Lepton [1]: Key Encapsulation Mechanisms from a variant of Learning Parity with Noise

Name: Lepton

Principal submitters: Yu Yu, Shanghai Jiao Tong University, China
Jiang Zhang, State Key Laboratory of Cryptology, China

Inventors: Yu Yu, Shanghai Jiao Tong University, China
Jiang Zhang, State Key Laboratory of Cryptology, China

Owners: Yu Yu, Shanghai Jiao Tong University, China
Jiang Zhang, State Key Laboratory of Cryptology, China

Contact information: Yu Yu & Jiang Zhang
Postal addresses: #4-2803, 688 South Xizang Rd, Shanghai, China 200011
P.O. Box 5159, Beijing, China 100878
E-mail addresses: yyuu@sjtu.edu.cn jiangzhang09@gmail.com
Contact numbers: +86-15000088966 +86-15110204521

Date: November 28, 2017

Signature:

[1] The design and analysis of the Lepton crypto-system [51] are based on preliminary results obtained in [52], which is currently in submission and will appear in IACR ePrint at an appropriate time.

Table of Contents

1 Introduction

Informally, the (search version of) Learning Parity with Noise (LPN) problem of size n refers to:

"given matrix $\mathbf{A}$ and vector $\mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathbf{e}$, find out the value of secret vector $\mathbf{x}$",

where $\mathbf{A}$ and $\mathbf{x}$ are respectively $q \times n$ Boolean matrix and n-bit column vector sampled uniformly at random, and $\mathbf{e}$ is a q-bit column vector drawn from a q-fold Bernoulli distribution of noise rate $0 < \mu < 1/2$, i.e., $\mathbf{A} \xleftarrow{\$} \mathbb{F}_2^{q \times n}$, $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^n$ and $\mathbf{e} \xleftarrow{\$} \mathcal{B}_\mu^q$ (see Section 2 for notations), and '$\cdot$' and '$+$' denote multiplication and modulo-2 addition respectively. Other parameters, such as noise rate $0 < \mu < 1/2$ and sample complexity $q \in \mathbb{N}$, are either constants or functions of the main security parameter n. The decisional version of LPN problem challenges to distinguish between

$$(\mathbf{A}, \mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathbf{e}) \text{ and } (\mathbf{A}, \mathbf{y}' \xleftarrow{\$} \mathbb{F}_2^q) \ .$$

It is known that the two versions of LPN are polynomially equivalent [9,36,5]. The search version corresponds to key recovery attacks whereas the decisional one is more convenient for constructing cryptographic schemes and making security reductions. Both are believed to be hard even for quantum algorithms.

Symmetric-key Cryptography from LPN. The first cryptographic application of LPN is lightweight authentication schemes (e.g. [31,35,36], see [1] for a more complete list). Recently, Kiltz et al. [38] and Dodis et al. [17] constructed randomized MACs from LPN, which implies a two-round authentication scheme with man-in-the-middle security. Lyubashevsky and Masny [43] gave a more efficient actively secure three-round authentication scheme from LPN and recently Cash, Kiltz, and Tessaro [14] reduced the round complexity to 2 rounds. Applebaum et al. [4] used LPN to construct efficient symmetric encryption schemes with certain key-dependent message (KDM) security. Jain et al. [33] constructed an efficient perfectly binding string commitment scheme from LPN. We refer to a recent survey [45] about cryptography from the LPN assumption.

Public-key Cryptography from Low-noise LPN. All aforementioned cryptographic applications reside in minicrypt[1] and they rely on the hardness of the standard LPN with constant noise[2] (i.e., μ is a constant independent of secret size n). Alekhnovich [2] constructed the first cryptomania application from the low-noise LPN, in particular, he showed that LPN with noise rate $O(1/\sqrt{n})$ implies CPA secure public-key encryption schemes, and more recently Döttling et al. [18] and Kiltz et al. [37] further showed that CCA secure public-key encryption schemes can be constructed from the hardness of low-noise LPN, i.e., for noise rate $\mu = O(1/\sqrt{n})$.

Hardness of LPN. The search LPN problem can be seen as the average-case analogue of the NP-complete problem "decoding random linear codes" from coding theory [7]. LPN has been also extensively studied in learning theory, and it was shown in [20] that an efficient algorithm for LPN would allow to learn several important function classes such as 2-DNF formulas, juntas, and any function with a sparse Fourier spectrum. Under a constant noise rate the BKW (Blum Kalai and Wasserman) algorithm [10] solves LPN with time/sample complexity $2^{O(n/\log n)}$. Lyubashevsky [42] gave a non-trivial variant of the BKW algorithm with sample complexity $q = n^{1+\epsilon}$ and time complexity $2^{O(n/\log\log n)}$. Under low-noise rate $\mu = O(1/\sqrt{n})$, the best algorithms [8,39,6,40] solve LPN with time complexity $2^{O(\mu n)}$. Recently Esser et al. [19] proposed the first quantum algorithm for LPN. More precise, they reduce the problem to a version to which the quantum Grover search can be applied and thus achieves a speedup no more than (actually much less than) quadratic. For example, Esser et al. [19] predicted 64-bit quantum and 80-bit classical security for $n = 2048$ and $\mu = 1/\sqrt{n}$. We refer to[19, Table 8] for a more comprehensive comparison between classical and quantum security. Therefore, quantum algorithms are not known to have

[1] minicrypt refers to Impagliazzo's [32] hypothetical world where one-way functions exist but public-key cryptography does not, and cryptomania is the more optimistic world where public-key cryptography and multiparty computation are possible.

[2] Constant-noise and low-noise describe the asymptotic magnitudes of noise rate. Concretely, one can think of constant-noise rates as 1/4, 1/8 or other moderate values, and low-noise rates as those equal to (or slightly less than) $1/\sqrt{n}$ for $n \geq 2^{10}$.

any advantages (up to a quadratic speedup) in solving LPN over classic ones, which makes LPN a good candidate for "post-quantum cryptography". Further, LPN enjoys simplicity and is more suited for weak-power devices (e.g., RFID tags) than the large-modulus analogue Learning with Errors (LWE) [46] as the many modular additions and multiplications in LWE would be simplified to AND and XOR gates in LPN.

Proposal outline. The rest of this proposal is organized as follows: Section 2 clarifies the symbols, notations, parameters, abbreviations and some basic definitions that appear in this proposal. Section 3 gives the rationale for the design of the proposed cryptosystem at the intuitive level. Section 4 presents the algorithms proposed in full details. Section 5 and Section 6 provide the security analysis and implementation/performance of the proposed schemes respectively with Known Answer Test values given in Section 7. Finally, Section 8 summarizes the pros and cons of the algorithms proposed. We refer to Appendix A, Appendix B and Appendix C for lemmata, proofs and definitions omitted in the main body.

2 Preliminaries

2.1 Symbols, Notations and Parameters

The following symbols, notations and parameters are used in this proposal.

$\mathbf{a}$, $\mathbf{b}$, ...	Bold lower-case letters denote column (Boolean) vectors or ring elements, which will be obvious from the context.
$\mathbf{a}^\mathsf{T}$, $\mathbf{b}^\mathsf{T}$, ...	Row vectors, i.e., transposed column vectors with T denoting transpose operation.
$\mathbf{A}$, $\mathbf{B}$	Bold upper-case letter denote Boolean matrix.
$\mathcal{S}$	Calligraphic letters denote sets.
$\lvert\mathbf{a}\rvert$	The Hamming weight of bit vector $\mathbf{a}$, i.e., the number of 1's in $\mathbf{a}$.
e	The natural constant 2.71828...
$\lvert\mathcal{S}\rvert$	The number of elements in a set $\mathcal{S}$.
$\mathbb{N}$	The set of positive integers $\{1, 2, 3, \ldots\}$.
n	$n \in \mathbb{N}$ is bit length of the secret in the (Ring-)(C)LPN problem.
κ	The length of the random seed to be expanded by function Samp. We set $\kappa = 256$.
m	$m \in \mathbb{N}$ is the degree of the second term of trinomial $g = X^n + X^m + 1$ $(n > m)$.
log	The binary logarithm, i.e., the logarithm to base 2.
1^n	The unary representation of $n \in \mathbb{N}$.
$\mathbb{F}_2$	The set {0,1} that forms the smallest Galois field, where addition and multiplication are logical XOR and logical AND respectively.
$\mathbb{F}_2^n$	$\underbrace{\mathbb{F}_2 \times \cdots \times \mathbb{F}_2}_{n}$
$\mathbb{F}_2^*$	$\mathbb{F}_2^1 \cup \mathbb{F}_2^2 \cup \ldots \cup \mathbb{F}_2^i \cup \ldots$
R	A quotient ring $R = \mathbb{F}_2[X]/(g)$, where g is a polynomial of degree n with coefficients from $\mathbb{F}_2$. Thus, a polynomial over R is often represented (and used interchangeably) with a bit vector over $\mathbb{F}_2^n$.
$+$	Addition over R or bit-wise XOR (which are equivalent to each other).
$-$	Subtraction over R or bit-wise XOR (which are equivalent to each other).
$\cdot$	Multiplication between matrix and vector or elements over R, which will be obvious from the context.
$\lVert$	Bit string concatenation operator.

