mathtt{NCE}$ is a weak non-committing encryption scheme if it satisfies the above weak correctness and weak security.

3.3.2 Construction from Key Encapsulation Mechanism

In this section we show a construction of weak non-committing scheme with $\mathcal{O}(\lambda)$ public-key and ciphertext expansion. Before describe the construction, we define its building block, key-encapsulation mechanism (KEM). Basically KEM is almost a public-key encryption scheme, in which the encapsulation algorithm E, instead of taking message m as input, outputs a session key $K \in \{0,1\}^\ell$ that is indistinguishable to a uniformly random string. Furthermore we require KEM to satisfy *oblivious samplability* which guarantees that we can sample random public-key and ciphertext without knowing corresponding randomness $r_{\mathsf{Gen}}, r_{\mathsf{Enc}}$.

Definition 3.8 (Key-Encapsulation Mechanism)**.** A key-encapsulation mechanism $\mathtt{KEM}$ consists of the following PPT algorithms $(\mathsf{G}, \mathsf{E}, \mathsf{D})$, where G is the key-generation algorithm, E is the encapsulation algorithm, and D is the decapsulation algorithm.

- $\mathsf{G}(1^\lambda; r_{\mathsf{G}})$: Given the security parameter 1^λ, it outputs a public key pk and a secret key sk.
- $\mathsf{E}(\mathsf{pk}; r_{\mathsf{E}})$: Given the public key pk, it outputs a ciphertext ct and a session key $K \in \{0,1\}^\ell$.

- $\mathsf{D}(\mathsf{sk}, \mathsf{ct})$: Given the secret key sk and the ciphertext ct, it outputs the session key K.

Definition 3.9 (Correctness)**.** The correctness for key-encapsulation mechanism means that the session keys output by the encapsulation and decapsulation algorithms are the same. In formal, we say a key-encapsulation mechanism is correct if after executing

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{G}(1^\lambda), (\mathsf{ct}, K) \leftarrow \mathsf{E}(\mathsf{pk}),$$
$$\Pr[K \neq \mathsf{D}(\mathsf{sk}, \mathsf{ct})] = \mathsf{negl}(\lambda)$$

holds.

Definition 3.10 (Oblivious Samplability)**.** Consider the following oblivious sampling algorithms for key-generation and encapsulation $(\widehat{\mathsf{G}}, \widehat{\mathsf{E}})$, and an invert sampling algorithm $\mathsf{Inv}_{\mathtt{KEM}}$.

- $\widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$: Given the security parameter 1^λ, it outputs a obliviously sampled public key $\widehat{\mathsf{pk}}$. The randomness used in this algorithm $r_{\widehat{\mathsf{G}}}$ is essentially its output $\widehat{\mathsf{pk}}$ itself.
- $\widehat{\mathsf{E}}(\widehat{\mathsf{pk}}; r_{\widehat{\mathsf{E}}})$: Given a public key $\widehat{\mathsf{pk}}$, it outputs a obliviously sampled ciphertext $\widehat{\mathsf{ct}}$. The randomness used in this algorithm $r_{\widehat{\mathsf{E}}}$ is essentially its output $\widehat{\mathsf{ct}}$ itself.
- $\mathsf{Inv}_{\mathtt{KEM}}(r_{\mathsf{G}}, r_{\mathsf{E}})$: Given randomness for real-life execution of KEM $(r_{\mathsf{G}}, r_{\mathsf{E}})$, it outputs randomness for oblivious sampling $(r_{\widehat{\mathsf{G}}}, r_{\widehat{\mathsf{E}}})$.

For a PPT adversary $\mathcal{A}$, we define the following real and ideal experiments.

$\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{KEM},\mathcal{A}}$	$\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{KEM},\mathcal{A}}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G}})$	$\widehat{\mathsf{pk}}, \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$
$(\mathsf{ct}, K) \leftarrow \mathsf{E}(\mathsf{pk}; r_{\mathsf{E}})$	$\widehat{\mathsf{ct}} \leftarrow \widehat{\mathsf{E}}(\widehat{\mathsf{pk}}; r_{\widehat{\mathsf{E}}})$
$(r_{\widehat{\mathsf{G}}}, r_{\widehat{\mathsf{E}}}) \leftarrow \mathsf{Inv}_{\mathtt{KEM}}(r_{\mathsf{G}}, r_{\mathsf{E}})$	$K \leftarrow \{0,1\}^\ell$
$\mathsf{out} \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}, K, r_{\widehat{\mathsf{G}}}, r_{\widehat{\mathsf{E}}})$	$\mathsf{out} \leftarrow \mathcal{A}(\widehat{\mathsf{pk}}, \widehat{\mathsf{ct}}, K, r_{\widehat{\mathsf{G}}}, r_{\widehat{\mathsf{E}}})$

We say that $\mathtt{KEM}$ is obliviously samplable if there exist above algorithms such that for all PPT adversary $\mathcal{A}$,

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

holds.

If $\mathtt{KEM}$ satisfies the above oblivious samplability, its public-key is obliviously samplable. Concretely, there exist Inv_G, such that

$$\left\{\mathsf{pk}, r_{\widehat{\mathsf{G}}} \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G}}), r_{\widehat{\mathsf{G}}} \leftarrow \mathsf{Inv}_G(r_{\mathsf{G}})\right\} \overset{c}{\approx} \left\{\widehat{\mathsf{pk}}, r_{\widehat{\mathsf{G}}} \mid \widehat{\mathsf{pk}}, \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})\right\}$$

holds.

Construction Let $\mathtt{KEM} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be an obliviously samplable key-encapsulation mechanism as in the above. We construct a weak non-committing encryption scheme $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for single bit message $x \in \{0,1\}$. Weak non-committing encryption scheme for multiple number of bits can be obtained by repeating the scheme for single bit in parallel.

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

- Sample a uniformly random bit $z \leftarrow \{0,1\}$.
- Generate a key pair of KEM: $(\mathsf{pk}_z, \mathsf{sk}_z) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G}})$.
- Obliviously sample a public key: $\mathsf{pk}_{1-z} \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$.
- Output $pk \coloneqq (\mathsf{pk}_0, \mathsf{pk}_1)$ and $sk \coloneqq (z, \mathsf{sk}_z)$.

This key-generation algorithm uses randomness $r_{\mathsf{Gen}} = (z, r_{\mathsf{G}}, r_{\widehat{\mathsf{G}}})$.

$\mathsf{Enc}(pk, x \in \{0,1\}; r_{\mathsf{Enc}})$:

- Execute encapsulation: $(\mathsf{ct}_x, K_x) \leftarrow \mathsf{E}(\mathsf{pk}_x; r_{\mathsf{E}})$.
- Obliviously sample a ciphertext: $\mathsf{ct}_{1-x} \leftarrow \widehat{\mathsf{E}}(\mathsf{pk}_{1-x}; r_{\widehat{\mathsf{E}}})$.
- Sample a uniformly random session key: $K_{1-x} \leftarrow \{0,1\}^\ell$.
- Output $CT \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1, K_0, K_1)$.

This encryption algorithm uses randomness $r_{\mathsf{Enc}} = (r_{\mathsf{E}}, r_{\widehat{\mathsf{E}}}, K_{1-x})$.

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

- If $K_z = \mathsf{D}(\mathsf{sk}_z, \mathsf{ct}_z)$, output $x = z$, otherwise $x = 1 - z$. Equivalently, output $x = z \oplus \bigvee_{t=1}^{\ell} (K_z \oplus \mathsf{D}(\mathsf{sk}_z, \mathsf{ct}_z))_t$.

Theorem 3.6 (Weak Correctness)**.** Let ℓ be a constant noticeably larger than $\log(1/\epsilon) - 1$. Suppose $\mathtt{KEM}$ is correct, then the above NCE has ϵ-decryption error.

Proof. Decryption error is caused either when $x \neq z$ and $K_z = \mathsf{D}(\mathsf{sk}_z, \mathsf{ct}_z)$ happen or $x = z$ and the underlying KEM scheme causes decryption error.

$$
\begin{aligned}
&\Pr[x \neq \mathsf{Dec}\,(sk, CT)] \\
=&\Pr[z = x] \Pr[x \neq \mathsf{Dec}\,(sk, CT)\,|z = x] + \Pr[z \neq x] \Pr[[x \neq \mathsf{Dec}\,(sk, CT)\,|z \neq x] \\
=&\frac{1}{2}\left(\Pr[K_x \neq \mathsf{D}(\mathsf{sk}_z, \mathsf{ct}_x)|z = x] + \Pr[K_{1-x} = \mathsf{D}(\mathsf{sk}_z, \mathsf{ct}_{1-x})|z \neq x]\right) \\
=&\frac{1}{2}\left(\mathsf{negl}(\lambda) + \frac{1}{2^\ell}\right) \leq \epsilon
\end{aligned}
$$

□

Theorem 3.7 (Weak Security)**.** If KEM is an obliviously samplable key-encapsulation mechanism, then NCE is weakly secure with respect to $\mathsf{Leak} = \mathsf{BEC}_{0.5}$.

Proof. We construct a tuple of simulators as follows.

$\mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$:

- Generate two KEM key pairs: $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},0})$ and $(\mathsf{pk}_1, \mathsf{sk}_0) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},1})$.
- Output a simulated public key $pk \coloneqq (\mathsf{pk}_0, \mathsf{pk}_1)$.

$\mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{x} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}}); r_{\mathsf{SimEnc}})$:

- Regenerate pk_0 and pk_1 from r_{SimGen}.
- If $\tilde{x} = \bot$,
  - Execute the encapsulation twice: $(\mathsf{ct}_0, K_0) \leftarrow \mathsf{E}(\mathsf{pk}_0; r_{\mathsf{E},0})$ and $(\mathsf{ct}_1, K_1) \leftarrow \mathsf{E}(\mathsf{pk}_1; r_{\mathsf{E},1})$.
  - Output a simulated ciphertext $CT \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1, K_0, K_1)$.
- If $\tilde{x} = x \neq \bot$, just execute $CT \leftarrow \mathsf{Enc}(pk, x; r_{\mathsf{Enc}})$.

$\mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, x, r_{\mathsf{ch}})$:

- Set $z \coloneqq x \oplus 1 \oplus r_{\mathsf{ch}}$.[2]
- If $z = x$ $(\Leftrightarrow \tilde{x} = \bot)$,
  - Execute the invert sampling: $(r_{\widehat{\mathsf{G}},1-z}, r_{\widehat{\mathsf{E}},1-z}) \leftarrow \mathsf{Inv}_{\mathtt{KEM}}(r_{\mathsf{G},1-z}, r_{\mathsf{E},1-z})$.
  - Output simulated randomness:

$$r_{\mathsf{Gen}} \coloneqq \left(z, r_{\mathsf{G},z}, r_{\widehat{\mathsf{G}},1-z}\right) \quad \text{and} \quad r_{\mathsf{Enc}} \coloneqq \left(r_{\mathsf{E},x}, r_{\widehat{\mathsf{E}},1-x}, K_{1-x}\right).$$

- If $z \neq x$ $(\Leftrightarrow \tilde{x} \neq \bot)$,
  - Execute invert sampling for the key-generation: $(r_{\widehat{\mathsf{G}},1-z}) \leftarrow \mathsf{Inv}_{\mathsf{G}}(r_{\mathsf{G},1-z})$.
  - Output simulated randomness: $r_{\mathsf{Gen}} \coloneqq \left(z, r_{\mathsf{G},z}, r_{\widehat{\mathsf{G}},1-z}\right)$ and r_{Enc}.

Let $\mathcal{A}$ be a PPT adversary against weak security of NCE. The message is set to $x \in \{0,1\}$. We define the following sequence of experiments.

Exp 0: This experiment is exactly the same as $\mathsf{Exp}^{\text{Weak Real}}_{\mathtt{NCE},\mathcal{A}}$. Specifically, the experiment proceeds as follows.

1. Sample a uniformly random bit $z \leftarrow \{0,1\}$.
2. Generate a key pair of KEM: $(\mathsf{pk}_z, \mathsf{sk}_z) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G}})$.

[2]Remind that $r_{\mathsf{ch}} = 1 \Leftrightarrow z = x$ means that message to the adversary is erased in BEC, and thus the encryption is secure.

3. Obliviously sample a public key: $\mathsf{pk}_{1-z} \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$.
4. Set $pk \coloneqq (\mathsf{pk}_0, \mathsf{pk}_1)$ and $r_{\mathsf{Gen}} \coloneqq (z, r_{\mathsf{G}}, r_{\widehat{\mathsf{G}}})$.
5. Execute encapsulation: $(\mathsf{ct}_x, K_x) \leftarrow \mathsf{E}(\mathsf{pk}_x; r_{\mathsf{E}})$.
6. Obliviously sample a ciphertext: $\mathsf{ct}_{1-x} \leftarrow \widehat{\mathsf{E}}(\mathsf{pk}_{1-x}; r_{\widehat{\mathsf{E}}})$.
7. Sample a uniformly random session key: $K_{1-x} \leftarrow \{0,1\}^\ell$.
8. Set $CT \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1, K_0, K_1)$ and $r_{\mathsf{Enc}} \coloneqq (r_{\mathsf{E}}, r_{\widehat{\mathsf{E}}}, K_{1-x})$.
9. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}})$.

Exp 1: This experiment is the same as Exp 0, except that the sampling $z \leftarrow \{0,1\}$ is replaced to the followings:

- Compute $\tilde{x} \leftarrow \mathsf{BEC}(x; r_{\mathsf{ch}})$.
- Set $z \coloneqq x \oplus 1 \oplus r_{\mathsf{ch}}$.

Since r_{ch} is sampled from $\mathcal{B}_{0.5}$, z computed in Exp 1 is also uniformly random. Thus $\Pr[\mathsf{out} = 1]$ in Exp 0 and Exp 1 are the same.

Exp 2: This experiment is the same as Exp 1, except that if $z = x$, or equivalently $\tilde{x} = \perp$, oblivious sample of the public key, the ciphertext, and the session key $\mathsf{pk}_{1-z} \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$, $\mathsf{ct}_{1-x} \leftarrow \widehat{\mathsf{E}}(\mathsf{pk}_{1-x}; r_{\widehat{\mathsf{E}}})$, $K_{1-x} \leftarrow \{0,1\}^\ell$ is replaced to the followings:

- Generate public key by the key-generation: $(\mathsf{pk}_{1-z}, \mathsf{sk}_{1-z},) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},1-z})$,
- Generate ciphertext and session key by the encapsulation: $(\mathsf{ct}_{1-x}, K_{1-x}) \leftarrow \mathsf{E}(\mathsf{pk}_{1-x}; r_{\mathsf{E},1-x})$.
- The randomness is inverted as $(r_{\widehat{\mathsf{G}},1-z}, r_{\widehat{\mathsf{E}},1-z}) \leftarrow \mathsf{Inv}_{\mathtt{KEM}}(r_{\mathsf{G},1-z}, r_{\mathsf{E},1-z})$.

We also rename $(r_{\mathsf{G}}, r_{\mathsf{E}})$ to $(r_{\mathsf{G},z}, r_{\mathsf{E},x})$.

Lemma 3.8. Assume the oblivious samplability of KEM, the difference of $\Pr[\mathsf{out} = 1]$ between Exp 1 and Exp 2 is negligible.

Exp 3: This experiment is the same as Exp 2, except that if $z \neq x$, or equivalently $\tilde{x} = x \neq \perp$, the oblivious sample of the public key $\mathsf{pk}_{1-z} \leftarrow \widehat{\mathsf{G}}(1^\lambda; r_{\widehat{\mathsf{G}}})$ is replaced to the following:

- Generate public key by the key-generation: $(\mathsf{pk}_{1-z}, \mathsf{sk}_{1-z},) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},1-z})$.
- The randomness is inverted as $(r_{\widehat{\mathsf{G}},1-z}) \leftarrow \mathsf{Inv}_{\mathsf{G}}(r_{\mathsf{G},1-z})$.

We also rename r_{G} to $r_{\mathsf{G},z}$.

Lemma 3.9. Assume the oblivious samplability of public key, the difference of $\Pr[\mathsf{out} = 1]$ between Exp 2 and Exp 3 is negligible.

Now, Exp 3 is exactly the same as $\mathsf{Exp}^{\text{Weak Ideal}}_{\mathtt{NCE},\mathcal{A}}$. Specifically, this experiment proceeds as follows.

1. Generate two KEM key pairs: $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},0})$ and $(\mathsf{pk}_1, \mathsf{sk}_0) \leftarrow \mathsf{G}(1^\lambda; r_{\mathsf{G},1})$.
2. Set $pk \coloneqq (\mathsf{pk}_0, \mathsf{pk}_1)$.
3. If $\tilde{x} = \bot$, execute the encapsulation twice: $(\mathsf{ct}_0, K_0) \leftarrow \mathsf{E}(\mathsf{pk}_0; r_{\mathsf{E},0})$ and $(\mathsf{ct}_1, K_1) \leftarrow \mathsf{E}(\mathsf{pk}_1; r_{\mathsf{E},1})$.
4. If $\tilde{x} = x \neq \bot$, just execute $\mathsf{Enc}(pk, x; r_{\mathsf{Enc}})$.
5. Set $CT \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1, K_0, K_1)$.
6. Set $z \coloneqq x \oplus 1 \oplus r_{\mathsf{ch}}$.
7. If $z = x$ $(\Leftrightarrow \tilde{x} = \bot)$,
   - Execute the invert sampling: $(r_{\widehat{\mathsf{G}},1-z}, r_{\widehat{\mathsf{E}},1-z}) \leftarrow \mathsf{Inv}_{\mathtt{KEM}}(r_{\mathsf{G},1-z}, r_{\mathsf{E},1-z})$.
   - Output simulated randomness:
   $$r_{\mathsf{Gen}} \coloneqq \left(z, r_{\mathsf{G},z}, r_{\widehat{\mathsf{G}},1-z}\right) \quad \text{and} \quad r_{\mathsf{Enc}} \coloneqq \left(r_{\mathsf{E},x}, r_{\widehat{\mathsf{E}},1-x}, K_{1-x}\right).$$
8. If $z \neq x$ $(\Leftrightarrow \tilde{x} \neq \bot)$,
   - Execute invert sampling for the key-generation: $(r_{\widehat{\mathsf{G}},1-z}) \leftarrow \mathsf{Inv}_{\mathsf{G}}(r_{\mathsf{G},1-z})$.
   - Output simulated randomness: $r_{\mathsf{Gen}} \coloneqq \left(z, r_{\mathsf{G},z}, r_{\widehat{\mathsf{G}},1-z}\right)$ and r_{Enc}.
9. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}})$.

By combining the above lemma, the proof of Theorem 3.7 completes.

□

3.4 Full-Fledged NCE from Weak NCE

In this section, we amplify a weak NCE scheme into a full-fledged one using conditionally invertible wiretap codes.

Construction. Let $\mathtt{NCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a weak NCE scheme which has ϵ-decryption error and weak security with respect to $\mathsf{BEC}_{0.5}$, and wiretap codes $\mathtt{WC} = (\mathsf{WC.Setup}, \mathsf{WC.Encode}, \mathsf{WC.Decode})$ which is correct over receiver channel $\mathsf{BSC}_{\leq\epsilon}$ and conditionally invertible against the adversary channel $\mathsf{BEC}_{0.5}$. We construct a full-fledged NCE scheme $\mathtt{NCE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows. Note that this construction includes the hybrid encryption with one-time pad, thus the amplified scheme satisfies the non-committing security for messages that is chosen depending on the public key.

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

- Sample a public seed of the wiretap codes $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.

- Generate a key pair of weak NCE $(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$.
- Output $(pk', sk') \coloneqq ((p, pk), sk)$.

The randomness for key generation ${r_{\mathsf{Gen}}}'$ is r_{Gen}.

$\mathsf{Enc}'(pk', m)$:

- Sample a key for one-time pad $k \leftarrow \{0,1\}^\mu$.[3]
- Encode the key as $x \leftarrow \mathsf{WC.Encode}(p, k; s) \in \{0,1\}^n$.
- Compute $CT \leftarrow \mathsf{Enc}(pk, x; r_{\mathsf{Enc}})$.
- Output ciphertext $CT' = (CT, m \oplus k)$.

The randomness for encryption ${r_{\mathsf{Enc}}}'$ is (r_{Enc}, k, s).

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

- Parse CT' as (c_1, c_2).
- Compute $k = \mathsf{WC.Decode}(p, \mathsf{Dec}(sk, c_1))$.
- Output $m = c_2 \oplus k$.

Ciphertext Expansion. The ciphertext expansion of $\mathtt{NCE}'$ is

$$\frac{\text{ciphertext expansion of } \mathtt{NCE}}{\text{rate of } \mathtt{WC}} + 1. \tag{3.1}$$

Since the rate of the wiretap codes is constant, this amplification increases ciphertext expansion only by a constant factor. Combining the ciphertext expansion given in Section 4.5, we will estimate its concrete value for our scheme in Section 5.

Public-key Expansion. The public-key expansion of $\mathtt{NCE}'$ is

$$\text{public-key expansion of } \mathtt{NCE} + o(1) \tag{3.2}$$

because this amplification only puts common random seed p into public-key, whose size is independent to the message length, thus it dose not increase public-key expansion of the amplified NCE scheme asymptotically.

Correctness. Due to the decryption error of $\mathtt{NCE}$, each bit of the decrypted code word x is flipped with probability at most ϵ. The wiretap codes correct this error as shown below.

Theorem 3.10 (Correctness)**.** If $\mathtt{NCE}$ has ϵ-decryption error, and $\mathtt{WC}$ is correct over $\mathsf{BSC}_{\leq\epsilon}$, then $\mathtt{NCE}'$ is correct.

[3]Note that weak security of NCE requires the challenge message to be independent of the public key. To address this issue, we use one-time pad in this amplification.

Proof. The probability of $\mathtt{NCE}'$ fails to decrypt is evaluated as

$$\begin{aligned}&\Pr[k \neq \mathsf{WC.Decode}(p, \mathsf{Dec}(sk, \mathsf{Enc}(pk, x)))]\\ &= \Pr[k \neq \mathsf{WC.Decode}(p, \mathsf{BSC}_{\leq \epsilon}(\mathsf{WC.Encode}(p, k; s)))]\\ &= \mathsf{negl}(\lambda)\,.\end{aligned}$$

Thus $\mathtt{NCE}'$ is correct. □

Security. We now show the non-committing security of $\mathtt{NCE}'$.

Theorem 3.11 (Security)**.** If $\mathtt{NCE}$ is weakly secure with respect to $\mathsf{BEC}_{0.5}$, and $\mathtt{WC}$ is invertible conditioned on $\mathsf{BEC}_{0.5}$, then $\mathtt{NCE}'$ is secure.

Proof. We first construct a simulator of $\mathtt{NCE}'$ $(\mathsf{Sim}', \mathsf{Open}')$ from the simulator $(\mathsf{SimGen}, \mathsf{SimEnc}, \mathsf{Open})$ of $\mathtt{NCE}$, and the inverter $\mathsf{WC.Invert}$ of $\mathtt{WC}$.

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

- Sample $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.
- Generate $pk \leftarrow \mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$.
- Sample $k \leftarrow \{0,1\}^\mu$.
- Compute $\tilde{x} \leftarrow \mathsf{BEC}_{0.5}(\mathsf{WC.Encode}(p, 0^\mu; s'); r_{\mathsf{ch}}{}')$.
- Compute $CT \leftarrow \mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{x}; r_{\mathsf{SimEnc}})$.
- Set $pk' = (p, pk), CT' = (CT, k)$.
- Output (pk', CT').

$\mathsf{Open}'(r_{\mathsf{Sim}}, m)$:

- $(s, r_{\mathsf{ch}}) \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m \oplus k)$.
- $(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, \mathsf{WC.Encode}(p, m \oplus k; s), r_{\mathsf{ch}})$.
- Output $(r_{\mathsf{Gen}}{}', r_{\mathsf{Enc}}{}') = (r_{\mathsf{Gen}}, (r_{\mathsf{Enc}}, m \oplus k, s))$.

Let $\mathcal{A}$ be an adversary against the security of $\mathtt{NCE}'$. We then define the following experiments:

Exp 0 : This experiment is the same as $\mathsf{Exp}^{\mathrm{Real}}_{\mathtt{NCE}'\mathcal{A}}$. Specifically,

1. Sample $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.
2. Generate the key pair $(pk, sk) \leftarrow \mathsf{Gen}\left(1^\lambda; r_{\mathsf{Gen}}\right)$.
3. Run the adversary to output plaintext $m \leftarrow \mathcal{A}(p, pk)$.
4. Sample $k \leftarrow \{0,1\}^\mu$ and encoded it as $x \leftarrow \mathsf{WC.Encode}(p, k; s)$.
5. Encrypt the codeword as $CT \leftarrow \mathsf{Enc}(pk, x; r_{\mathsf{Enc}})$.

6. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}((CT, m \oplus k), r_{\mathsf{Gen}}, (r_{\mathsf{Enc}}, k, s))$.

Exp 1 : In this experiment, we use the simulator $(\mathsf{SimGen}, \mathsf{SimEnc}, \mathsf{Open})$ for NCE. The ciphertext CT is simulated by SimEnc only given partial information of the message $\tilde{x} \leftarrow \mathsf{Leak}(x)$, where $\mathsf{Leak} = \mathsf{BEC}_{0.5}$ and $x \leftarrow \mathsf{WC.Encode}(p, k; s)$ now. Specifically,

1. Sample $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.
2. Simulate the public key as $pk \leftarrow \mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$.
3. Run the adversary to output plaintext $m \leftarrow \mathcal{A}(p, pk)$.
4. Sample $k \leftarrow \{0,1\}^\mu$ and encoded it as $x \leftarrow \mathsf{WC.Encode}(p, k; s)$.
5. Compute partial information $\tilde{x} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}})$.
6. Simulate the ciphertext as $CT \leftarrow \mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{x}; r_{\mathsf{SimEnc}})$.
7. Explain the randomness for key generation and encryption as $(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, \mathsf{WC.Encode}(p, k; s), r_{\mathsf{ch}})$.
8. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}((CT, m \oplus k), r_{\mathsf{Gen}}, (r_{\mathsf{Enc}}, k, s))$.

Exp 2 : In this experiment, we completely eliminate the information of k from the input of SimEnc to simulate the ciphertext. Later $\mathsf{WC.Invert}$ determines the randomness s used in the encode. Specifically,

1. Sample $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.
2. Simulate the public key as $pk \leftarrow \mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$.
3. Run the adversary to output plaintext $m \leftarrow \mathcal{A}(p, pk)$.
4. Sample $k \leftarrow \{0,1\}^\mu$, but the codeword is $x \leftarrow \mathsf{WC.Encode}(p, 0^\mu; s')$.
5. Compute partial information $\tilde{x} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}}')$.
6. Simulate the ciphertext as $CT \leftarrow \mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{x}; r_{\mathsf{SimEnc}})$.
7. Invert the randomness for encode as $(s, r_{\mathsf{ch}}) \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, k)$.
8. Explain the randomness for key generation and encryption as $(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, \mathsf{WC.Encode}(p, k; s), r_{\mathsf{ch}})$.
9. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}((CT, m \oplus k), r_{\mathsf{Gen}}, (r_{\mathsf{Enc}}, k, s))$.

Exp 3 : In this experiment, we completely eliminate m from the ciphertext by switching k to $m \oplus k$. Specifically,

1. Sample $p \leftarrow \mathsf{WC.Setup}(1^\lambda)$.
2. Simulate the public key as $pk \leftarrow \mathsf{SimGen}(1^\lambda; r_{\mathsf{SimGen}})$.
3. Run the adversary to output plaintext $m \leftarrow \mathcal{A}(p, pk)$.
4. Sample $k \leftarrow \{0,1\}^\mu$, but the codeword is $x \leftarrow \mathsf{WC.Encode}(p, 0^\mu; s')$.