$\mathcal{B}_\mu$	Bernoulli distribution with parameter $0 < \mu < 1/2$.
$\mathcal{B}_\mu^n$	The concatenation of n independent $\mathcal{B}_\mu$,*i.e.*, $\mathcal{B}_\mu^n \stackrel{\mathsf{def}}{=} \underbrace{\mathcal{B}_\mu \times \cdots \times \mathcal{B}_\mu}_{n}$.
$\lceil \cdot \rceil$	The ceiling function, $\lceil x \rceil$ is the smallest integer not less than x.
χ_μ^n	It denotes either the set of length-n-weight-μn strings, i.e., $\{\mathbf{s} \in \mathbb{F}_2^n : \lvert\mathbf{s}\rvert = \mu n \in \mathbb{N}\}$ or a uniform distribution over such a set, which will be obvious from the context.
k	$k = \mu n \in \mathbb{N}$ represents the weight in the length-n-weight-(μn) distribution χ_μ^n.
$\mathsf{REP}(\cdot,\cdot)$	$\mathsf{REP}(\cdot,\mathsf{RCN})$: $\mathbb{F}_2^z \to \mathbb{F}_2^{\ell=z\cdot\mathsf{RCN}}$ is a repetition code that repeats each bit RCN times: $\mathsf{REP}(m_1 \ldots m_z,\mathsf{RCN})=\underbrace{m_1 \ldots m_1}_{\mathsf{RCN}} \ldots \underbrace{m_z \ldots m_z}_{\mathsf{RCN}}$ and this proposal sets $z = 508$.
$\mathbf{x} \stackrel{\$}{\leftarrow} D$	Draw a value $\mathbf{x}$ according to distribution D.
$\mathbf{x} \stackrel{\$}{\leftarrow} \mathcal{S}$	Draw a value $\mathbf{x}$ from set $\mathcal{S}$ uniformly at random.
$\mathbf{a} \stackrel{\$}{\leftarrow} R$	Used interchangeably with $\mathbf{a} \stackrel{\$}{\leftarrow} \mathbb{F}_2^n$ with the emphasis that $\mathbf{a} \in R$.
$\mathbf{x} \stackrel{\$}{\leftarrow} \chi_\mu^R$	Used interchangeably with $\mathbf{x} \stackrel{\$}{\leftarrow} \chi_\mu^n$ with the emphasis that $\mathbf{x} \in R$.
$\mathsf{Trunc}(\cdot,\cdot)$	$\mathsf{Trunc}(\cdot,\ell) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^\ell$ is a truncating function that returns the first $\ell \leq n$ bits of its input (and discards the rest).
$\mathsf{Integer}(\cdot)$	$\mathsf{Integer}(\mathbf{r})$ converts a bit-string $\mathbf{r} \in \mathbb{F}_2^k$ into an unsigned integer in $\{0, \ldots, 2^k - 1\}$.
$:=$	Value assignment operator.
$\mathsf{mat}(\mathbf{a})$, $\mathsf{vec}(\mathbf{s})$	For ring elements $\mathbf{a}, \mathbf{s} \in R = \mathbb{F}_2[X]/(g)$, $\mathsf{mat}(\mathbf{a}) \in \mathbb{F}_2^{n\times n}$ and $\mathsf{vec}(\mathbf{s}) \in \mathbb{F}_2^{n\times 1}$ denote their respective square matrix and column vector representations such that $\mathsf{vec}(\mathbf{a}\cdot\mathbf{s}) \equiv \mathsf{mat}(\mathbf{a}) \cdot \mathsf{vec}(\mathbf{s})$, where for every $0 \leq i < n$ the $(i{+}1)$-th column of $\mathsf{mat}(\mathbf{a})$ corresponds to the binary representation $\mathbf{a} \cdot X^i \mod g$, and $\mathsf{vec}(\mathbf{s})$ and $\mathbf{s}$ are of the same representation and thus $\mathsf{vec}(\cdot)$ is often omitted .
$\mathsf{poly}(\cdot)$	A polynomial function.
$\mathsf{negl}(\cdot)$	A negligible function.

2.2 Definitions

1. A trinomial is a special polynomial with 3 terms.
2. A non-constant polynomial is irreducible over $\mathbb{F}_2$ if its coefficients belong to $\mathbb{F}_2$ and it cannot be factored into the product of two non-constant polynomials with coefficients in $\mathbb{F}_2$.
3. The (Hamming) weight of a bit string $\mathbf{s}$ is the number of 1's in $\mathbf{s}$.
4. A negligible function is a function $\mathsf{negl} : \mathbb{N}\to\mathbb{R}$ such that for every positive integer c there exists an integer N_c such that for all $n > N_c$ it holds that $\mathsf{negl}(n) < 1/n^{N_c}$.

2.3 Abbreviations

BCH	Bose-Chaudhuri-Hocquenghem
BKW	the Blum Kalai and Wasserman algorithm [10]
CLPN	Compact Learning Parity with Noise (Definition 1)
HNF	Hermite Normal Form
HNF-LPN	the Hermite Normal Form of LPN (Definition 5)
i.i.d.	independent and identically distributed
KAT	Known Answer Test
KEM	Key Encapsulation Mechanism
LPN	Learning Parity with Noise (Definition 4)
LSPN	Learning Sparse Parity with Noise
PKE	Public-Key Encryption
PPT	Probabilistic Polynomial-Time
Ring-CLPN	the Ring variant of CLPN (Definition 2)
Ring-LPN	the Ring variant of LPN (Definition 6)
ROM	Random Oracle Model
XOR	eXclusive-OR

2.4 Key Encapsulation Mechanism

Key Encapsulation Mechanism. A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with session key space $\mathcal{K}$ consists of the following efficient algorithms:

- Key Generation. A key generation algorithm $\mathsf{KeyGen}(\cdot)$ takes as input the security parameter in unary and outputs a pair of public key pk and secret key sk, denoted by $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)$.
- Encapsulation. An encapsulation algorithm $\mathsf{Encaps}(\cdot)$ takes as input a public key pk, and outputs a ciphertext C and a session key $K \in \mathcal{K}$, denoted by $(C, K) \xleftarrow{\$} \mathsf{Encaps}(pk)$.
- Decapsulation. A decapsulation algorithm $\mathsf{Decaps}(\cdot)$ takes as input a secret key sk and a ciphertext C, and outputs either a session key $K \in \mathcal{K}$ or a failure symbol $\perp$, denoted by $K / \perp \leftarrow \mathsf{Decaps}(sk, C)$.

Correctness. We say that a KEM is $(1-\delta)$-*correct* if it holds that

$$\Pr\left[\ (pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n),\ (C, K) \xleftarrow{\$} \mathsf{Encaps}(pk)\ :\ \mathsf{Decaps}(sk, C) = K\ \right] \geq 1 - \delta.$$

for some quantity δ negligible in n. A KEM is said to be *perfectly correct* if it is 1-correct.

2.5 Public-Key Encryption

Public-Key Encryption. A public-key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ consists of three efficient algorithms as below:

1. Key generation. A key generation algorithm $\mathsf{KeyGen}(\cdot)$ takes as input the security parameter n in unary, and outputs a pair of public key pk and secret key sk, which is denoted by $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)$.
2. Encryption. An encryption algorithm Enc takes as input a public key pk and a message $M \in \mathcal{M}$, and outputs a ciphertext C, which is denoted by $C \xleftarrow{\$} \mathsf{Enc}(pk, M)$.
3. Decryption. A decryption algorithm Dec takes as input a secret key sk and a ciphertext C, and outputs either a message $M \in \mathcal{M}$ or a failure symbol $\perp$, which is denoted by $M / \perp \leftarrow \mathsf{Dec}(sk, C)$.

CORRECTNESS. We say that PKE is $(1-\delta)$-*correct*, if for all messages $M \in \mathcal{M}$ it holds that

$$\Pr\left[\ (pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)\ :\ \mathsf{Dec}(sk, \mathsf{Enc}(pk, M)) = M\ \right] \geq 1-\delta.$$

for some δ negligible in n. A PKE is said to be *perfectly correct* if it is 1-correct (i.e., $\delta = 0$).

2.6 CLPN and Its Ring Variant

We formally define the Compact Learning Parity with Noise (CLPN) problem.

Definition 1 (Compact Learning Parity with Noise). *The decisional CLPN problem with secret length n and noise rate $0 < \mu < 1/2$, denoted by decisional (n,μ)-CLPN, is hard if for every PPT algorithm $\mathcal{D}$ we have*

$$\left|\Pr[\mathcal{D}(\mathbf{A},\ \mathbf{A}\cdot\mathbf{x}+\mathbf{e}) = 1] - \Pr[\mathcal{D}(\mathbf{A}, \mathbf{r}) = 1]\right| = \mathsf{negl}(n)\ , \tag{1}$$

and the computational LPN problem with the same n and μ, denoted by computational (n,μ)-LPN, is hard if for every $q = \mathsf{poly}(n)$ and every PPT algorithm $\mathcal{D}$ we have

$$\Pr[\ \mathcal{D}(\mathbf{A},\ \mathbf{A}\cdot\mathbf{x}+\mathbf{e}) = \mathbf{x}\ ]\ =\ \mathsf{negl}(n)\ , \tag{2}$$

where the square matrix $\mathbf{A} \xleftarrow{\$} \mathbb{F}_2^{n\times n}$ and vectors $\mathbf{x}, \mathbf{e} \xleftarrow{\$} \chi_\mu^n, \mathbf{r} \xleftarrow{\$} \mathbb{F}_2^n$.

Definition 2 (Ring-CLPN). *The ring version of the Compact Learning Parity with Noise (Ring-CLPN) problem is defined same as Definition 1 except that square matrix $\mathbf{A}$ is replaced with a ring element $\mathbf{a}$ and $\mathbf{a}\cdot\mathbf{x}$ is interpreted as the product of elements $\mathbf{a}$ and $\mathbf{x}$ over $R = \mathbb{F}_2[X]/(g)$ for polynomial g of degree n. Our designed crypto-systems reply on the (n,μ,2)-Ring-CLPN where the adversary gets to see two noisy elements, i.e., ($\mathbf{a}_1$,$\mathbf{a}_2$,$\mathbf{a}_1\mathbf{x}+\mathbf{e}_1$,$\mathbf{a}_2\mathbf{x}+\mathbf{e}_2$), where $\mathbf{a}_1, \mathbf{a}_2 \xleftarrow{\$} R$, $\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \chi_\mu^R$. Throughout we consider only rings whose g is an irreducible trinomial over $\mathbb{F}_2$.*

PARAMETERS. The CLPN and its ring version set the number of samples to $q = n$ and $q = 2n$ bits respectively and thus parameter q is implicit and quite minimal. Further, for PKE the noise rate $\mu = O(1/\sqrt{n})$ is typically used. Thus, the (Ring-)CLPN needs no explicit parameters other than the main parameter n. Unlike LPN which is insensitive to sample complexity, it is crucial for (Ring-)CLPN to use as minimal as possible $q = n$ (or $q = 2$ for its ring version) and to use structured noise distribution of size n, i.e., χ_μ^n (in case of Ring-CLPN, the $2n$-bit noise is sampled from two independent χ_μ^n). This is because when $q \gg n$ CLPN degenerates into LPN of roughly the same noise rate, i.e., for $q \gg n$ one can divided the noise of CLPN into n-bit blocks such that each block behaves very similar to a Bernoulli distribution $\mathcal{B}_\eta^n$ for $\eta \leq q\mu/(q-n+1) \approx \mu$ in terms of bounded errors and the mount of entropy (see Fact 1 and Fact 2).

CORRECTNESS. To construct a (perfectly) correct PKE, it is crucial to ensure that the product of two sparse ring elements results into another sparse one. We prove Lemma 1 below that improves upon the counterpart in [15, Lemma 2.23] by reducing the expansion factor from 7 to 3 (and even 2). The additional requirement $m \leq n/2$ on trinomial g is not a constraint due to the symmetry, i.e., if $g = X^n + X^m + 1$ is irreducible in $\mathbb{F}_2[X]$ then so is $g' = X^n + X^{n-m} + 1$. Refer to Appendix B for its full proof.

Lemma 1 (Lightness-Preserving Ring). *If $g = X^n + X^m + 1$ is an irreducible polynomial in $\mathbb{F}_2[X]$ for $m \leq n/2$, then the ring $R = \mathbb{F}_2[X]/(g)$ is lightness-preserving with expansion factor 3, i.e., for any polynomials $p, q \in \mathbb{F}_2[X]$ of degree up to $n-1$, the reduced polynomial of their product $r = p\cdot q$ has weight*

$$|\ p\cdot q \mod g\ |\ \leq\ \min\left(3|p|\cdot|q|,\ 2|p|\cdot|q| + m - 1\right)$$

where $|p|$ denotes the weight (i.e., the number of non-zero coefficients) of p.
***Tightness**: for $m > 1$ the equality is met for certain p,q, e.g., $p = q = X^{n-1}$.*

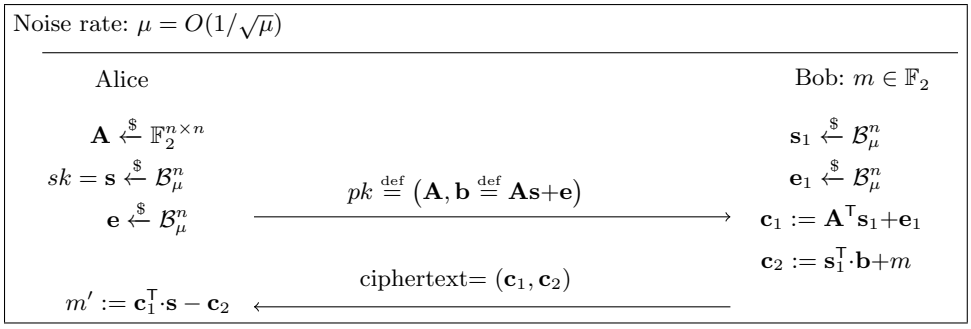

Fig. 1. A two-pass protocol by which Bob transmits a message bit m to Alice with passive security and noticeable correctness (for $\mu = O(1/\sqrt{n})$), where Bob receives $m' = m+(\mathbf{s}_1^\mathsf{T}\cdot\mathbf{e})+(\mathbf{e}_1^\mathsf{T}\cdot\mathbf{s})$.