5. Compute partial information $\tilde{x} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}}{}')$.
6. Simulate the ciphertext as $CT \leftarrow \mathsf{SimEnc}(r_{\mathsf{SimGen}}, \tilde{x}; r_{\mathsf{SimEnc}})$.
7. Invert the randomness for encoding as $(s, r_{\mathsf{ch}}) \leftarrow \mathsf{WC.Invert}(p, \tilde{x}, m \oplus k)$.
8. Explain the randomness for key generation and encryption as $(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}) \leftarrow \mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, \mathsf{WC.Encode}(p, m \oplus k; s), r_{\mathsf{ch}})$.
9. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}((CT, k), r_{\mathsf{Gen}}, (r_{\mathsf{Enc}}, m \oplus k, s))$.

Note that the last experiment Exp 3 is identical to $\mathsf{Exp}^{\mathrm{Ideal}}_{\mathtt{NCE}'\mathcal{A}}$.

We show the difference between each experiments are negligible.

Lemma 3.12. If $\mathtt{NCE}$ is weakly secure with respect to $\mathsf{BEC}_{0.5}$, the difference of $\Pr[\mathsf{out} = 1]$ in Exp 0 and Exp 1 is negligible.

This lemma directly follows from the weak security of $\mathtt{NCE}$. Note that the message encrypted by $\mathtt{NCE}$ is the key of one-time pad k, which is independent of the public key.

Lemma 3.13. If $\mathtt{WC}$ is invertible conditioned on $\mathsf{BEC}_{0.5}$, the difference of $\Pr[\mathsf{out} = 1]$ in Exp 1 and Exp 2 is negligible.

By the conditional invertibility of $\mathtt{WC}$, the following items are statistically indistinguishable.

- $(\mathsf{BEC}_{0.5}(\mathsf{WC.Encode}(p, k; s); r_{\mathsf{ch}}), (s, r_{\mathsf{ch}}))$
- $(\mathsf{BEC}_{0.5}(\mathsf{WC.Encode}(p, 0^{\mu}; s'); r_{\mathsf{ch}}{}'), (s, r_{\mathsf{ch}}))$
  where (s, r_{ch}) is output of $\mathsf{WC.Invert}(p, \mathsf{BEC}_{0.5}(\mathsf{WC.Encode}(p, 0^{\mu}; s'); r_{\mathsf{ch}}{}'), k)$

The lemma follows because $(CT', r'_{\mathsf{Gen}}, r'_{\mathsf{Enc}})$, and hence out in Exp 1 are computed from the former item, while those in Exp 2 are computed from the latter item.

Lemma 3.14. $\Pr[\mathsf{out} = 1]$ is identical in Exp 2 and Exp 3.

This lemma holds unconditionally, because $(k, m \oplus k)$ and $(m \oplus k, k)$ distribute identically when k is sampled uniformly at random.

Combining the above lemmas, we complete the proof of Theorem 3.11. □

Chapter 4

NCE with Constant Ciphertext-Expansion

In this chapter, we show the main contribution of this thesis, construction of a non-committing encryption scheme with constant ciphertext expansion. First, in section 4.1, we give a brief idea to compress ciphertext of the scheme presented in section 3.1. Then, in section 4.2, we introduce obliviously samplable chameleon encryption, which is the central building block of the NCE scheme with constant ciphertext expansion. Section 4.3 and 4.4 shows the instantiations of obliviously samplable chameleon encryption based on the DDH and LWE problem, respectively. Finally, we construct a weak NCE scheme with constant ciphertext expansion from obliviously samplable chameleon encryption in section 4.5. Note that we can obtain a full-fledged NCE scheme with constant ciphertext expansion via the transformation presented in section 3.4.

4.1 Idea Towards Constant Ciphertext-Expansion

We show how to achieve the ciphertext expansion $\mathcal{O}(1)$ by compressing the ciphertext of $\mathtt{NCE}_{\mathtt{lin}}$. This is done by compressing the first part of the ciphertext of $\mathtt{NCE}_{\mathtt{lin}}$, that is,

$$\begin{pmatrix} B_{1,0}, \ldots, B_{n,0} \\ B_{1,1}, \ldots, B_{n,1} \end{pmatrix}.$$

By this step, we compress it into just a single group element.

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 $B_{i,z_i}^{\rho_i} = K_{i,z_i}$ holds. This recovery of x_i can be done even if the sender sends only B_{i,x_i}, and not $B_{i,1-x_i}$.

This is because, similarly to the equation $\mathsf{H}_{\mathbb{G}}(B_{i,z_i}^{\rho_i}) = K_{i,z_i}$, with overwhelming probability, the equation $\mathsf{H}_{\mathbb{G}}(B_{i,x_i}^{\rho_i}) = K_{i,z_i}$ holds if and only if $z_i = x_i$. For this reason, we can

compress the first part of the ciphertext on $\mathtt{NCE}_{\mathtt{lin}}$ into $(B_{1,x_1}, \ldots, B_{n,x_n})$.

We further compress $(B_{1,x_1}, \ldots, B_{n,x_n})$ into a single group element generated by multiplying them, that is, $y = \prod_{j\in[n]} B_{j,x_j}$. 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 K_{i,x_i} as

$$K_{i,x_i} = \mathsf{H}_{\mathbb{G}}\left(\prod_{j\in[n]} A_{i,x_i}^{\alpha_j}\right),$$

where α_j is the exponent of B_{j,x_j} for every $j \in [n]$ generated by the sender. The sender still generates $K_{i,1-x_i}$ as a random ℓ bit string for every $i \in [n]$. In this case, with overwhelming probability, the receiver can recover x_i by checking whether $K_{i,z_i} = \mathsf{H}_{\mathbb{G}}(y^{\alpha_i})$ 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 $z_i = x_i$ as in the proof of $\mathtt{NCE}_{\mathtt{lin}}$, we have to change the distribution of $K_{i,1-x_i}$ from a random string to $\mathsf{H}_{\mathbb{G}}(\prod_{j\in[n]} A_{i,1-x_i}^{\alpha_j})$ so that $K_{i,0}$ and $K_{i,1}$ 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 avoid this problem, we further modify the scheme and construct an NCE scheme NCE as follows.

The resulting NCE scheme NCE. In NCE, the receiver first generates $z \leftarrow \{0,1\}^n$ 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

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

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

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

to the public key. However, the receiver cannot do it since the receiver generates $A_{i,1-z_i}$ 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 $\{A_{i,1-z_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,z_i)}$ 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 $\{A_{i,1-z_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,z_i)}$ instead of random group elements.

When encrypting a message m by NCE, the sender first generates $x \leftarrow \{0,1\}^n$ and computes $y = \prod_{j\in[n]} B_{j,x_j}$. Then, for every $i \in [n]$, the sender computes K_{i,x_i} as

$$K_{i,x_i} = \mathsf{H}_{\mathbb{G}}\left(\prod_{j\in[n]} A_{i,x_i}^{\alpha_{j,x_j}}\right) = \mathsf{H}_{\mathbb{G}}(y^{\rho_i})$$

just multiplying $A_{i,x_i}^{\alpha_{1,x_1}}, \ldots, A_{i,x_i}^{\alpha_{n,x_n}}$ included in the pubic key. Recall that $A_{i,x_i} = g^{\rho_i}$. Note that $A_{i,z_i}^{\alpha_{i,1-z_i}}$ is not included in the public key, but we do not need it to compute K_{i,x_i}. The sender generates K_{i,x_i} as a random string for every $i \in [n]$ as before. The resulting ciphertext is

$$\left(y, \begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}\right).$$

The receiver can recover x_i by checking whether $K_{i,z_i} = \mathsf{H}_{\mathbb{G}}\left(y^{\alpha_i}\right)$ holds, and decrypt the ciphertext.

By defining the simulator appropriately, the security proof of $\mathtt{NCE}$ proceeds in a similar way to that of $\mathtt{NCE}_{\mathtt{lin}}$. In $\mathtt{NCE}$, for indices $i \in [n]$ such that $z_i = x_i$, we can eliminate the information of x_i. We can change $K_{i,1-x_i}$ from a random string to $\mathsf{H}_{\mathbb{G}}\left(\prod_{j\in[n]} A_{i,1-x_i}^{\alpha_{j,x_j}}\right)$ by relying on the fact that $A_{i,1-x_i}^{\alpha_{i,x_i}}$ is indistinguishable from a random group element by the DDH assumption. By this change, $K_{i,0}$ and $K_{i,1}$ 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 $\mathtt{NCE}$, the first component of the ciphertext, that is, $y = \prod_{j\in[n]} B_{j,x_j}$ has the information of x. In order to deal with the issue, in our real construction, we replace y with $g^r \prod_{j\in[n]} B_{j,x_j}$, where $r \leftarrow \mathbb{Z}_p$. Then, y no longer leaks any information of x. Moreover, after y is fixed, for any $x' \in \{0,1\}^n$, we can efficiently find r' such that $y = g^{r'} \prod_{j\in[n]} B_{j,x'_j}$. This is important to ensure that the simulator of $\mathtt{NCE}$ runs in polynomial time.

4.1.1 Abstraction by Chameleon Encryption

We can describe $\mathtt{NCE}$ by using obliviously samplable chameleon encryption. Informally, chameleon encryption is public-key encryption whose public key corresponds to the output of a chameleon hash function, and whose secret key corresponds to the preimage of that hash. In the construction of $\mathtt{NCE}$, $g^r \prod_{j\in[n]} B_{j,x_j}$ can be seen as an output of the chameleon hash function

$$\mathsf{H}(x; r) = g^r \prod_{j\in[n]} B_{j,x_j},$$

where $\{B_{i,b}\}_{i\in[n],b\in\{0,1\}}$ is the hash key. Moreover, by defining chameleon encryption as key encapsulation mechanism instead of public-key encryption[1], group elements contained in the public key and $\{K_{i,b}\}_{i\in[n],b\in\{0,1\}}$ together form multiple 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 $\{A_{i,1-z_i}^{\alpha_{j,b}}\}_{j\in[n],b\in\{0,1\}\ s.t.\ (j,b)\neq(i,z_i)}$ for every $i \in [n]$.

[1] In other words, we define it so that it satisfies a property called recyclability introduced by Garg and Hajiabadi [GH18]. For more details, see Remark 2 in the next section.

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 [GH18].

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

$$\left(y, \begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}\right)$$

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, $\{\rho_{i,z_i}\}_{i\in[n]}$. Garg and Hajiabadi already stated that their TDF can be inverted with $\{\rho_{i,z_i}\}_{i\in[n]}$ for any $z \in \{0,1\}^n$, 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.

4.2 Obliviously Samplable Chameleon Encryption

Chameleon encryption (CE) was proposed by Döttling and Garg [DG17b]. Since then, variant of chameleon encryption is appeared in studies on identity based encryption[DG17a, DGHM18, BLSV18], secure MPC [CDG+17, GS18a], adaptive garbling schemes [GS18b, GOS18], and so on. We define its obliviously samplable variant, obliviously samplable chameleon encryption as a building block of NCE schemes with constant ciphertext expansion. Note that in order to give a unified view of oblivious samplability, the following definition of obliviously samplable chameleon encryption is further modified from the previous definition [YKT19, YKXT20]. We show an instantiation of obliviously samplable chameleon encryption based on the DDH problem in Section 4.3. The instantiation based on the LWE problem is described in Section 4.4.

Definition 4.1 (Obliviously Samplable Chameleon Encryption)**.** An obliviously samplable chameleon encryption scheme CE consists of the following PPT algorithms $(\mathsf{G}, \mathsf{H}, \mathsf{E}_1, \mathsf{E}_2, \mathsf{D})$, where key-generation algorithm G and hash function H compose a family of probabilistic

hash functions, $(\mathsf{E}_1, \mathsf{E}_2)$ is the associated encapsulation algorithm, and D is the algorithm for its decapsulation.

$\mathsf{G}\left(1^\lambda; r_\mathsf{G}\right)$: Given the security parameter 1^λ, it outputs a hash key hk.

$\mathsf{H}\left(\mathsf{hk}, x; r_\mathsf{H}\right)$: Given a hash key hk and an input $x \in \{0,1\}^n$, it outputs a hash value y.

$\mathsf{E}_1\left(\mathsf{hk}, (i,b); r_\mathsf{E}\right)$: Given a hash key hk, an index $i \in [n], b \in \{0,1\}$, it outputs a ciphertext ct.

$\mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_\mathsf{E}\right)$: Given a hash key hk, an index $i \in [n], b \in \{0,1\}$, and a hash value y, using the same randomness as E_1, it outputs $K \in \{0,1\}^\ell$.

$\mathsf{D}\left(\mathsf{hk}, (x, r_\mathsf{H}), \mathsf{ct}\right)$: Given a hash key hk, a preimage of the hash function (x, r_H), and a ciphertext ct, it outputs $K \in \{0,1\}^\ell$.

An obliviously samplable CE scheme satisfies the following correctness, trapdoor collision property, oblivious samplability of hash keys, and security with oblivious samplability.

Definition 4.2 (Correctness)**.** We say that a obliviously samplable chameleon encryption scheme is correct if for all $x \in \{0,1\}^n$ and $i \in [n]$, after execution of

$$\begin{aligned}
&\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_\mathsf{G}\right), y \leftarrow \mathsf{H}\left(\mathsf{hk}, x; r_\mathsf{H}\right), \\
&\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{hk}, (i, x_i); r_\mathsf{E}), K \leftarrow \mathsf{E}_2(\mathsf{hk}, (i, x_i), y; r_\mathsf{E}), \\
&\Pr[K \neq \mathsf{D}\left(\mathsf{hk}, (x, r_\mathsf{H}), \mathsf{ct}\right)] = \mathsf{negl}(\lambda)
\end{aligned}$$

holds, where x_i denotes the i-th bit of x.

Definition 4.3 (All-in-One Security)**.** We give an all-in-one security definition for obliviously samplable chameleon encryption, which is required to construct the proposed NCE scheme. This definition contains non-committing security for the hash function of chameleon encryption and oblivious samplability for the associated encryption scheme.

Consider the following PPT algorithms for simulation and opening of chameleon hash and oblivious sampling and invert of associated encryption $(\mathsf{SimCH}, \mathsf{Open}_{\mathtt{CH}}, \widehat{\mathsf{E}_1}, \mathsf{Inv}_{\mathtt{CE}})$.

We introduce an obliviously sampling algorithms for ciphertexts.