3 Design Rationale

The folklore LPN-based PKE. Following Alekhnovich's blueprint, an IND-CPA PKE can be constructed from the HNF-LPN [4] (which is equivalent to LPN, see Lemma 5) where the secret $\mathbf{s}$ is also sampled from $\mathcal{B}_\mu^n$ instead of from uniform. We depict in Figure 1 a two-pass protocol by which Alice and Bob agree on a single bit with passive security, which is equivalent to an IND-CPA secure PKE for single bit messages. That is, Alice samples a pair of keys (pk,sk) and sends the public one pk to Bob, and Bob encrypts with pk his message m and returns the ciphertext, which is decrypted with sk to m'. Correctness requires the inner product of two noise vectors sampled from $\mathcal{B}_\mu^{2n}$ is noticeably biased to 0 (using Piling-up Lemma and Stirling's approximation), i.e.,

$$\Pr[m = m'] = \Pr\left[[\mathbf{s}_1^\mathsf{T}, \mathbf{e}_1^\mathsf{T}] \cdot \begin{bmatrix}\mathbf{e}\\ \mathbf{s}\end{bmatrix} = 0\right] \geq 1/2 + 2^{-O(\mu^2 n)}$$

and thus $\mu = O(1/\sqrt{n})$ is inherently essential. However, the lower noise rate makes the PKE scheme inefficient in terms of key and ciphertext lengths. The current best attack [40, Appendix C] on low-noise LPN runs in time $e^{\mu n}$ with an overwhelming success probability. As estimated in [15], an 80-bit classic secure LPN-based PKE already needs public keys of several megabytes. To improve efficiency, Damgard and Park [15] introduced an efficient variant of Ring-LPN[3] [26], called Transposed Ring-LPN, which results into a more efficient (yet still not practical) PKE of 128-bit classical security that encrypts 128-bit plaintext into 36-KB ciphertext with a public key of about 230kB. Finally, we mark that the recent constant-noise LPN-based PKE [50] is even less efficient than the Alekhnovich-style counterparts analyzed above due to its quasi-polynomial security and strong decryption errors, and is thus mainly of theoretical interests.

CLPN. Firstly we introduce the Compact Learning Parity with Noise (CLPN) problem where $\mathbf{A}$ is simply a square matrix (i.e., $q = n$), and secret $\mathbf{s}$ and noise $\mathbf{e}$ both follow a uniformly random length-n-weight-μn distribution, denoted by χ_μ^n. Substituting this variant into Figure 1 (and replacing $\mathcal{B}_\mu^n$ with χ_μ^n) we first need to guarantee the correctness, i.e., $\mathbf{s}_1^\mathsf{T}\mathbf{e} + \mathbf{e}_1^\mathsf{T}\mathbf{s}$ is a biased bit bounded away from 1/2. It suffices to show by Lemma 7 that

$$\Pr[\mathbf{s}_1^\mathsf{T}\mathbf{e} = 0] = \Pr[\mathbf{e}_1^\mathsf{T}\mathbf{s} = 0] \geq \frac{1}{2} + \frac{1}{2}(1 - \frac{2\mu n}{n - \mu n + 1})^{\mu n} \approx \frac{1 + e^{-2\mu^2 n}}{2} \ .$$

where we set $\mu = O(1/\sqrt{n})$ (which is the same as LPN for the same μ) for correctness. We stress that hereafter we consider only the low-noise regime for $\mu = 1/\sqrt{n}$ or less.

Why CLPN? The intuition to replace $\mathcal{B}_\mu^n$ with χ_μ^n is to further tighten up security and defend against state-of-the-art attacks [40,19]. Note that $\mathcal{B}_\mu^n$ is not a uniform distribution but more

[3] Informally, seen as a special variant of LPN, Ring-LPN parses matrix $\mathbf{A}$ into $n \times n$ matrices $\mathbf{A}_1, \cdots, \mathbf{A}_i, \cdots$, and instead of being random every $\mathbf{A}_i$ is succinctly described using a random element $\mathbf{a}_i$ over a ring $R = \mathbb{F}_2[X]/(g)$ for polynomial g of degree n, i.e., $\mathbf{A}_i = \mathsf{mat}(\mathbf{a}_i)$, such that $\mathbf{A}_i\mathbf{s} \equiv \mathbf{a}_i\mathbf{s}$.

prone to any lower-weight values than heavier ones, i.e., $\mu^k(1-\mu)^{n-k}$ decreases monotonically with respect to weight k. Thus, an attacker may start off with weak keys, e.g., he simply bets on $\mathbf{e}=0$ which occurs with probability $(1-\mu)^n \approx e^{-\mu n}$. Intuitively this explains why the current best attack [40, Appendix C] on low-noise LPN is of time complexity $e^{\mu n}$. Observing that $\mathcal{B}_\mu^n$ is a convex combination of uniform distributions of different weights, i.e., $\chi_0^n, \cdots, \chi_\mu^n, \cdots, \chi_1^n$, we keep only distribution χ_μ^n (recall that $\mu n \in \mathbb{N}$), where the lower and upper trails are cut off for security (i.e., to avoid weak key attacks) and correctness (as we will see later having an upper bound on the weight of noise vectors would even allow us to achieve perfect correctness) respectively. A simple reduction suggests that CLPN is at least as hard as LPN for the same μ (see Lemma 2). We further argue that CLPN turns out to be significantly harder. In fact, CLPN can be seen as a hybrid of the exact LPN [33] (whose noise follows χ_μ^q for a typically larger q) and the Learning Sparse Parity with Noise (LSPN) problem (where $\mathbf{s} \xleftarrow{\$} \chi_\mu^n$ and $\mathbf{e} \xleftarrow{\$} \mathcal{B}_\eta^q$ for $\mu = o(1)$ and $\eta \gg \mu$). Now that $\mathbf{s}$ and $\mathbf{e}$ are both from χ_μ^n, a brute force search on CLPN takes complexity $\binom{n}{\mu n} \approx n^{\mu n - O(\log n)}$ and the best known classical attacks [23] are of complexity $n^{(0.5+o(1))\mu n}$. We refer to Section 5 for more details about security analysis of CLPN. Compared with LPN, the hardness of CLPN is significantly strengthened, e.g., for a typical choice $n = 2^{15}$ and the same $\mu = O(1/\sqrt{n})$ the classical hardness of LPN is $\log(e) \cdot \mu n = 1.44\mu n$ bits and in contrast that of the CLPN is $(\log n/2)\mu n = 7.5\mu n$.

Ring-CLPN. In order to obtain more efficient constructions, ring heuristics are further applied to CLPN to get a ring variant we call Ring-CLPN (just like how Ring-LPN [26] relates to LPN). Informally, our search (resp., decisional) Ring-CLPN assumption postulates that it is hard to recover $\mathbf{s}$ given (resp., distinguish the following from uniform randomness)

$$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1\mathbf{s}+\mathbf{e}_1, \mathbf{a}_2\mathbf{s}+\mathbf{e}_2) \ ,$$

where $\mathbf{a}_1, \mathbf{a}_2 \xleftarrow{\$} \mathbb{F}_2^n$ and $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \chi_\mu^n$ are treated as elements over a ring $R = \mathbb{F}_2[X]/(g)$ for an irreducible polynomial g. One can compare the above with CLPN with $q = 2n$ samples, i.e., the $2n \times n$ matrix $\mathbf{A}$ is parsed as square submatrices matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, and instead of being uniform random each $\mathbf{A}_i$ is succinctly described using a random ring element $\mathbf{a}_i$, i.e., $\mathbf{A}_i = \mathsf{mat}(\mathbf{a}_i)$. As we will show later, this can yield very efficient constructions.

Irreducible g. The choice of an irreducible g provides a couple of good security-relevant properties when interpreting Ring-CLPN in matrix form (to be in line with CLPN):

1. **Linear independent** rows/columns. For any non-zero $\mathbf{a} \in R = \mathbb{F}_2[X]/(g)$ with irreducible g, square matrix $\mathbf{A} = \mathsf{mat}(\mathbf{a})$ always has full-rank. [4]
2. **Uniform** rows/columns. Every row/column of $\mathbf{A}$ is uniformly random if the underlying $\mathbf{a}$ is drawn uniformly at random.

Therefore, if CLPN is secure on a uniformly random matrix, then CLPN is secure with $\mathbf{A}$ sampled from most values from $\mathbb{F}_2^{n\times n}$. In this regard, Ring-CLPN postulates that the ring (field for irreducible g) representation of matrix constitutes a subset of those 'good' matrix values. Moreover, we give some non-trivial evidence tha Ring-CLPN could be as nearly secure as the CLPN problem (of a sightly lower noise rate) under reasonable assumptions. Furthermore, the use of irreducible polynomials prevents specific attacks [25,24] (originally proposed for Ring-LPN and Ring-LWE) utilizing the factorization of the underlying polynomials. To our best knowledge, the best way to solve our Ring-CLPN (or Field-LPN more precisely) is to apply generic attacks for CLPN and we refer to Section 5 for more details.

4 Description of the Key Encapsulation Mechanism

In this section, we give two KEM schemes from Ring-CLPN. The first one $\mathsf{Lepton.CPA} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ aims at achieving CPA-security by only assuming the computationally hardness of Ring-CLPN. The second one $\mathsf{Lepton.CCA} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ is a KEM

[4] Otherwise (i.e., if $\mathbf{A}$ has no full rank), there exists $\mathbf{x} \neq \mathbf{0}$ s.t. $\mathbf{Ax} = \mathbf{ax} = \mathbf{0}$, which is not possible for nonzero elements $\mathbf{a}$ and $\mathbf{x}$ over a field.

scheme for achieving CCA-security, which is obtained by applying the Fujisaki-Okamoto (FO) transform to Lepton.CPA. We begin by first introducing some building blocks. Let κ be the natural security parameter, and we use $\kappa = 256$ as the minimal length random seed for achieving a targeted security level.

- Let $g = X^n + X^m + 1$ be an irreducible polynomial function in $\mathbb{F}_2[X]$ of degree n and small integer m, and let $R = \mathbb{F}_2[X]/(g)$. It is naturally to identify R as $\mathbb{F}_2^n$.
- Let $\mathsf{Samp} : \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be an algorithm which expands length κ random bit strings into $\mathbb{F}_2^n$. Following the schemes in [3,11,12,13], we use Samp to compress public keys. Refer to Section 6.1 for the concrete instantiation.
- Let $G(\cdot, \ell) : \mathbb{F}_2^* \longrightarrow (\mathbb{F}_2^\kappa)^\ell$ be a hash function which takes as input a random source, and outputs ℓ length κ bit strings. We usually omit the parameters if $\ell = 1$. Refer to Section 6.1 for the concrete instantiation.
- Let $F(\cdot, \ell) : \mathbb{F}_2^\kappa \longrightarrow (\chi_\mu^n)^\ell$ be a sampling algorithm which takes as input a random source, and outputs ℓ noise vectors over χ_μ^n. Refer to Section 6.1 for the concrete instantiation.
- Let $\mathsf{Trunc}(\cdot, \ell) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^\ell$ be a function that on input $\mathbf{s}$ and simply truncates the n-bit $\mathbf{s}$ into one of $\ell \leq n$ bits by keeping the first ℓ bits of $\mathbf{s}$ (and discarding the rest).
- Let $\mathsf{ECC} : \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^\ell$ be an error correction code with an efficient decoding algorithm ECC^{-1}, which can correct at least an α-fraction of errors with $0 < \alpha < 1$ a constant.

4.1 Lepton.CPA

We define $\mathsf{Lepton.CPA} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with session key space $\mathcal{K} = \mathbb{F}_2^\kappa$ as follows.

Algorithm 1: $\mathsf{Lepton.CPA.KeyGen}(1^\kappa)$

$\eta \xleftarrow{\$} \mathbb{F}_2^\kappa$;
$(\rho, \sigma) := G(\eta, 2)$;
$(\mathbf{s}, \mathbf{e}) := F(\rho, 2)$;
$\mathbf{a} := \mathsf{Samp}(\sigma)$;
$\mathbf{b} := \mathbf{as} + \mathbf{e}$;
return $(pk, sk) = \big((\sigma, \mathbf{b}), \mathbf{s}\big)$;

Algorithm 2: $\mathsf{Lepton.CPA.Encaps}(pk = (\sigma, \mathbf{b}))$

$\mathbf{a} := \mathsf{Samp}(\sigma)$;
$\eta \xleftarrow{\$} \mathbb{F}_2^\kappa$;
$(m, r) := G(\eta \| pk, 2)$;
$(\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2) \sim (\chi_\mu^n)^3 := F(r, 3)$;
$\mathbf{c}_1 := \mathbf{ax} + \mathbf{e}_1$;
$\mathbf{c}_2 := \mathsf{Trunc}(\mathbf{bx} + \mathbf{e}_2, \ell) + \mathsf{ECC}(m)$;
$\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2)$;
$K := G(m \| \mathbf{C})$;
return $\big(\mathbf{C}, K\big)$;

Algorithm 3: $\mathsf{Lepton.CPA.Decaps}\big(sk = \mathbf{s}, \mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2)\big)$

$\tilde{\mathbf{c}}_0 := \mathbf{c}_2 - \mathsf{Trunc}(\mathbf{c}_1\mathbf{s}, \ell)$;
$m' := \mathsf{ECC}^{-1}(\tilde{\mathbf{c}}_0)$;
return $K := G(m' \| \mathbf{C})$;

(Perfect) correctness. It is crucial to ensure that the product of two sparse ring elements results into another sparse one. We prove Lemma 1 that for any trinomial $g = X^n + X^m + 1$ with $m \leq n/2$ and for any ring elements $p, q \in R = \mathbb{F}_2[X]/(g)$ it holds that $|p \cdot q| \leq |p| \cdot |q|$, which improves upon the counterpart results in [15, Lemma 2.23] by reducing the expansion factor from 7 to 3 (and even 2). Thanks to the use of χ_μ^n, the individual $\mathbf{s}$, $\mathbf{e}$, $\mathbf{e}_1$ and $\mathbf{e}_2$, all from

χ_μ^n, have an exact amount of weight. This allows to achieve a perfectly correct scheme as long as the error vector is within the capacity of the error-correction code [52].

Theorem 1 (Correctness). Lepton.CPA *is perfectly correct as long as the error correction code* ECC *employed can correct any* $(4k^2 + k + 2m - 2)$*-bit errors (recall* $k = \mu n$*).*

Proof. For any valid $(pk, sk) = ((\mathbf{a}, \mathbf{b} = \mathbf{as} + \mathbf{e}), \mathbf{s})$ and plaintext $\mathbf{p}$, let $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2)$ be the corresponding ciphertext and define error vector $\mathbf{v}$ as follows

$$\tilde{\mathbf{c}}_0 := \mathbf{c}_2 - \mathsf{Trunc}(\mathbf{c}_1\mathbf{s}, \ell) \;=\; \mathsf{ECC}(m) + \underbrace{\mathsf{Trunc}(\mathbf{ex} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2, \ell)}_{\text{error term } \mathbf{v}} \;.$$

and decryption is correct if $\mathsf{ECC}^{-1}(\tilde{\mathbf{c}}_0) = m$. The conclusion follows from Lemma 1:

$$|\mathbf{ex} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2| \le |\mathbf{ex}| + |\mathbf{e}_1\mathbf{s}| + |\mathbf{e}_2| \le 4k^2 + k + 2m - 2 \;.$$

Trading correctness for efficiency. In fact, there's no need for the truncating function Trunc in both encapsulation and decapsulation as long as a sufficiently long code (of length n) is used to perfectly correct all possible errors. However, as analyzed in [52], the time spent on decoding would take roughly 10^8 cycles. Alternatively, we use a much shorter code (by truncation) and give an empirical estimation about the error rates, which helps to achieve a 2000 times speedup on decoding time at the cost of an exponentially small error (see Section 6 for details).

4.2 Lepton.CCA

We now define $\mathsf{Lepton.CCA} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with session key space $\mathcal{K} = \mathbb{F}_2^\ell$, which is obtained by applying the FO transform [28,21,47] to Lepton.CPA.