- $\mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$: Give a security parameter 1^λ, it simulates a hash key hk and a hash value y.
- $\mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$: Give an input x to the chameleon hash, it outputs randomness for key-generation and hash $(r_\mathsf{G}, r_\mathsf{H})$.
- $\widehat{\mathsf{E}_1}\left(\mathsf{hk}, (i,b); r_{\widehat{\mathsf{E}}}\right)$: Given a hash key hk, an 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.
- $\mathsf{Inv}_{\mathtt{CE}}(\mathsf{hk}, r_\mathsf{E})$: Give a hash key hk and randomness for encryption r_E, it outputs randomness for oblivious sampling ciphertext $r_{\widehat{\mathsf{E}}}$.

For a chameleon encryption scheme and a stateful adversary $\mathcal{A}$, we define two experiments as follows.

$\mathsf{Exp}^{\mathrm{Real}}$	$\mathsf{Exp}^{\mathrm{Ideal}}$
$\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_{\mathsf{G}}\right)$	$(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$
$y \leftarrow \mathsf{H}\left(\mathsf{hk}, x; r_{\mathsf{H}}\right)$	$(r_{\mathsf{G}}, r_{\mathsf{H}}) \leftarrow \mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$
$\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{hk}, (i, 1-x_i); r_{\mathsf{E}})$	$\mathsf{ct} \leftarrow \widehat{\mathsf{E}_1}(\mathsf{hk}, (i, (1-x_i); r_{\widehat{\mathsf{E}}}))$
$K \leftarrow \mathsf{E}_2(\mathsf{hk}, (i, 1-x_i), y; r_{\mathsf{E}})$	$K \leftarrow \{0,1\}^\ell$
$r_{\widehat{\mathsf{E}}} \leftarrow \mathsf{Inv}_{\mathtt{CE}}(\mathsf{hk}, r_{\mathsf{E}})$	
$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\mathsf{G}}, r_{\mathsf{H}}, r_{\widehat{\mathsf{E}}}\right)$	$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\mathsf{G}}, r_{\mathsf{H}}, r_{\widehat{\mathsf{E}}}\right)$

We say that $\mathtt{CE}$ is secure if there exist above algorithms such that for all $x \in \{0,1\}^n, i \in \{0,1\}$, and for all PPT adversary $\mathcal{A}$,

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

holds.

The above all-in-one security captures the required security notion at once. However when we use it in the security proof of NCE scheme, it is useful to separate it into the chameleon hash part and the associated encryption part. We give separated security definitions below.

Definition 4.4 (Security of Chameleon Hash)**.** As an extension of hiding property of chameleon hash function, we define a non-committing style security notion for chameleon hash function. Note that this security notion is different from the one usually defined for chameleon hash functions. Usually, the hash algorithm with another input $\mathsf{H}(\mathsf{hk}, x'; r_{\mathsf{H}})$ plays the role of generating the hash value y in the ideal world. Non-committing property is somewhat weaker than the usual chameleon property, still, it is enough for construction of the NCE scheme.

Furthermore, this security notion captures oblivious samplability of the hash key. This means that in the real world, we can sample a hash key without knowing trapdoor information contained in r_{SimCH} which is used in the opening in the ideal world. So, this primitive can be called *non-committing hash function with oblivious key generation*, rather than chameleon hash function.

$\mathsf{Exp}^{\mathrm{Real}}$	$\mathsf{Exp}^{\mathrm{Ideal}}$
$\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_{\mathsf{G}}\right)$	$(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$
$y \leftarrow \mathsf{H}\left(\mathsf{hk}, x; r_{\mathsf{H}}\right)$	$(r_{\mathsf{G}}, r_{\mathsf{H}}) \leftarrow \mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$
$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, r_{\mathsf{G}}, r_{\mathsf{H}}\right)$	$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, r_{\mathsf{G}}, r_{\mathsf{H}}\right)$

We say that $\mathtt{CE}$ is secure if there exist above algorithms such that for all $x \in \{0,1\}^n$, and for all PPT adversary $\mathcal{A}$,

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

holds.

Next, we focus on the obliviously samplable security of the associated encryption under simulated chameleon hash.

Definition 4.5 (Oblivious Samplability of Associated Encryption)**.** We say ciphertext of the chameleon encryption scheme is obliviously samplable (under simulated chameleon hash) if for any $x \in \{0,1\}^n$, $i \in [n]$, and PPT adversary $\mathcal{A}$, define two experiments as follows.

$\mathsf{Exp}^{\mathrm{Real}}$	$\mathsf{Exp}^{\mathrm{Ideal}}$
$(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$	$(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$
$\mathsf{ct} \leftarrow \mathsf{E}_1(\mathsf{hk}, (i, 1-x_i); r_{\mathsf{E}})$	$\mathsf{ct} \leftarrow \widehat{\mathsf{E}}_1(\mathsf{hk}, (i, (1-x_i); r_{\widehat{\mathsf{E}}}))$
$K \leftarrow \mathsf{E}_2(\mathsf{hk}, (i, 1-x_i), y; r_{\mathsf{E}})$	$K \leftarrow \{0,1\}^\ell$
$r_{\widehat{\mathsf{E}}} \leftarrow \mathsf{Inv}_{\mathsf{CE}}(\mathsf{hk}, r_{\mathsf{E}})$	
$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\widehat{\mathsf{E}}}\right)$	$\mathsf{out} = \mathcal{A}\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\widehat{\mathsf{E}}}\right)$

Then, we have

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

Remark 2 (On the Recyclability)**.** The above syntax of chameleon encryption is different from that of original chameleon encryption proposed by Döttling and Garg [DG17b]. We define chameleon encryption to satisfy a property called recyclability introduced by Garg and Hajiabadi [GH18], in which recyclability is defined for one-way function with encryption, that is a similar primitive to chameleon encryption.

More specifically, in our definition, there are two encryption algorithms E_1 and E_2. E_1 outputs only a ciphertext and E_2 outputs only a session key. 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. What is important here is that, an output of E_1 does not depend on a hash value y. This makes it 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.

4.3 Instantiation based on the DDH Problem

We show an instantiation of obliviously samplable chameleon encryption scheme based on the DDH problem. The main idea for this construction is based on the original construction of chameleon encryption proposed by Döttling and Garg [DG17b].

4.3.1 Preliminaries on the Decisional Diffie-Hellman Problem

We give a definition of the decisional Diffie-Hellman (DDH) assumption and its variants used in the proof of Theorem 4.3. Below, we let $\mathbb{G}$ be a cyclic group of prime order p with a generator g.

Let $\mathcal{H} = \{\mathsf{H}_{\mathbb{G}} : \mathbb{G} \to \{0,1\}^{\ell}\}$ be a family of universal hash functions for the group elements. We use this hash function in the construction of chameleon encryption scheme to compress the session key size to constant ℓ.

We start with the standard DDH assumption.

Definition 4.6 (Decisional Diffie-Hellman Assumption)**.** We say that the DDH assumption holds if for any PPT adversary $\mathcal{A}$,

$$|\Pr\left[\mathcal{A}\left(g_1, g_2, g_1^{\rho}, g_2^{\rho}\right) = 1\right] - \Pr\left[\mathcal{A}\left(g_1, g_2, u_1, u_2\right) = 1\right]| = \mathsf{negl}(\lambda)$$

holds, where $g_1, g_2, u_1, u_2 \leftarrow \mathbb{G}$ and $\rho \leftarrow \mathbb{Z}_p$.

We also define a generalized version of the DDH assumption. This generalized version is useful in the proof of oblivious samplability of the chameleon encryption scheme.

Definition 4.7 (n-Generalized DDH Assumption)**.** We say that the n-Generalized DDH assumption holds if for any PPT adversary $\mathcal{A}$,

$$|\Pr\left[\mathcal{A}\left(\{g_i\}_{i\in[n]}, \{g_i^{\rho}\}_{i\in[n]}\right) = 1\right] - \Pr\left[\mathcal{A}\left(\{g_i\}_{i\in[n]}, \{u_i\}_{i\in[n]}\right) = 1\right]| = \mathsf{negl}(\lambda)$$

holds, where $g_i, u_i, \leftarrow \mathbb{G}$ for all $i \in [n]$ and $\rho \leftarrow \mathbb{Z}_p$.

Note that the 2-generalized DDH assumption is exactly same as the standard DDH assumption. Moreover n-generalized DDH assumption with $n \geq 2$ is equivalent to the standard DDH assumption.

Lemma 4.1. Assume the DDH assumption holds, the n-generalized DDH assumption also holds.

Proof. Let $\mathcal{A}$ be an adversary against the n-generalized DDH problem. We construct a reduction algorithm $\mathcal{A}'$ that solves the DDH problem.

At first, the reduction algorithm receives (g_1, g_2, u_1, u_2) where $(u_1, u_2) = (g_1^{\rho}, g_2^{\rho})$ or uniformly random group elements. Then samples $\{(s_i, t_i)\}_{i\in\{3,\ldots,n\}} \leftarrow \mathbb{Z}_p^{2\times(n-2)}$ and set $g_i = g_1^{s_i} g_2^{t_i}, u_i = u_1^{s_i} u_2^{t_i}$ for $i = 3, ..., n$. The reduction algorithm executes $\mathcal{A}\left(\{g_i\}_{i\in[n]}, \{u_i\}_{i\in[n]}\right)$ and output its result.

We have

$$\log \begin{pmatrix} g_3 & g_4 & \ldots & g_n \\ u_3 & u_4 & \ldots & u_n \end{pmatrix} = \begin{pmatrix} 1 & \log(g_2) \\ \log(u_1) & \log(u_2) \end{pmatrix} \begin{pmatrix} s_3 & s_4 & \ldots & s_n \\ t_3 & t_4 & \ldots & t_n \end{pmatrix},$$

where the first logarithm is applied to each component, and the base of these logarithm is g_1.

In case $(u_1, u_2) = (g_1^{\rho}, g_2^{\rho})$ holds, we have $\{g_i\}_{i\in[n]}$ are uniformly and independently random and $u_i = u_1^{s_i} u_2^{t_i} = g_1^{\rho s_i} g_2^{\rho t_i} = (g_1^{s_i} g_2^{t_i})^{\rho} = g_i^{\rho}$ holds.

In case (u_1, u_2) are random group elements, The matrix $\begin{pmatrix} 1 & \log(g_2) \\ \log(u_1) & \log(u_2) \end{pmatrix}$ is non-singular with overwhelming probability. Therefore $\left(\{g_i\}_{i\in[n]}, \{u_i\}_{i\in[n]}\right)$ are uniformly and independently random with overwhelming probability.

In each case, the reduction algorithm simulates the n-generalized DDH problem for $\mathcal{A}$. Thus it solves the DDH problem with the same but negligible loss of advantage as $\mathcal{A}$.

□

4.3.2 Construction

We construct a chameleon encryption scheme $\mathtt{CE} = (\mathsf{G}, \mathsf{H}, \mathsf{E}_1, \mathsf{E}_2, \mathsf{D})$ based on the hardness of the DDH problem.

Let $\mathbb{G}$ be a cyclic group of order p with a generator g. The description of the group $\mathbb{G}$ is generated in G and shared among other algorithms with its generator g. Strictly speaking, the randomness used to generate the group description should be described in r_{G}. We omit is because this does not matter if the group is oblivious samplable.

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 also shared to all the algorithms implicitly.

$\mathsf{G}\left(1^{\lambda}; r_{\mathsf{G}}\right)$:

- For all $i \in [n]$ and $b \in \{0,1\}$, sample $g_{i,b} \leftarrow \mathbb{G}$.
- Output $\mathsf{hk} \coloneqq \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right)$.

The randomness used in this algorithm is $r_{\mathsf{G}} \coloneqq \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}$.

$\mathsf{H}\left(\mathsf{hk}, x; r_{\mathsf{H}}\right)$:

- Sample $r \leftarrow \mathbb{Z}_p$.
- Output
$$y = g^r \prod_{i \in [n]} g_{i,x_i}.$$

The randomness used in this algorithm is $r_{\mathsf{H}} \coloneqq r$.

$\mathsf{E}_1\left(\mathsf{hk}, (i,b); r_{\mathsf{E}}\right)$:

- Parse hk as $\left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right)$.
- Sample $\rho \leftarrow \mathbb{Z}_p$ and compute $c \coloneqq g^{\rho}$.
- Compute $c_{i,b} \coloneqq (g_{i,b})^{\rho}$ and $c_{i,1-b} \coloneqq \perp$.
- For all $j \in [n]$ such that $j \neq i$, compute $c_{j,0} \coloneqq (g_{j,0})^{\rho}$ and $c_{j,1} \coloneqq (g_{j,1})^{\rho}$

- Output

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

$\mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_{\mathsf{E}}\right)$:

- Output $K \leftarrow \mathsf{H}_{\mathbb{G}}\left(y^{\rho}\right)$.

The randomness commonly used in the above two algorithm is $r_{\mathsf{E}} \coloneqq \rho$.

$\mathsf{D}\left(\mathsf{hk}, (x, r_{\mathsf{H}}), \mathsf{ct}\right)$:

- Parse ct as $\left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.
- Output $K \leftarrow \mathsf{H}_{\mathbb{G}}\left(c^r \prod_{i\in[n]} c_{i,x_i}\right)$.

Size of Parameters The group elements is represented in λ bit string. The length of the hash key and ciphertext are both $(2n+1)\lambda$.

Theorem 4.2 (Correctness)**.** This DDH-based chameleon encryption scheme is correct.

Proof. Since the hash function on the group $\mathsf{H}_{\mathbb{G}}$ is common for all algorithm, it is enough to show the equivalence of the inputs to $\mathsf{H}_{\mathbb{G}}$ in E_2 and D, that is examined as

$$y^{\rho} = \left(g^r \prod_{i\in[n]} g_{i,x_i}\right)^{\rho} = (g^{\rho})^r \prod_{i\in[n]} (g_{i,x_i})^{\rho} = c^r \prod_{i\in[n]} c_{i,x_i}.$$

□

Theorem 4.3. This obliviously samplable chameleon encryption scheme is secure assuming the hardness of the DDH problem.

Proof. First, we construct algorithms appear in the security definition.

$\mathsf{SimCH}(1^{\lambda}; r_{\mathsf{SimCH}})$:

- For all $i \in [n]$, and $b \in \{0,1\}$, sample $\alpha_{i,b} \leftarrow \mathbb{Z}_p$ and set $g_{i,b} \coloneqq g^{\alpha_{i,b}}$.
- Sample $r \leftarrow \mathbb{Z}_p$ and compute $y = g^r$.
- Output $\mathsf{hk} \coloneqq \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right)$ and y.

The randomness used in this algorithm is $r_{\mathsf{SimCH}} \coloneqq \left(\begin{pmatrix} \alpha_{1,0}, \ldots, \alpha_{n,0} \\ \alpha_{1,1}, \ldots, \alpha_{n,1} \end{pmatrix}, r\right)$.

$\mathsf{Open}_{\mathsf{CH}}(r_{\mathsf{SimCH}}, x)$:

- Recompute $g_{i,b} = g^{\alpha_{i,b}}$ for all i and b.
- Output

$$r_{\mathsf{G}} \coloneqq \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix} \quad \text{and} \quad r_{\mathsf{H}} \coloneqq r - \sum_{i \in [n]} \alpha_{i,x_i}.$$

$\widehat{\mathsf{E}_1}\left(\mathsf{hk}, (i,b); r_{\widehat{\mathsf{E}}}\right)$:

- Set $c_{i,1-b} \coloneqq \bot$, and sample $c \leftarrow \mathbb{G}$ and $c_{i,b} \leftarrow \mathbb{G}$.
- For all $j \in [n]$ such that $j \neq i$, sample $c_{j,0} \leftarrow \mathbb{G}$ and $c_{j,1} \leftarrow \mathbb{G}$.
- Output $\mathsf{ct} \coloneqq \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.

The randomness used in this algorithm is $r_{\widehat{\mathsf{E}}} = \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.

$\mathsf{Inv}_{\mathsf{CE}}(\mathsf{hk}, r_{\mathsf{E}})$:

- Recompute c and $c_{i,b}$ for all $i \in [n], b \in \{0,1\}$ from $g_{i,b}$ and ρ in the same way as E_1.
- Output $r_{\widehat{\mathsf{E}}} \coloneqq \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.

Lemma 4.4 (Non-Committing Security of Hash)**.** The chameleon hash function in the above chameleon encryption scheme satisfies Definition 4.4 unconditionally.

Proof. To proof this lemma, we need to check the distribution of $(\mathsf{hk}, y, r_{\mathsf{G}}, r_{\mathsf{H}})$ is identical in the real and ideal experiments. Especially, it is enough to check the randomness $(r_{\mathsf{G}}, r_{\mathsf{H}})$ is identically distributed and (hk, y) is determined from these randomness.

The randomness for key generation r_{G} appears in the real experiment is random $2n$ group elements $g_{i,b}$. In the ideal experiment, the group elements $g_{i,b}$ are computed from their uniformly random exponent $\alpha_{i,b}$, which are eventually uniformly random group elements. Thus r_{G} distributes identically in the both experiments. Since The hash key hk is determined by r_{G}, it is also identically distributed in the experiments.

The randomness for hash r_{H} is sampled from $\mathbb{Z}_p$ in the real experiment. In the ideal experiment, $r_{\mathsf{H}} = r - \sum_{i \in [n]} \alpha_{i,x_i}$ where r is uniformly random, hence r_{H} is also uniformly random and distributed identically to the real experiment. Since we can compute y from $\mathsf{hk}, x, r_{\mathsf{H}}$ as

$$y = g^r = g^{r - \sum_{i \in [n]} \alpha_{i,x_i}} \prod_{i \in [n]} g_{i,x_i} = \mathsf{H}(\mathsf{hk}, x; r_{\mathsf{H}})$$

in the ideal experiment, thus y is also identically distributed. □

Lemma 4.5 (Oblivious Samplability of Ciphertext)**.** Assume the $2n+1$-generalized DDH problem is hard, the associated encryption part of the chameleon encryption scheme satisfies the oblivious samplability (Definition 4.5).

Proof. Let $\mathcal{A}$ be an adversary that attacks the oblivious samplability of the chameleon encryption. We construct a reduction algorithm $\mathcal{A}'$ that solves the $2n+1$-generalized DDH problem.

Given the generalized DDH instance $\left(\{g_i^*\}_{i\in[2n+1]}, \{u_i^*\}_{i\in[2n+1]}\right)$, the reduction algorithm simulates $\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\widehat{\mathsf{E}}}\right)$ as follows.

1. Set $g \coloneqq g_1^*$ and $(g_{1,0}, \ldots, g_{n,0}, g_{1,1}, \ldots, g_{n,1}) \coloneqq (g_2^*, \ldots, g_{2n}^*)$ except g_{i,x_i} is uniformly sampled. The hash key is set to $\mathsf{hk} \coloneqq \left(g, \begin{pmatrix} g_{1,0}, \ldots, g_{n,0} \\ g_{1,1}, \ldots, g_{n,1} \end{pmatrix}\right)$.
2. The hash value is set to $y \coloneqq g_{2n+1}^*$.
3. Set $c = u_1^*$, and $(c_{1,0}, \ldots, c_{n,0}, c_{1,1}, \ldots, c_{n,1}) \coloneqq (u_2^*, \ldots, u_{2n}^*)$ except c_{i,x_i} is set to $\perp$. The ciphertext is set to $\mathsf{ct} \coloneqq \left(c, \begin{pmatrix} c_{1,0}, \ldots, c_{n,0} \\ c_{1,1}, \ldots, c_{n,1} \end{pmatrix}\right)$.
4. The session key is set to $K \coloneqq \mathsf{H}_{\mathbb{G}}(u_{2n+1}^*)$.
5. The inverted randomness for encryption r_{E} is set to the same as ct.

The reduction algorithm executes $\mathcal{A}$ with input $\left(\mathsf{hk}, y, \mathsf{ct}, K, r_{\widehat{\mathsf{E}}}\right)$ and output its result.

In case the received tuple $\left(\{g_i^*\}_{i\in[2n+1]}, \{u_i^*\}_{i\in[2n+1]}\right)$ is the generalized DDH instance, i.e. it satisfies $u_i^* = g_i^{*\rho}$ for all $i \in [2n+1]$, the reduction algorithm perfectly simulates the view of $\mathcal{A}$ in the real experiment of oblivious samplability.

In case the received tuple is uniformly and independently random, the reduction algorithm simulates the view of $\mathcal{A}$ in the ideal experiment of oblivious samplability except the session key K is hash of a random group element, not uniformly random string. This gap closes if we choose the hash $\mathsf{H}_{\mathbb{G}}$ to satisfies that if its input is uniformly random, its output is also uniformly random. More concretely, we can use a universal hash family as $\mathsf{H}_{\mathbb{G}}$.

□

Remind that $2n+1$-generalized DDH problem is as hard as the DDH problem.

Combining the above two lemma, the proof of Theorem 4.3 completes.

□

4.4 Instantiation based on Lattices

We propose a lattice-based construction of obliviously samplable CE. The public-Key length of the proposed scheme is $\lambda \cdot \mathsf{poly}(\log \lambda)$, which is smaller than $\mathcal{O}(\lambda^2)$ of the construction from the DDH problem.

The construction is similar to the construction of hash encryption from LWE proposed by Döttling et al. [DGHM18]. However we need a super-polynomially large modulus $\mathbb{Z}_q$ for the scheme to satisfy correctness. Although security of the hash encryption is claimed to be proved from a valiant of the LWE assumption, called extended-LWE, we prove the security directly from the LWE assumption.

Before describing our construction, we recall preliminaries on lattices.

4.4.1 Preliminaries on Lattices

Notations Let $\boldsymbol{A}, \boldsymbol{B}$ be matrices or vectors. $[\boldsymbol{A}|\boldsymbol{B}]$ and $[\boldsymbol{A}; \boldsymbol{B}]$ denotes concatenation of columns and rows respectively. $\boldsymbol{A}_{\setminus i}$ denotes the matrix obtained by removing the i-th column of $\boldsymbol{A}$.

The n-dimensional Gaussian function with parameter s is defined as

$$\rho_s(\boldsymbol{x}) := \exp(-\pi\|\boldsymbol{x}\|^2/s^2).$$

For positive real s and countable set A, the discrete Gaussian distribution $D_{A,s}$ is defined by

$$D_{A,s}(\boldsymbol{x}) = \rho_s(\boldsymbol{x})/\sum_{\boldsymbol{y}\in A}\rho_s(\boldsymbol{y}).$$

We note that, if $s = \omega(\log m)$,

$$\Pr_{\boldsymbol{r}\leftarrow D_{\mathbb{Z}^m,s}}[\|\boldsymbol{r}\| \le s\sqrt{m}] \ge 1 - 2^{-m+1}.$$

(See [MR07].)

Parameters. We let $n = \lambda$, $m = \mathcal{O}(n\log q)$ (e.g., $m = 2n\log q$), $q = 2^{\mathsf{poly}(\log\lambda)}$. Let χ be the discrete Gaussian distribution over $\mathbb{Z}$ with parameter $s = \omega(\sqrt{m\log n})$, that is, $D_{\mathbb{Z},s}$.

Rounding function $\mathsf{round} : \mathbb{Z}_q \to \{0,1\}$ is defined as $\mathsf{round}(v) = \lfloor 2v/q\rceil$. If input for round is a vector $\boldsymbol{v} \in \mathbb{Z}_q^\ell$, the rounding is applied to each component. Let ℓ be a constant.

Definition 4.8 ((Decisional) Learning with Errors [Reg05])**.** The LWE assumption with respect to n dimension, m samples, modulus q, and error distribution χ over $\mathbb{Z}_q$ states that for all PPT adversary $\mathcal{A}$, we have

$$\left|\Pr[\mathcal{A}(\boldsymbol{A}, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{A} + \boldsymbol{E}) = 1] - \Pr[\mathcal{A}(\boldsymbol{A}, \boldsymbol{B}) = 1]\right| = \mathsf{negl}(\lambda)\,,$$

where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}, \boldsymbol{S} \leftarrow \mathbb{Z}_q^{n\times\ell}, \boldsymbol{E} \leftarrow \chi^{m\times\ell}, \boldsymbol{B} \leftarrow \mathbb{Z}_q^{m\times\ell}$.

Definition 4.9 (Lattice Trapdoor [GPV08, MP12])**.** There exists following PPT algorithms $\mathsf{TrapGen}$ and Sample.

$\mathsf{TrapGen}(1^\lambda)$: Output a matrix $\boldsymbol{A}_T \in \mathbb{Z}_q^{n\times m}$ together with its trapdoor $\boldsymbol{T}$.

$\mathsf{Sample}(\boldsymbol{A}_T, \boldsymbol{T}, \boldsymbol{u}, s)$: Given a matrix $\boldsymbol{A}_T$ with its trapdoor $\boldsymbol{T}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a parameter s, output a vector $\boldsymbol{r} \in \mathbb{Z}^m$.

These algorithms satisfy the following two properties.

1. $\boldsymbol{A}_T$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$.
2. If $s \geq \omega(\sqrt{m \cdot \log n})$, then $\boldsymbol{r} \in \mathbb{Z}^m$ output by $\mathsf{Sample}(\boldsymbol{A}_T, \boldsymbol{T}, \boldsymbol{u}, s)$ is statistically close to $D_{\mathbb{Z}^m,s}$ conditioned on $\boldsymbol{r} \in \Lambda_{\boldsymbol{u}}(\boldsymbol{A}_T) := \{\boldsymbol{r} \in \mathbb{Z}^m \mid \boldsymbol{A}_T\boldsymbol{r} \equiv \boldsymbol{u} \pmod q\}$.

4.4.2 Construction

We construct an obliviously samplable CE scheme from the LWE problem on super-polynomially large modulus. Note that only in this section, we denotes the length of x by $|x| = N$.

$\mathsf{G}\left(1^\lambda; r_\mathsf{G}\right)$:

- Sample
$$\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times(N+m)}.$$
- Output $\mathsf{hk} := \boldsymbol{A}$.

The randomness used in this algorithm is $r_\mathsf{G} := \boldsymbol{A}$.

$\mathsf{H}\left(\mathsf{hk}, x; r_\mathsf{H}\right)$:

- Sample $\boldsymbol{r} \in \mathbb{Z}_q^m$ according to distribution $\mathcal{R}_\mathsf{H} = \chi^m$.
- Output
$$\boldsymbol{y} := \boldsymbol{A} \cdot [\boldsymbol{x}; \boldsymbol{r}] \in \mathbb{Z}_q^n.$$

The randomness used in this algorithm is $r_\mathsf{H} := \boldsymbol{r}$.

$\mathsf{E}_1\left(\mathsf{hk}, (i, b); r_\mathsf{E}\right)$:

- Sample $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n\times\ell}$ and $\boldsymbol{E} \leftarrow \chi^{\ell\times(N+m)}$.
- Output
$$\mathsf{ct} := \boldsymbol{S}^{\mathrm{T}}\boldsymbol{A}_{\setminus i} + \boldsymbol{E}_{\setminus i} \in \mathbb{Z}_q^{\ell\times(N+m-1)}.$$

$\mathsf{E}_2(\mathsf{hk}, (i, b), y; r_\mathsf{E})$:

- Compute $\boldsymbol{v} = \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{y} - b \cdot \boldsymbol{a}_i) + \boldsymbol{e}_i$, where $\boldsymbol{a}_i$ and $\boldsymbol{e}_i$ are the i-th rows of $\boldsymbol{A}$ and $\boldsymbol{E}$.
- Output $K := \mathsf{round}(\boldsymbol{v})$.

The randomness commonly used in the above two algorithm is $r_\mathsf{E} := (\boldsymbol{S}, \boldsymbol{E})$.

$\mathsf{D}\left(\mathsf{hk}, (x, r_\mathsf{H}), \mathsf{ct}\right)$:

- Compute $\boldsymbol{v}' = \mathsf{ct} \cdot [\boldsymbol{x}_{\setminus i}; \boldsymbol{r}]$.
- Output $K := \mathsf{round}(\boldsymbol{v}')$.