Algorithm 4: $\mathsf{Lepton.CCA.KeyGen}(1^\kappa)$

$\eta \xleftarrow{\$} \mathbb{F}_2^\kappa$;
$(\rho, \sigma) := G(\eta, 2)$;
$(\mathbf{s}, \mathbf{e}) \sim (\chi_\mu^n)^2 := F(\rho, 2)$;
$\mathbf{a} := \mathsf{Samp}(\sigma) \in \mathbb{F}_2^n$;
$\mathbf{b} := \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$;
return $(pk, sk) = \big((\sigma, \mathbf{b}), \mathbf{s}\big)$;

Algorithm 5: $\mathsf{Lepton.CCA.Encaps}(pk = (\sigma, \mathbf{b}))$

$\mathbf{a} := \mathsf{Samp}(\sigma) \in \mathbb{F}_2^n$;
$m \xleftarrow{\$} \mathbb{F}_2^\kappa$;
$(K', r, d) := G(m \| pk, 3) \in (\mathbb{F}_2^\kappa)^3$;
$(\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2) \sim (\chi_\mu^n)^3 := F(r, 3)$;
$\mathbf{c}_1 := \mathbf{a}\mathbf{x} + \mathbf{e}_1$;
$\mathbf{c}_2 := \mathsf{Trunc}(\mathbf{b}\mathbf{x} + \mathbf{e}_2, \ell) + \mathsf{ECC}(m)$;
$\mathbf{C} := (\mathbf{c}_1, \mathbf{c}_2, d) \in \mathbb{F}_2^{n+\ell+\kappa}$;
$K := G(K' \| \mathbf{C}) \in \mathbb{F}_2^\kappa$;
return $\big(\mathbf{C}, K\big)$;

Algorithm 6: $\mathsf{Lepton.CCA.Decaps}\big(sk = \mathbf{s}, \mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, d)\big)$

$\tilde{\mathbf{c}}_0 := \mathbf{c}_2 - \mathsf{Trunc}(\mathbf{c}_1\mathbf{s}, \ell)$;
$m := \mathsf{ECC}^{-1}(\tilde{\mathbf{c}}_0)$;
$(K', \mathbf{C}') = \mathsf{Lepton.CCA.Encaps}(pk; m)$;
if $\mathbf{C}' = \mathbf{C}$ **then**
  $K := K'$;
else
  $K := \perp$;
return K ;

Theorem 2 (Correctness). Lepton.CCA *is perfectly correct as long as the error correction code* ECC *employed can correct any* $(4k^2 + k + 2m - 2)$*-bit errors (recall* $k = \mu n$*).*

Proof. For any valid $(pk, sk) = ((\mathbf{a}, \mathbf{b} = \mathbf{a}\mathbf{s} + \mathbf{e}), \mathbf{s})$ and plaintext $\mathbf{p}$, let $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2)$ be the corresponding ciphertext and define error vector $\mathbf{v}$ as follows

$$\tilde{\mathbf{c}}_0 := \mathbf{c}_2 - \mathbf{c}_1\mathbf{s} \;=\; \mathsf{ECC}(\mathbf{p}) + \underbrace{\mathsf{Trunc}(\mathbf{e}\mathbf{x} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2, \ell)}_{\text{error term } \mathbf{v}} \ .$$

and decryption is correct if $\mathsf{ECC}^{-1}(\tilde{\mathbf{c}}_0) = \mathbf{p}$. The conclusion follows from Lemma 1:

$$|\mathbf{e}\mathbf{x} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2| \le |\mathbf{e}\mathbf{x}| + |\mathbf{e}_1\mathbf{s}| + |\mathbf{e}_2| \le 4k^2 + k + 2m - 2 \ .$$

5 Security Analysis

5.1 Hardness of (Ring)-CLPN

Figure 2 depicts the connections among LPN, CLPN and their ring variants. In particular, (Ring-)CLPN is at least as hard as (Ring-)LPN, whose proof seems folklore and simply combines the known reduction techniques from LPN to exact LPN [33] and from LPN to the HNF-LPN [4].

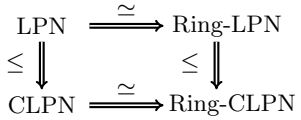

Fig. 2. A visualized view of the connections among LPN, CLPN and their ring variants, where '$\leq$' denotes implication and '$\simeq$' denotes a non-trivial connection (via a win-win argument).

Lemma 2. *We have following reductions from (Ring-)LPN to (Ring-)CLPN.*

1. LPN $\leq$ CLPN. *Any algorithm $\mathcal{A}$ that breaks computational (n,μ)-CLPN with probability ϵ implies another algorithm $\mathcal{A}'$ of roughly the same complexity as $\mathcal{A}'$ that breaks computational (n,μ,$q = 2n$)-LPN with probability $\Omega(\epsilon/n)$.*
2. Ring-LPN $\leq$ Ring-CLPN. *Any algorithm $\mathcal{A}$ that breaks computational (n,μ,2)-Ring-CLPN with probability ϵ implies another algorithm $\mathcal{A}'$ of roughly the same complexity as $\mathcal{A}'$ that breaks computational (n,μ,3)-Ring-LPN with probability $\Omega(\epsilon/n^{3/2})$.*

Proof. It suffices to prove the first statement as the reduction for the second one is similar. Consider applying $\mathcal{A}$ on the HNF-LPN, where $\mathbf{x}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_\mu^n$. Conditioned on the event $|\mathbf{x}| = |\mathbf{e}| = \mu n$ which occurs with probablity

$$\Pr[|\mathbf{x}| = |\mathbf{e}| = \mu n] = \Pr[|\mathbf{x}| = \mu n] \cdot \Pr[|\mathbf{e}| = \mu n] = \Omega((1/\sqrt{n})^2) = \Omega(1/n) \ ,$$

the problem is exactly CLPN, and thus $\mathcal{A}$ inverts the HNF-LPN with an overall probability $\Omega(\epsilon/n)$. We complete the proof by a reduction from LPN to the HNF-LPN [4] (see Lemma 5).

(C)LPN $\simeq$ Ring-(C)LPN. Inspired by Ring-LPN [26], we apply the ring heuristics to CLPN and use the resulting Ring-CLPN to construct practically efficient KEM and PKE. Unfortunately, unlike its large-modulus analogue Ring-LWE (to which reductions are known from ideal lattice), Ring-LPN is short of formal security treatments of any sort, e.g., we don't even known how the security of Ring-LPN relates to that of LPN. To fill this gap, below we argue that Ring-LPN may be as nearly secure as the LPN problem (of a sightly lower noise rate) as long as Ring-LPN can serve certain heuristic purposes (for public randomness generation). We note that the connection extends to Ring-CLPN as well (and we recall that Ring-LPN already implies Ring-CLPN by Lemma 2). We establish the connection by a hybrid of hardness assumptions (called Assumptions 1,2 and 3 respectively) from the (relatively) most well believed (i.e., LPN) to the least understood (i.e., Ring-LPN) and connecting adjacent ones.

1. **Conservative.** The (n,μ,nq)-LPN assumption.
2. **De facto.** The (n,μ,nq)-LPN assumption except that public matrix $\mathbf{B}$ is generated with (n,μ',q)-$\mathsf{Ring\text{-}LPN}$ instances by $\mathbf{B} = \mathbf{A} \cdot \mathbf{X} + \mathbf{E}$ (see matrix visualization below), where $\mathbf{A} \xleftarrow{\$} R^{q\times 1}$, $\mathbf{X} \xleftarrow{\$} (\mathcal{B}_{\mu'}^R)^{1\times n}$ and $\mathbf{E} \xleftarrow{\$} (\mathcal{B}_{\mu'}^R)^{q\times n}$.
3. **Idealized.** The (n,δ,q)-$\mathsf{Ring\text{-}LPN}$ assumption, where δ is roughly the noise rate of taking the XOR sum of μn independent samples from $\mathcal{B}_{\mu'}$, e.g., $\delta \approx \frac{1}{2}(1 - e^{-2\mu\mu' n}) \approx \mu\mu' n$ when $\mu\mu' n = o(1)$ (see Lemma 6).

$$\mathbf{A} \stackrel{\text{def}}{=} \begin{bmatrix} \mathsf{mat}(\mathbf{a}_1) \\ \vdots \\ \mathsf{mat}(\mathbf{a}_q) \end{bmatrix}, \ \mathbf{X} \stackrel{\text{def}}{=} [\mathbf{x}_1, \cdots, \mathbf{x}_n], \ \mathbf{E} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{e}_{11}, \mathbf{e}_{12}, \cdots, \mathbf{e}_{1n} \\ \mathbf{e}_{21}, \mathbf{e}_{22}, \cdots, \mathbf{e}_{2n} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ \mathbf{e}_{q1}, \mathbf{e}_{q2}, \cdots, \mathbf{e}_{qn} \end{bmatrix},$$

$$\mathbf{B} := \mathbf{A} \cdot \mathbf{X} + \mathbf{E} = \begin{bmatrix} \mathbf{a}_1\mathbf{x}_1 + \mathbf{e}_{11}, \mathbf{a}_1\mathbf{x}_2 + \mathbf{e}_{12}, \cdots, \mathbf{a}_1\mathbf{x}_n + \mathbf{e}_{1n} \\ \mathbf{a}_2\mathbf{x}_1 + \mathbf{e}_{21}, \mathbf{a}_2\mathbf{x}_2 + \mathbf{e}_{22}, \cdots, \mathbf{a}_2\mathbf{x}_n + \mathbf{e}_{2n} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ \mathbf{a}_q\mathbf{x}_1 + \mathbf{e}_{q1}, \mathbf{a}_q\mathbf{x}_2 + \mathbf{e}_{q2}, \cdots, \mathbf{a}_q\mathbf{x}_n + \mathbf{e}_{qn} \end{bmatrix} \in \mathbb{F}_2^{qn\times n} \ .$$

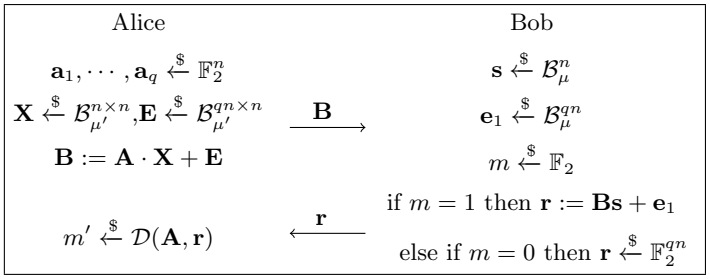

Fig. 3. A two-pass key agreement protocol that enables Alice and Bob agree on a single key bit, where the passive security is ensured by Assumption 2 and the (noticeable) correctness is guaranteed by any efficient distinguisher $\mathcal{D}$ that falsifies Assumption 3.

Assumption 1 → Assumption 2. A reduction can be established if we additionally assume (n,μ',q)-Ring-LPN as $\mathbf{B} = \mathbf{A}\cdot\mathbf{X} + \mathbf{E}$ is an instance of n-fold (n,μ',q)-Ring-LPN (but then the statement becomes meaningless). However, we argue that this may not be necessary as the matrix $\mathbf{A}$ can also be a secret and in practice we may not even need pseudorandomness for public matrix. Intuitively, the operations of $\mathbf{A}\cdot\mathbf{X} + \mathbf{E}$ may already destroy the ring structures in $\mathbf{A}$ sufficiently enough in order to safely replace a random matrix in standard LPN. In fact, similar heuristic-based approaches for generating public matrix were also adopted by postquantum public-key cryptographic schemes such as Frodo, New Hope and Kyber, where either a RO or a PRG (with seed made public) is used to generate as much public randomness as needed without a formal proof in standard model.

Lemma 3 (Assumption 2 → Assumption 3). *Under assumption 2, i.e., the (n,μ,nq)-LPN problem is hard even when the public matrix is replaced with $\mathbf{B} = \mathbf{A}\cdot\mathbf{X} + \mathbf{E}$, then at least one of the following is true:*

1. *Assumption 3 holds, i.e., the (n,δ,q)-Ring-LPN problem is hard;*
2. *IND-CPA secure PKEs are implied.*

We defer the proof to Appendix B.

How to interpret the results. We stress that the above is not a proof that LPN implies Ring-LPN. Instead, it essentially states that, either it is not even secure to use Ring-LPN for generating public randomness for LPN (e.g., when the ring has a reducible polynomial that introduces vulnerability), or that (n,μ,nq)-LPN may imply either (n,δ,q)-Ring-LPN or public-key encryptions. The last case is very unlikely, especially for certain parameter settings, e.g., for $\mu = 1/n^{2/5}$, $\mu' = 1/n^{3/4}$ (and thus $\delta \approx 1/n^{3/20}$) it is unlike to build a PKE from LPN of a noise rate $1/n^{2/5}$, which is beyond Alekhnovich's $1/\sqrt{n}$ noise regime[5], and therefore $(n,1/n^{3/20},q)$-Ring-LPN is hard. In fact, even for $\mu \leq 1/\sqrt{n}$ it would be interesting to see any LPN-based PKE that does not follow the conventional blueprint in Figure 1, but a quite exotic one as in Figure 3.

5.2 Known Solvers and Attacks for (Ring-)CLPN

Concrete hardness of (Ring-)LPN. Moreover, (Ring-)CLPN appears significantly harder than LPN, in particular, the best known (low-noise) LPN solvers [8,39,6,40] run much beyond complexity $2^{O(\mu n)}$ when applied to (Ring-)CLPN. This is attributed to the new noise distribution χ_μ^n introduced. In this respect, CLPN is more related to the LSPN problem, which also samples from χ_μ^n for its secret and whose best known algorithms (in low and high noise regimes respectively) are stated below.

[5] The only LPN-based PKE beyond the $1/\sqrt{n}$ noise regime [50] has only quasi-polynomial security. This possibility can be ruled out as the PKE in Figure 3 is as secure as the underlying LPN.

Theorem 3 (LSPN solvers). *Let (n, μ, η)-LSPN refer to the problem of recovering* $\mathbf{x}$ *from* $(\mathbf{A},\ \mathbf{A}\cdot\mathbf{x}+\mathbf{e})$ *where* $\mathbf{A} \xleftarrow{\$} \mathbb{F}_2^{q\times n}$, $\mathbf{x} \xleftarrow{\$} \chi_\mu^n$ *and* $\mathbf{e} \xleftarrow{\$} \mathcal{B}_\eta^q$. *Then,*

- ***(Grigorescu et al.'s Algorithm [23]).*** *For any $\mu = o(1)$, $0 < \eta < 1/2$, there is an algorithm that solves the (n, μ, η)-LSPN problem with probability more than 1/2 and time complexity*
$$\mathsf{poly}(\frac{1}{1-2\eta}) \cdot n^{\frac{(1+(2\eta)^2+o(1))\mu n}{2}}$$
and sample complexity $q = \omega(\mu n \log n)/(1-2\eta)^2$.
- ***(Valiant's Algorithm [48,49]).*** *For any $\mu = o(1)$, $0 < \eta < 1/2$, there is an algorithm that solves the (n, μ, η)-LSPN problem with probability more than 1/2 and time complexity*
$$\mathsf{poly}(\frac{1}{1-2\eta}) \cdot n^{\frac{\omega+\epsilon}{3}\mu n} \approx \mathsf{poly}(\frac{1}{1-2\eta}) \cdot n^{0.8\mu n}\ ,$$
where $\omega \approx 2.38$ is the exponent in the complexity of the fast matrix multiplication algorithm and ϵ is an arbitrary constant.

CLPN SOLVERS. The only difference between LSPN and CLPN is that CLPN uses a non-Bernoulli noise $\mathbf{e} \xleftarrow{\$} \chi_\mu^n$. It is easy to observe that short substrings of χ_μ^n behave quite close to a Bernoulli distribution of rate μ, which allows us to adapt the LSPN solver [23] to yield a CLPN solver as stated in Corollary 1 (with proof given in Appendix B).

Corollary 1 (CLPN Solvers). *For any $\mu = O(1/\sqrt{n})$, there exists an algorithm [23] that solves the search (n,μ)-CLPN problem with probability more than 1/2 and time complexity*
$$n^{\frac{(1+o(1))\mu n}{2}}\ .$$

CLASSICAL VS. QUANTUM HARDNESS OF SEARCH (RING-)CLPN. It turns out that even reducing the constant factor in the exponent is highly non-trivially as the LPN problem and its variants have been well-studied and some solvers (e.g., [48,49]) already employ heavy techniques, such as fast matrix multiplication, only to bring down constant factor in exponent from 1 to 0.8. Throughout, we will use $n^{0.5\mu n}$ as the estimated classical complexity for solving search (Ring)-CLPN, and use $n^{0.25\mu n}$ for its quantum hardness, which is reasonable (and already quite conservative) as the only known quantum algorithm [19] for LPN and its variants is based on Grover search and thus achieves a speedup of no more (and actually much less) than quadratic.