Size of Parameters The ciphertext space of this chameleon encryption is $\mathbb{Z}_q^{\ell\times(N+m)}$, where $q = 2^{\mathsf{poly}(\log\lambda)}$, $\ell = \mathcal{O}(1)$, $m = \mathcal{O}(n\log q) = \lambda\cdot\mathsf{poly}(\log\lambda)$. We set $N = \mathcal{O}(\lambda)$, i.e., when weak NCE is constructed from this chameleon encryption scheme, it can encrypt message of length $\mathcal{O}(\lambda)$. Thus the length of ciphertexts is

$$|\mathsf{ct}| = \mathsf{poly}(\log\lambda)\cdot\mathcal{O}(1)\cdot(\mathcal{O}(\lambda)+\lambda\cdot\mathsf{poly}(\log\lambda)) = \lambda\cdot\mathsf{poly}(\log\lambda).$$

The length of the hash key is

$$|\mathsf{hk}| = \mathsf{poly}(\log\lambda)\cdot\lambda\cdot(\mathcal{O}(\lambda)+\lambda\cdot\mathsf{poly}(\log\lambda)) = \lambda^2\cdot\mathsf{poly}(\log\lambda).$$

Theorem 4.6 (Correctness)**.** This LWE-based chameleon encryption scheme is correct.

Proof. Let $\Delta := \left|v_j - v'_j\right|$, where v_j and v'_j are the j-th component of the inputs to the rounding function in the computation of E_2 and D respectively.

$$\begin{aligned}
\Delta &= \left|\left(\boldsymbol{s}_j{}^{\mathrm{T}}(\boldsymbol{y} - x_i\cdot\boldsymbol{a}_i) + e_{i,j}\right) - \left(\mathsf{ct}_j\cdot[\boldsymbol{x}_{\setminus i};\boldsymbol{r}]\right)\right| \\
&= \left|\boldsymbol{s}_j{}^{\mathrm{T}}(\boldsymbol{A}\cdot[\boldsymbol{x};\boldsymbol{r}] - x_i\cdot\boldsymbol{a}_i) + e_{i,j} - \left(\boldsymbol{s}_j{}^{\mathrm{T}}\boldsymbol{A}_{\setminus i} + \boldsymbol{e}_{\setminus i,j}\right)[\boldsymbol{x}_{\setminus i};\boldsymbol{r}]\right| \\
&= \left|e_{i,j} - \boldsymbol{e}_{\setminus i,j}[\boldsymbol{x}_{\setminus i};\boldsymbol{r}]\right| \\
&\le \|\boldsymbol{e}_j\|\cdot\|[\boldsymbol{x};\boldsymbol{r}\| \\
&\le s\sqrt{N+m}\cdot\sqrt{N+s^2m} \le s^2(N+m),
\end{aligned}$$

holds with overwhelming probability. The probability of decryption error on j-th bit is bounded by

$$\Pr[\mathsf{round}(v_j) \neq \mathsf{round}(v'_j)] \le 2\Delta/q = \mathsf{negl}(\lambda)\,,$$

which is negligible since the modulus q is super-polynomially large. Thus, by taking the union bound for all $|\boldsymbol{v}| = \ell$ bits, the probability of decryption error is bounded by

$$\Pr[\mathsf{round}(\boldsymbol{v}) \neq \mathsf{round}(\boldsymbol{v}')] \le 2\ell\Delta/q = \mathsf{negl}(\lambda)\,.$$

□

Theorem 4.7. This obliviously samplable chameleon encryption scheme is secure assuming the hardness of the LWE problem.

Proof. First, we construct algorithms appear in the security definition.

$\mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$:

- Sample $\boldsymbol{R} \leftarrow \mathbb{Z}_q^{n\times N}$ and $(\boldsymbol{A}_T \in \mathbb{Z}_q^{n\times m}, \boldsymbol{T}) \leftarrow \mathsf{TrapGen}(1^\lambda)$.
- Sample $\boldsymbol{y} \leftarrow \mathbb{Z}_q^n$.
- Output $\mathsf{hk} \coloneqq \boldsymbol{A} = [\boldsymbol{R} \mid \boldsymbol{A}_T]$ and $\boldsymbol{y}$.

The randomness used in this algorithm is $r_{\mathsf{SimCH}} \coloneqq (\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{y})$.

$\mathsf{Open}_{\mathsf{CH}}(r_{\mathsf{SimCH}}, x)$:

- Set $\boldsymbol{y}' \coloneqq \boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}$.
- Using the lattice trapdoor, sample a short vector

$$\boldsymbol{r} \leftarrow \mathsf{Sample}(\boldsymbol{A}_T, \boldsymbol{T}, \boldsymbol{y}', s).$$

- Output $r_{\mathsf{G}} \coloneqq \boldsymbol{A}$ and $r_{\mathsf{H}} \coloneqq \boldsymbol{r}$.

$\widehat{\mathsf{E}_1}\left(\mathsf{hk}, (i, b); r_{\widehat{\mathsf{E}}}\right)$:

- Sample and output

$$\mathsf{ct} \leftarrow \mathbb{Z}_q^{\ell \times (N+m-1)}.$$

The randomness used in this algorithm is $r_{\widehat{\mathsf{E}}} = \mathsf{ct}$.

$\mathsf{Inv}_{\mathsf{CE}}(\mathsf{hk}, r_{\mathsf{E}})$:

- Recompute $\mathsf{ct} \coloneqq \boldsymbol{S}^{\mathrm{T}}\boldsymbol{A}_{\setminus i} + \boldsymbol{E}_{\setminus i} \in \mathbb{Z}_q^{\ell \times (N+m-1)}$.
- Output $r_{\widehat{\mathsf{E}}} \coloneqq \mathsf{ct}$.

Lemma 4.8 (Non-Committing Security of Hash)**.** The chameleon hash function in the above chameleon encryption scheme satisfies the security in Definition 4.4 unconditionally.

Proof. This directly follows form the properties of lattice trapdoor. First, $\boldsymbol{R}$ distributes uniformly at random, and the distribution of $\boldsymbol{A}_T$ output by $\mathsf{TrapGen}(1^\lambda)$ is also statistically close to uniform. Thus the entire hash key generated by the simulator is statistically indistinguishable from uniformly random matrix. This is the same distribution as the hash key generated by the key generation algorithm.

Second, the opened randomness for the hash r_{H} for $\boldsymbol{x}$, $\mathsf{H}(\mathsf{hk}, \boldsymbol{x}; r_{\mathsf{H}}) = \boldsymbol{y}$ holds, because the lattice trapdoor samples $\boldsymbol{r}$ such that $\boldsymbol{A}_T\boldsymbol{r} \equiv \boldsymbol{y}'$ (mod q) where $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}$ mod q. Moreover the distribution of $\boldsymbol{r}$ is statistically close to χ^m conditioned on $\boldsymbol{y} \equiv \boldsymbol{R}\boldsymbol{x} + \boldsymbol{A}_T\boldsymbol{r}$ (mod q). □

Lemma 4.9 (Oblivious Samplability of Ciphertext)**.** Assume the LWE problem is hard, the associated encryption part of the chameleon encryption scheme satisfies the oblivious samplability (Definition 4.5).

Proof. Let $\mathcal{A}$ be an adversary that attacks the oblivious samplability of the chameleon encryption. We construct a reduction algorithm $\mathcal{A}'$ that breaks the LWE assumption with $(N + m)$ samples by using $\mathcal{A}$ as follows:

1. $\mathcal{A}'$ receives $\left(\boldsymbol{A} = [\boldsymbol{R} \mid \boldsymbol{A}_T] \in \mathbb{Z}_q^{n \times (N+m)}, \boldsymbol{B} \in \mathbb{Z}_q^{\ell \times (N+m)}\right)$, where $\boldsymbol{B}$ is either $\boldsymbol{S}^{\mathrm{T}}\boldsymbol{A} + \boldsymbol{E}$ or uniformly random.

2. $\mathcal{A}'$ sets

$$\begin{aligned}\boldsymbol{a}' &\coloneqq (2x_i - 1)\left(\boldsymbol{a}_i - \boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}]\right), \\ \boldsymbol{R}' &\coloneqq [\boldsymbol{a}_1 \mid \cdots \mid \boldsymbol{a}_{i-1} \mid \boldsymbol{a}' \mid \boldsymbol{a}_{i+1} \mid \cdots \mid \boldsymbol{a}_N].\end{aligned}$$

   Remind that the matrix $\boldsymbol{R}'$ is the same as the received matrix $\boldsymbol{R}$ except for the i-th column is replaced by vector $\boldsymbol{a}'$. Furthermore, $\mathcal{A}'$ set s

$$\begin{aligned}\mathsf{hk} \coloneqq [\boldsymbol{R}' \mid \boldsymbol{A}_T], \boldsymbol{y} \coloneqq \mathsf{hk} \cdot [\boldsymbol{x}; \boldsymbol{r}], \mathsf{ct} \coloneqq \boldsymbol{B}_{\setminus i}, \\ K \coloneqq \mathsf{round}(\boldsymbol{b}_i), \text{ and } r_{\widehat{\mathsf{E}}} \coloneqq \mathsf{ct}.\end{aligned}$$

3. Finally, $\mathcal{A}'$ returns $\mathcal{A}(\mathsf{hk}, \boldsymbol{y}, \mathsf{ct}, K, r_{\widehat{\mathsf{E}}})$.

In the LWE case, that is, $\boldsymbol{B} = \boldsymbol{S}^{\mathrm{T}}\boldsymbol{A} + \boldsymbol{E}$ and $\boldsymbol{b}_i = \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a}_i + \boldsymbol{e}_i$, $\mathcal{A}'$ statistically simulates the real experiment of the oblivious samplability:

1. The hash key $\mathsf{hk} = [\boldsymbol{R}' \mid \boldsymbol{A}_T]$ generated by the reduction perfectly simulates the hash key since the only different element, i-th column of $\boldsymbol{R}'$ is also distributes uniformly random.

2. The distribution of ct is perfectly correct.

3. The distribution of $K = \mathsf{round}(\boldsymbol{b}_i)$ is also perfectly correct. This is because by our reduction algorithm, we have $\boldsymbol{y} = \mathsf{H}(\mathsf{hk}, \boldsymbol{x}; \boldsymbol{r}) = \mathsf{hk} \cdot [\boldsymbol{x}; \boldsymbol{r}] = \boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}] + x_i\boldsymbol{a}'$. Thus, in the computation of $K \leftarrow \mathsf{E}_2(\mathsf{hk}, (i, 1 - x_i), \boldsymbol{y}; r_{\mathsf{E}})$, we compute

$$\begin{aligned}\boldsymbol{v}_i &= \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{y} - (1 - x_i) \cdot \boldsymbol{a}') + \boldsymbol{e}_i \\ &= \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}] + x_i\boldsymbol{a}' - (1 - x_i) \cdot \boldsymbol{a}') + \boldsymbol{e}_i \\ &= \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}] + (2x_i - 1)\boldsymbol{a}') + \boldsymbol{e}_i \\ &= \boldsymbol{S}^{\mathrm{T}}\left(\boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}] + (2x_i - 1)(2x_i - 1)\left(\boldsymbol{a}_i - \boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}]\right)\right) + \boldsymbol{e}_i \\ &= \boldsymbol{S}^{\mathrm{T}}\left(\boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}] + \left(\boldsymbol{a}_i - \boldsymbol{A}_{\setminus i}[\boldsymbol{x}_{\setminus i}; \boldsymbol{r}]\right)\right) + \boldsymbol{e}_i \\ &= \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a}_i + \boldsymbol{e}_i = \boldsymbol{b}_i,\end{aligned}$$

   where we use the fact $(2x_i - 1)(2x_i - 1) = 1$ for $x_i \in \{0, 1\}$ to move forth line to fifth line. Therefore, $K = \mathsf{round}(\boldsymbol{v}_i) = \mathsf{round}(\boldsymbol{b}_i)$ has the correct distribution.

Also in the random case, $\mathcal{A}'$ perfectly simulates the ideal experiment of the oblivious samplability.

Therefore, assuming the LWE assumption, the associated encryption satisfies oblivious samplability. □

Combining the above two lemma, the proof of Theorem 4.7 completes. □

4.5 Construction of Weak NCE

In this section, we show a construction of weak NCE scheme $\mathtt{NCE}$ based on an obliviously samplable chameleon encryption scheme $\mathtt{CE}$. The constructed weak NCE scheme can encrypt n bit message. It satisfies weak security with respect to $\mathsf{Leak} = \mathsf{BEC}_{0.5}$. Decryption error of this scheme is ϵ, where we can set ϵ to be arbitrarily small constant by appropriately choosing the constant parameter ℓ; we require that $\epsilon \geq 1/2^{\ell+1} + \mathsf{negl}(\lambda)$. Its ciphertext expansion is constant, $2\ell + o(1)$.

This construction is similar to the construction from obliviously samplable KEM described in section 3.3.2. It is also similar to the construction of trapdoor function proposed by Garg and Hajiabadi [GH18].

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

- Sample a uniformly random string $z \leftarrow \{0,1\}^n$.
- Generate a hash key: $\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_{\mathsf{G}}\right)$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute and obliviously sample ciphertexts:

$$\mathsf{ct}_{i,b} \leftarrow \begin{cases} \mathsf{E}_1\left(\mathsf{hk}, (i,b); r_{\mathsf{E}i}\right) & (\text{if } b = z_i) \\ \widehat{\mathsf{E}}_1\left(\mathsf{hk}, (i,b); r_{\widehat{\mathsf{E}}i}\right) & (\text{otherwise}) \end{cases}.$$

- Output

$$pk := \left(\mathsf{hk}, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right) \quad \text{and} \quad sk := \left(z, \left(r_{\mathsf{E}1}, \ldots, r_{\mathsf{E}n}\right)\right). \tag{4.1}$$

The key generation randomness r_{Gen} is $\left(r_{\mathsf{G}}, z, \left\{r_{\mathsf{E}i}\right\}_{i\in[n]}, \left\{r_{\widehat{\mathsf{E}}i}\right\}_{i\in[n]}\right)$.

$\mathsf{Enc}(pk, x \in \{0,1\}^n; r_{\mathsf{Enc}})$:

- Parse public key pk as the equation 4.1.
- Compute the hash function: $y \leftarrow \mathsf{H}(\mathsf{hk}, x; r_{\mathsf{H}})$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute the decapsulation or sample uniformly random session keys:

$$K_{i,b} \leftarrow \begin{cases} \mathsf{D}\left(\mathsf{hk}, (x, r_{\mathsf{H}}), \mathsf{ct}_{i,b}\right) & (\text{if } b = x_i) \\ \{0,1\}^\ell & (\text{otherwise}) \end{cases}.$$

- Output

$$CT := \left(y, \begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}\right). \tag{4.2}$$

The encryption randomness r_{Enc} is $\left(r_{\mathsf{H}}, \{K_{i,1-x_i}\}_{i\in[n]}\right)$.

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

- Parse sk and CT as the equations 4.1 and 4.2, respectively.
- For all $i \in [n]$, compute

$$x_i := \begin{cases} z_i & (\text{if } K_{i,z_i} = \mathsf{E}_2\,(\mathsf{hk}, (i, z_i), y; r_{\mathsf{E}i})) \\ 1 - z_i & (\text{otherwise}) \end{cases}.$$

- Output x.

Ciphertext Expansion. Ciphertext length of this scheme is $|CT| = |y| + 2n\ell$, where length of output of the chameleon hash $|y|$ does not depend on n. Therefore ciphertext expansion of this scheme is

$$|CT|\,/n = 2\ell + o(1).$$

Public-key Expansion. Public-key length is $|\mathsf{hk}| + 2n\,|\mathsf{ct}|$.

Next, we show that NCE is a weak NCE scheme. Concretely, we show that NCE has ϵ-decryption error and satisfies weak security with respect to $\mathsf{BEC}_{0.5}$.

Theorem 4.10 (Weak Correctness)**.** Let ℓ be a constant noticeably larger than $\log(1/\epsilon) - 1$. If CE satisfies correctness, then NCE has ϵ-decryption error.

Proof. Let $x \in \{0,1\}^n$ be a message encrypted by NCE and $z \in \{0,1\}^n$ a random string sampled when generating a key pair of NCE.

We fail to decrypt x_i if the underlying chameleon encryption causes correctness error when $z_i = x_i$, or uniformly random ℓ bit string $K_{i,1-z_i}$ accidentally coincides with $\mathsf{E}_2\,(\mathsf{hk}, (i, z_i), y; r_{\mathsf{E}i})$ when $z_i \neq x_i$. The probability of the former is negligible since CE is correct, and that of the later is $1/2^\ell$. Thus, the probability of failure to decrypt x_i is evaluated as

$$\begin{aligned} &\Pr\left[x_i \neq (\mathsf{Dec}\,(sk, CT))_i\right] \\ &= \Pr\left[\begin{array}{c}\left(z_i = x_i \wedge \mathsf{D}\big(\mathsf{hk}, (x, r), \mathsf{ct}_{i,x_i}\big) \neq \mathsf{E}_2\big(\mathsf{hk}, (i, z_i), y; r_{\mathsf{E}i}\big)\right) \\ \vee \left(z_i \neq x_i \wedge K_{i,1-x_i} = \mathsf{E}_2\big(\mathsf{hk}, (i, z_i), y; r_{\mathsf{E}i}\big)\right)\end{array}\right] \\ &= \frac{1}{2}\left(\mathsf{negl}(\lambda) + \frac{1}{2^\ell}\right) \leq \epsilon\ . \end{aligned}$$

□

Theorem 4.11 (Weak Security)**.** If CE is an obliviously samplable CE scheme, then NCE is weakly secure with respect to $\mathsf{Leak} = \mathsf{BEC}_{0.5}$.

Proof. We construct a tuple of simulators as follows.

$\mathsf{SimGen}\left(1^\lambda; r_{\mathsf{SimGen}}\right)$:

- Generate $(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$.
- For all $i \in [n]$ and $b \in \{0,1\}$, compute ciphertexts: $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1\left(\mathsf{hk}, (i,b); r_{\mathsf{E}i,b}\right)$.
- Output a simulated public key $pk \coloneqq \left(\mathsf{hk}, \begin{pmatrix}\mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0}\\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1}\end{pmatrix}\right)$.

The randomness used in simulating public key r_{SimGen} is $\left(r_{\mathsf{SimCH}}, \{r_{\mathsf{E}i,b}\}_{i\in[n],b\in\{0,1\}}\right)$.

$\mathsf{SimEnc}(r_{\mathsf{SimGen}}, x_{\mathcal{I}} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}}); r_{\mathsf{SimEnc}})$:

- Regenerate (hk, y) from r_{SimGen}.
- For all $i \notin \mathcal{I}$, compute $K_{i,b} \leftarrow \mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_{\mathsf{E}i,b}\right)$ for $b \in \{0,1\}$.
- For all $i \in \mathcal{I}$, compute

$$K_{i,b} \leftarrow \begin{cases}\mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_{\mathsf{E}i,b}\right) & (\text{if } b = x_i)\\ \{0,1\}^\ell & (\text{otherwise})\end{cases}.$$

- Output a simulated ciphertext $CT \coloneqq \left(y, \begin{pmatrix}K_{1,0}, \ldots, K_{n,0}\\ K_{1,1}, \ldots, K_{n,1}\end{pmatrix}\right)$

The randomness used in simulating ciphertext r_{SimEnc} is $(\{K_{i,1-x_i}\}_{i\in\mathcal{I}})$. z

$\mathsf{Open}(r_{\mathsf{SimGen}}, r_{\mathsf{SimEnc}}, x, r_{\mathsf{ch}})$:

- Open randomness $(r_{\mathsf{G}}, r_{\mathsf{H}}) \leftarrow \mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$.
- Set $z = x \oplus 1^n \oplus r_{\mathsf{ch}}$.
- For all $i \in [n]$, invert randomness to oblivious sampling $r_{\widehat{\mathsf{E}}i,1-z_i} \leftarrow \mathsf{Inv}_{\mathtt{CE}}(\mathsf{hk}, r_{\mathsf{E}i,1-z_i})$.
- Output simulated randomness

$$r_{\mathsf{Gen}} \coloneqq \left(r_{\mathsf{G}}, z, \left\{r_{\mathsf{E}i,z_i}\right\}_{i\in[n]}, \left\{r_{\widehat{\mathsf{E}}i,1-z_i}\right\}_{i\in[n]}\right) \quad \text{and} \quad r_{\mathsf{Enc}} \coloneqq \left(r_{\mathsf{H}}, \{K_{i,1-x_i}\}_{i\in[n]}\right).$$

Let $\mathcal{A}$ be a PPT adversary against weak security of $\mathtt{NCE}$ and $x \in \{0,1\}^n$. We define the following sequence of experiments.

Exp 0: This experiment is exactly the same as $\mathsf{Exp}^{\text{Weak Real}}_{\mathtt{NCE},\mathcal{A}}$. Specifically, the experiment proceeds as follows.

1. Generate $\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_{\mathsf{G}}\right)$ and $z \leftarrow \{0,1\}^n$.
2. For all $i \in [n]$ and $b \in \{0,1\}$, compute

$$\mathsf{ct}_{i,b} \leftarrow \begin{cases}\mathsf{E}_1\left(\mathsf{hk}, (i,b); r_{\mathsf{E}_i}\right) & (\text{if } b = z_i)\\ \widehat{\mathsf{E}_1}(\mathsf{hk}, (i,b); r_{\widehat{\mathsf{E}}_i}) & (\text{otherwise})\end{cases}.$$

3. Set

$$pk \coloneqq \left(\mathsf{hk}, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right) \quad \text{and} \quad r_{\mathsf{Gen}} \coloneqq \left(r_{\mathsf{G}}, z, \{r_{\mathsf{E}i}\}_{i\in[n]}, \{r_{\widehat{\mathsf{E}}i}\}_{i\in[n]}\right).$$

4. Compute $y \leftarrow \mathsf{H}(\mathsf{hk}, x; r_{\mathsf{H}})$.
5. For all $i \in [n]$ and $b \in \{0,1\}$, compute

$$K_{i,b} \leftarrow \begin{cases} \mathsf{D}\left(\mathsf{hk}, (x,r), \mathsf{ct}_{i,b}\right) & (\text{if } b = x_i) \\ \{0,1\}^{\ell} & (\text{otherwise}). \end{cases}$$

6. Set

$$CT \coloneqq \left(y, \begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}\right) \quad \text{and} \quad r_{\mathsf{Enc}} \coloneqq \left(r_{\mathsf{H}}, \{K_{i,1-x_i}\}_{i\in[n]}\right).$$

7. Output of this experiment is $\mathsf{out} \leftarrow \mathcal{A}(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}})$.

Exp 1: In this experiment, instead of sampling $z \leftarrow \{0,1\}^n$, we first compute $x_{\mathcal{I}} \leftarrow \mathsf{BEC}_{0.5}(x; r_{\mathsf{ch}})$ and set $z = x \oplus 1^n \oplus r_{\mathsf{ch}}$.

Notice that z distributes uniformly at random over $\{0,1\}^n$ also in Exp 1 since $r_{\mathsf{ch}} \leftarrow \mathcal{B}_{0.5}^n$. Thus, $\Pr[\mathsf{out} = 1]$ in Exp 1 is identical to that in Exp 0. Also notice that $i \in \mathcal{I}$ iff $z_i \neq x_i$ holds by the setting of z.

Exp 2: This experiment is the same as Exp 1 except for executing $(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$ and $(r_{\mathsf{G}}, r_{\mathsf{H}}) \leftarrow \mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$ instead of $\mathsf{hk} \leftarrow \mathsf{G}\left(1^\lambda; r_{\mathsf{G}}\right), y \leftarrow \mathsf{H}\left(\mathsf{hk}, x; r_{\mathsf{H}}\right)$.

From the security of the chameleon hash in $\mathtt{CE}$, the difference of $\Pr[\mathsf{out} = 1]$ between Exp 1 and Exp 2 is negligible.

In the following experiments, we eliminate information of x_i for $i \notin \mathcal{I}$ from the ciphertext $CT = (y, \{K_{i,b}\}_{i\in[n],b\in\{0,1\}})$.

Exp 3.j: This experiment is defined for $j = 0, \ldots, n$. Exp 3.j is the same experiment as Exp 2 except that we modify the procedures 2. and 5. as follows.

2. For all $i \leq j$, compute $\mathsf{ct}_{i,b}$ for $b \in \{0,1\}$ as $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1\left(\mathsf{hk}, (i,b); r_{\mathsf{E}i,b}\right)$. Then, execute invert sampling as $r_{\widehat{\mathsf{E}}i} \leftarrow \mathsf{Inv}_{\mathtt{CE}}(\mathsf{hk}, r_{\mathsf{E}i,1-z_i})$.
   For all $i > j$, compute them in the same way as Exp 2.
5. For all $i \leq j$, if $i \notin \mathcal{I}$, compute $K_{i,0}, K_{i,1}$ as $K_{i,x_i} \leftarrow \mathsf{D}\left(\mathsf{hk}, (x, r_{\mathsf{H}}), \mathsf{ct}_{i,x_i}\right)$ and $K_{i,1-x_i} \leftarrow \mathsf{E}_2\left(\mathsf{hk}, (i, 1-x_i), y; r_{\mathsf{E}i,1-x_i}\right)$.
   For all $i \leq j$, if $i \in \mathcal{I}$, compute them in the same way as Exp 2.
   Also, for all $i > j$, compute them in the same way as Exp 2 regardless of whether $i \in \mathcal{I}$ or not.

Note that Exp 3.0 is exactly the same as Exp 2.

Lemma 4.12. If $\mathtt{CE}$ satisfies security with oblivious samplability, the difference of $\Pr[\mathsf{out} = 1]$ between Exp 3.$(j-1)$ and Exp 3.j is negligible for every $j \in [n]$.

Proof. Using $\mathcal{A}$, we construct a reduction algorithm $\mathcal{A}'$ which attacks the oblivious samplability of associated encryption in $\mathtt{CE}$ with respect to x and j.

What differ in Exp 3.$(j-1)$ and Exp 3.j are $\mathsf{ct}_{j,1-z_j}$ and $K_{j,1-x_j}$ if $j \notin \mathcal{I}$. For $j \in \mathcal{I}$, only $\mathsf{ct}_{j,1-z_j}$ differ. We consider the following two cases.

Case 1. $j \notin \mathcal{I}$ $(z_j = x_j)$: $\mathsf{ct}_{j,1-z_j}$ is output of $\widehat{\mathsf{E}_1}\left(\mathsf{hk}, (j, 1-z_j); r_{\widehat{\mathsf{E}}j}\right)$ in Exp 3.$(j-1)$ or $\mathsf{E}_1\left(\mathsf{hk}, (j, 1-z_j); r_{\mathsf{E}j,1-z_j}\right)$ in Exp 3.j. $K_{j,1-x_j}$ is uniformly random string or output of $\mathsf{E}_2\left(\mathsf{hk}, y; r_{\mathsf{E}i,1-x_j}\right)$, respectively. In this case, the reduction algorithm $\mathcal{A}'$, given $(\mathsf{hk}^*, y^*, \mathsf{ct}^*, K^*, r^*_{\widehat{\mathsf{E}}})$, embed $\mathsf{hk} = \mathsf{hk}^*, y = y^*, \mathsf{ct}_{j,1-z_j} = \mathsf{ct}^*$, $K_{j,1-x_j} = K^*, r_{\widehat{\mathsf{E}}j} = r^*_{\widehat{\mathsf{E}}}$ and simulate other parts of the experiment.

Case 2. $j \in \mathcal{I}$ $(z_j \neq x_j)$: $\mathsf{ct}_{j,1-z_j}$ is output of either $\widehat{\mathsf{E}_1}\left(\mathsf{hk}, (j, 1-z_j); r_{\widehat{\mathsf{E}}j}\right)$ in Exp 3.$(j-1)$ or $\mathsf{E}_1\left(\mathsf{hk}, (j, 1-z_j); r_{\mathsf{E}j,1-z_j}\right)$ in Exp 3.j. $K_{j,x_j} = \mathsf{D}(\mathsf{hk}, (x, r_{\mathsf{H}}), \mathsf{ct}_{j,x_j})$ is computed in the same way in both experiments.

In this case, the reduction algorithm $\mathcal{A}'$, given $(\mathsf{hk}^*, y^*, \mathsf{ct}^*, K^*, r^*_{\widehat{\mathsf{E}}})$, embed $\mathsf{hk} = \mathsf{hk}^*, y = y^*, \mathsf{ct}_{j,1-z_j} = \mathsf{ct}^*, r_{\widehat{\mathsf{E}}j} = r^*_{\widehat{\mathsf{E}}}$ and simulate other parts of the experiment.

In both cases, $\mathcal{A}'$ returns output $\mathsf{out} \leftarrow \mathcal{A}(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}})$.

Depending on $\mathcal{A}'$ playing in either $\mathsf{Exp}^{\mathrm{Real}}$ or $\mathsf{Exp}^{\mathrm{Ideal}}$, $\mathcal{A}'$ perfectly simulates Exp 3.$(j-1)$ or Exp 3.j for $\mathcal{A}$.

Hence assuming the associated encryption of the chameleon encryption scheme satisfies security, the difference of $\Pr[\mathsf{out} = 1]$ in Exp 3.$(j-1)$ and Exp 3.j is negligible.

□

Exp 4: This experiment is the same as Exp 3.n except that K_{i,x_i} is generated by $K_{i,x_i} \leftarrow \mathsf{E}_2\left(\mathsf{hk}, (i, x_i), y; r_{\mathsf{E}i,x_i}\right)$ instead of $K_{i,x_i} \leftarrow \mathsf{D}\left(\mathsf{hk}, (x, r_{\mathsf{H}}), \mathsf{ct}_{i,x_i}\right)$ for every $i \in [n]$.

From the correctness of chameleon encryption scheme, the difference of $\Pr[\mathsf{out} = 1]$ between Exp 3.n and Exp 4 is negligible.

This experiment is exactly the same as $\mathsf{Exp}^{\mathrm{Weak\ Ideal}}_{\mathtt{NCE},\mathcal{A}}$in which $\mathsf{Leak} = \mathsf{BSC}_{0.5}$ is used. In detail, the experiment proceeds as follows.

1. Generate $(\mathsf{hk}, y) \leftarrow \mathsf{SimCH}(1^\lambda; r_{\mathsf{SimCH}})$.
2. For all $i \in [n], b \in \{0,1\}$, compute $\mathsf{ct}_{i,b} \leftarrow \mathsf{E}_1\left(\mathsf{hk}, (i, b); r_{\mathsf{E}i,b}\right)$.
3. Set
$$pk := \left(\mathsf{hk}, \begin{pmatrix} \mathsf{ct}_{1,0}, \ldots, \mathsf{ct}_{n,0} \\ \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1} \end{pmatrix}\right).$$

4. For all $i \notin \mathcal{I}$, compute $K_{i,b} \leftarrow \mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_{\mathsf{E}i,b}\right)$ for $b \in \{0,1\}$.

5. For all $i \in \mathcal{I}$, compute

$$K_{i,b} \leftarrow \begin{cases} \mathsf{E}_2\left(\mathsf{hk}, (i,b), y; r_{\mathsf{E}i,b}\right) & (\text{if } b = x_i) \\ \{0,1\}^{\ell} & (\text{otherwise}) \end{cases}.$$

6. Set

$$CT \coloneqq \left(y, \begin{pmatrix} K_{1,0}, \ldots, K_{n,0} \\ K_{1,1}, \ldots, K_{n,1} \end{pmatrix}\right).$$

   Note that this CT can be computed only from $x_{\mathcal{I}}$, where $\mathcal{I} = \{i \in [n] \mid z_i \neq x_i\}$.

7. Compute $(r_{\mathsf{G}}, r_{\mathsf{E}}) \leftarrow \mathsf{Open}_{\mathtt{CH}}(r_{\mathsf{SimCH}}, x)$.

8. Set $z = x \oplus 1^n \oplus r_{\mathsf{ch}}$.

9. Set the randomness as

$$r_{\mathsf{Gen}} \coloneqq \left(r_{\mathsf{G}}, z, \left\{r_{\mathsf{E}i,z_i}\right\}_{i\in[n]}, \left\{r_{\widehat{\mathsf{E}}i}\right\}_{i\in[n]}\right)$$
$$r_{\mathsf{Enc}} \coloneqq \left(r_{\mathsf{H}}, \left\{K_{i,1-x_i}\right\}_{i\in[n]}\right).$$

10. $\mathsf{out} \leftarrow \mathcal{A}(pk, CT, r_{\mathsf{Gen}}, r_{\mathsf{Enc}})$

From the above arguments, we see that `NCE` satisfies weak security with respect to $\mathsf{Leak} = \mathsf{BSC}_{0.5}$. This completes the proof of Theorem 4.11.

□

4.6 Size and Expansion of the NCE Schemes

Finally, we summarize up ciphertext and public-key expansion of the proposed NCE scheme. Remind that the expansion of ciphertext and public-key is the length of them per message length for enough long messages.

Ciphertext Expansion The ciphertext expansion of the amplified NCE scheme is

$$\frac{\text{ciphertext expansion of weak NCE}}{\text{rate of wiretap codes}} + 1,$$

where the rate of the wiretap codes is constant. We need "+1" in hybrid encryption with one-time pad, which makes NCE satisfy non-committing security for public-key dependent messages.

The ciphertext expansion of the weak NCE scheme is,

$$|CT| / n = \frac{|y|}{n} + 2\ell,$$

where $n = \Theta(|m|)$ is length of the code word.

When instantiated based on the DDH, the size of hash $|y| = \lambda$. Thus the ciphertext expansion is constant for messages longer than $|m| = \Omega(\lambda)$

When instantiated based on the LWE, the size of hash $|y| = \lambda\mathsf{poly}(\log\lambda)$. Thus the ciphertext expansion is constant for messages longer than $|m| = \Omega(\lambda\mathsf{poly}(\log\lambda))$.

Now we estimate concrete ciphertext expansion of the resulting NCE scheme for messages longer than $|m| = \tilde{\omega}(\lambda)$. For such a long enough message, ciphertext expansion of the weak NCE scheme approaches 2ℓ. Suppose the wiretap codes used in the amplification achieve the secrecy rate

$$1/2 - h_2(\epsilon) = \frac{1}{2} + \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon),$$

where ϵ is error rate of each message bit $\frac{1}{2^{\ell+1}}$. The ciphertext expansion of the resulting NCE scheme has a minimum value appropriately 27 when $\ell = 5$.

Public-Key Expansion Public-key expansion of the amplified NCE scheme is

$$\frac{|p| + |\mathsf{hk}| + 2n\,|\mathsf{ct}|}{|m|},$$

where the length of public seed for the wiretap codes is $|p| = \mathcal{O}(\lambda)$. When instantiated from the DDH-based chameleon encryption scheme, the hash key has length $|\mathsf{hk}| = (2n+1)\lambda$ and the ciphertext length $|\mathsf{ct}| = (2n+1)\lambda$. The resulting public-key expansion is $\mathcal{O}(\lambda^2)$.

When instantiated from the LWE-based chameleon encryption scheme, the hash key has length $|\mathsf{hk}| = \lambda^2 \cdot \mathsf{poly}(\log\lambda)$ and the ciphertext length $|\mathsf{ct}| = \lambda \cdot \mathsf{poly}(\log\lambda)$. The resulting public-key expansion is $\lambda \cdot \mathsf{poly}(\log\lambda)$.

Note that these lengths of the hash key and ciphertext of the chameleon encryption schemes are evaluated for messages length $|m| = \mathcal{O}(\lambda)$. If we need to construct an NCE scheme for messages of length $\omega(\lambda)$, we first divide the message into λ bit blocks, then encrypt each block using the NCE scheme for λ bit message.

Expansion Trade-Off between Public-key and Ciphertext The public-key expansion of the DDH-based NCE scheme is $\mathcal{O}(\lambda^2)$. We can reduce it at expense of larger ciphertext expansion. In concrete, we set message length to $|m| = \lambda^c$ for constant $0 \le c \le 1$. Then the ciphertext expansion becomes $\mathcal{O}\big(\lambda^{(1-c)}\big)$ and public-key expansion $\mathcal{O}\big(\lambda^{(1+c)}\big)$. This trade-off technique allows us to freely bridge the gap between the schemes with constant ciphertext expansion and $\mathcal{O}(\lambda)$ ciphertext expansion.

Chapter 5

Conclusion

In this thesis, we constructed NCE schemes with constant ciphertext expansion based on the DDH or LWE problem.

In Chapter 2, we observe that the target of this thesis, non-committing security, and its building block, oblivious samplability can be defined in a similar manner through the focus on the randomness used by algorithms and simulators.

In Chapter 3, we proposed an NCE scheme with $\mathcal{O}(\lambda)$ ciphertext expansion which is constructed from obliviously samplable KEM. Along the way, we defined weak NCE. Given that the full-fledged NCE is a tool to establish private channels in adaptively secure MPC, weak NCE can be interpreted as a tool to establish wiretap channels in adaptively secure MPC. Through wiretap channels, we can securely transmit a message by encoding with wiretap codes that satisfy conditional invertibility.

This construction contains a fundamental idea to realize NCE. We believe it will help us further understand non-committing encryption, oblivious samplability, and possibly other security notions related to randomness used by algorithms.

In Chapter 4, We proposed a weak NCE scheme that has constant ciphertext expansion, which is amplified to an NCE scheme with constant ciphertext expansion. This weak NCE is constructed from obliviously samplable chameleon encryption. This thesis aims to find a suitable form of definition for chameleon encryption. As a result, the security definition of chameleon encryption can be regarded as a combination of non-committing security of hash function and oblivious samplability of the associated encryption. We believe this definition makes it easy to understand what security notion is essentially required in the construction of NCE schemes.

We also showed the public-key expansion of our NCE scheme can be reduced to $\lambda \cdot \mathsf{poly}(\log \lambda)$ if it is instantiated from the LWE problem. One may think that the use of the ring-LWE problem may further reduce public-key expansion similar to the LWE-based NCE scheme by Hemenway et al. [HORR16]. However, unfortunately, it seems that the ring-LWE problem is not helpful to reduce the public-key size asymptotically. Constructing an NCE scheme with constant ciphertext expansion and better public-key expansion is a natural future direction.

References

[Ari09] Erdal Arikan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory*, 55(7):3051–3073, 2009.

[Bac88] Eric Bach. How to generate factored random numbers. *SIAM J. Comput.*, 17(2):179–193, 1988.

[BBD+20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 58–87, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[Bea97] Donald Beaver. Plug and play encryption. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 75–89, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BH93] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323, Balatonfüred, Hungary, May 24–28, 1993. Springer, Heidelberg, Germany.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in*

Computer Science, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[BT12] Mihir Bellare and Stefano Tessaro. Polynomial-time, semantically-secure encryption achieving the secrecy capacity. Cryptology ePrint Archive, Report 2012/022, 2012. https://eprint.iacr.org/2012/022.

[BTV12a] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. A cryptographic treatment of the wiretap channel. Cryptology ePrint Archive, Report 2012/015, 2012. https://eprint.iacr.org/2012/015.

[BTV12b] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[CDD+15] Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 313–336, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CDMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively se-

cure protocols. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 287–302, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[CLNS17] Jan Camenisch, Anja Lehmann, Gregory Neven, and Kai Samelin. Uc-secure non-interactive public-key encryption. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 217–233. IEEE Computer Society, 2017.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[CPR17] Ran Canetti, Oxana Poburinnaya, and Mariana Raykova. Optimal-rate non-committing encryption. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 212–241, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes*

in Computer Science, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 362–391, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[GKRS20] Siyao Guo, Pritish Kamath, Alon Rosen, and Katerina Sotiraki. Limits on the efficiency of (ring) LWE based non-interactive key exchange. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 374–395, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

[GM06] Andrew Granville and Greg Martin. Prime number races. *Am. Math. Mon.*, 113(1):1–33, 2006.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 515–544, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[GS18a] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 535–565, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[GS18b] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[HOR15] Brett Hemenway, Rafail Ostrovsky, and Alon Rosen. Non-committing encryption from Φ-hiding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 591–608, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[HORR16] Brett Hemenway, Rafail Ostrovsky, Silas Richelson, and Alon Rosen. Adaptive security with quasi-optimal rate. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 525–541, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[HR05] Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 478–493, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *Advances in Cryptology –*

CRYPTO'98, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[Kal03] Adam Kalai. Generating random factored numbers, easily. *Journal of Cryptology*, 16(4):287–289, September 2003.

[LCC06] Feiyu Lei, Wen Chen, and Kefei Chen. A non-committing encryption scheme based on quadratic residue. In Albert Levi, Erkay Savas, Hüsnü Yenigün, Selim Balcisoy, and Yücel Saygin, editors, *Computer and Information Sciences - ISCIS 2006, 21th International Symposium, Istanbul, Turkey, November 1-3, 2006, Proceedings*, volume 4263 of *Lecture Notes in Computer Science*, pages 972–980. Springer, 2006.

[Leu77] Sik K. Leung-Yan-Cheong. On a special class of wiretap channels (corresp.). *IEEE Trans. Inf. Theory*, 23(5):625–627, 1977.

[LT13] Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 503–519, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[Nie01] Jesper Buus Nielsen. Non-committing encryption is too easy in the random oracle model. *BRICS Report Series*, 8(47), 2001.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RS94] Michael Rubinstein and Peter Sarnak. Chebyshev's bias. *Experimental Mathematics*, 3(3):173–197, 1994.

[Wyn75] Aaron D Wyner. The wire-tap channel. *Bell system technical journal*, 54(8):1355–1387, 1975.

[YKT19] Yusuke Yoshida, Fuyuki Kitagawa, and Keisuke Tanaka. Non-committing encryption with quasi-optimal ciphertext-rate based on the DDH problem. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 128–158, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

[YKXT20] Yusuke Yoshida, Fuyuki Kitagawa, Keita Xagawa, and Keisuke Tanaka. Non-committing encryption with constant ciphertext expansion from standard assumptions. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 36–65, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

[ZANS10] Huafei Zhu, Tadashi Araragi, Takashi Nishide, and Kouichi Sakurai. Adaptive and composable non-committing encryptions. In Ron Steinfeld and Philip Hawkes, editors, *Information Security and Privacy - 15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Proceedings*, volume 6168 of *Lecture Notes in Computer Science*, pages 135–144. Springer, 2010.