ON (UN)PRACTICAL RELEVANCE OF DISTINGUISHING ATTACKS. We stress that decisional CLPN is equivalent to its search counterpart, which simply follows the reductions between search and decisional LPN [9,36,5]. On the one hand solving search CLPN would lead to a distinguishing attack on decisional CLPN with the same advantage; and on the other hand for noise rate $\mu = 1/\sqrt{n}$ there exists much efficient distinguishing attacks with advantage about $n^{-\sqrt{n}/4}$ (see Lemma 4). However, such an attack is not practical as it applies only in the event of a bad matrix (which occurs with probability $2^{-3\sqrt{n}}$) and even conditioned on such an event the advantage is only about $n^{-\sqrt{n}/4}$. We refer to the proof of Lemma 4 for details. If one wants to boost the distinguishing attack with advantage ε (e.g., $n^{-\sqrt{n}/4}$) into one with constant advantage, then the time complexity is raised to $1/\varepsilon^2$ (e.g., $n^{\sqrt{n}/2}$), which matches (and justifies) the complexity we estimated for search (Ring-)CLPN.

Lemma 4 (Distinguishing attacks on (Ring-)CLPN). *For any constant $0 < v < 1$ and noise rate $\mu = n^{-v}$, there exists a polynomial-time algorithm that solves the decisional (n,μ)-CLPN and (n,μ,2)-Ring-CLPN (defined w.r.t. an irreducible polynomial g) with advantage $n^{-\frac{vn^{1-v}}{2}}$.*

Proof. Given a square matrix $\mathbf{A}$ (which is either uniformly random or equals to $\mathsf{mat}(\mathbf{a})$ for a random $\mathbf{a}$), we denote by $\mathbf{b}_i^\mathsf{T}$ the i-th row vector of $\mathbf{A}$. Let $\mathcal{D}$ be a distinguisher that outputs 0 if

there exists $\mathbf{b}_i^\mathsf{T}$ with $|\mathbf{b}_i^\mathsf{T}| \leq \frac{n}{2} - n^{1-c}$ (for some $0 < c \leq 1/2$ to be determined later), or otherwise $\mathcal{D}$ outputs a uniform random bit.

$$\text{We have} \quad \Pr\left[|\mathbf{b}_i| \leq \frac{n}{2} - n^{1-c}\right] = 2^{-n} \cdot \sum_{i=0}^{n(1/2-n^{-c})} \binom{n}{i} > 2^{n(\mathbf{H}(1/2-n^{-c})-1)} > 2^{-3n^{1-2c}} \ ,$$

where $\mathbf{H}(\alpha) = \alpha\log(1/\alpha)+(1-\alpha)\log(1/(1-\alpha))$ is the binary entropy function whose asymptotic estimations follow from Lemma 8 and Lemma 9. For any fixed $\mathbf{x}$ of weight $\mu n = n^{1-v}$ and $\mathbf{b}_i^\mathsf{T}$ uniformly chosen from strings of Hamming weight $n/2 - n^{1-c}$ (i.e., $\chi^n_{1/2-n^{-c}}$) or less we have by Lemma 7 $\Pr[\mathbf{b}_i^\mathsf{T}\mathbf{x} = 0] \geq 1/2 + n^{-cn^{1-v}}/2$. Take into account the noise bit $\mathbf{e}_i$ we have $\Pr[\mathbf{b}_i^\mathsf{T}\mathbf{x} + \mathbf{e}_i = 0] \geq 1/2 + (1/2-\mu)n^{-cn^{1-v}}$. Thus, overall $\mathcal{D}$ distinguishes with advantage

$$(1/2-\mu)n^{-cn^{1-v}} \cdot 2^{-3n^{1-2c}} \underset{c=v/2}{\approx} n^{-\frac{vn^{1-v}}{2}}$$

where it is natural to set $c = v/2$ for maximized advantage.

5.3 Security Reductions from Ring-CLPN

We first show that the KEM scheme Lepton.CPA, introduced in Section 4, in IND-CPA secure under the Ring-CLPN assumption and thus we defer the proof to Appendix B. The IND-CCA Security of Lepton.CCA follows from Fujisaki-Okamoto transform in the quantum random oracle model [28,21,47].

Theorem 4 (IND-CPA Secure KEM). *If both $H_1 : \mathbb{F}_2^\ell \to \chi_\mu^{\mu n} \times \chi_\mu^{\mu n} \times \chi_\mu^{\mu n}$ and $H_2 : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ are modeled as random oracles, then* Lepton.CPA *is an IND-CPA secure KEM under the (n,2,μ)-Ring-CLPN assumption.*

Corollary 2 (IND-CCA Secure KEM). *If both $H_1 : \mathbb{F}_2^\ell \to \chi_\mu^{\mu n} \times \chi_\mu^{\mu n} \times \chi_\mu^{\mu n}$ and $H_2 : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ are modeled as random oracles, then* Lepton.CCA *is an IND-CCA secure KEM under the (n,2,μ)-Ring-CLPN assumption.*

Proof. We apply a KEM variant [27] of the Fujisaki-Okamoto transform [21] to the IND-CPA secure KEM Lepton.CPA just proved, and obtain a CCA secure KEM in the classic/quantum random oracle model, i.e., H_1 and H_2 are both modeled as random oracles.

5.4 Concrete Security Strengths and Parameters

Ideally we prefer n to be exactly (or slightly less than) a power of 2 to facilitate the sampling of χ_μ^n. The irreducible polynomial of the form $g = X^n + X^m + 1$ needs as small as possible m (e.g., $m \leq 10$) for correctness. Taking into account of both considerations, we set $g = X^{8100} + X^9 + 1$ for the light security level, and set $g = X^{16153} + X^{10} + 1$ and $g = X^{32767} + X + 1$ for achieving moderate and paranoid security respectively, where n is set to $8100 = 2^{13} - 92$, $16153 = 2^{14} - 231$ and $32767 = 2^{15} - 1$ as irreducible g over $\mathbb{F}_2$ does not exist for $n = 2^{13}, 2^{14}$ or 2^{15}. By using difference choices of μn, we obtain estimated strengths from 103-bit to 299-bit classical security.

	n	m	μn	Clas-Sec.	Qua-Sec.
Light I	8100	9	16	103	51
Light II	8100	9	20	128	64
Moderate I	16153	10	19	132	66
Moderate II	16153	10	24	167	83
Moderate III	16153	10	28	195	97
Moderate IV	16153	10	37	258	129
Paranoid	32767	1	35	262	131
Paranoid II	32767	1	40	299	149

6 Implementations and Performances

In this section we give the details of our implementations of Lepton and report on performance of the subroutines. The NIST PQC CFP requires both a reference implementation that clearly maps to the algorithm description and an optimized implementation (targeting the Intel x64 processor) in order to demonstrate the performance. As our basic reference implementation with source code written in ANSI C already provides quite desirable performance and also due to the lack of specialized skills in deep optimizations, we give only the reference implementation and make it the same for the optimized one. The memory requirement is thus moderate as the basic implementation involves almost[6] no trading-space-for-speed optimization. We leave it as future work to investigate how much further can the implementation be optimized. All cycle counts in this section were obtained on a single core Intel Core-i7 4790 CPU at 3.6 GHz and 4.0 GB RAM running on the Ubuntu 14.04 LTS 64-bit OS.

6.1 Implementation Details

Instantiation of G. Let $G(\cdot, \ell) : \mathbb{F}_2^* \longrightarrow (\mathbb{F}_2^\kappa)^\ell$ be a hash function which takes as input a random source and outputs ℓ length-κ bit strings. In our implementations, we instantiate G with the extendable output function SHAKE-128, standardized in FIPS 202 [44]. The concrete code for implementing SHAKE-128 was based on the one provided by the authors of Kyber [11].

Instantiation of Samp. Let $\mathsf{Samp} : \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be an algorithm which expands length κ random bit strings into $\mathbb{F}_2^n$. Following the schemes in [3,11,12,13], we use Samp to compress public keys. Note that we only use binary polynomials, which allows us to simply instantiate Samp by simply calling a seed expander to generate sufficiently long random bits. Specifically, we first use the domain-separated version cSHAKE-128, which was standardized in FIPS 800-185 [34], to expands a length ℓ-bit seeds into $k = \lceil n/8 \rceil$ bytes, and then return the first n bits (i.e., clearing the last $8 * k - n$ bits).

Instantiation of F. Let $F(\cdot, \ell) : \mathbb{F}_2^\kappa \longrightarrow (\chi_\mu^n)^\ell$ be a sampling algorithm that takes as input a random source and outputs ℓ noise vectors over χ_μ^n. We instantiate F by first calling the domain-separated version cSHAKE-128 [34] to generate sufficiently long random bits, which were then used by a sample algorithm to generate the desired outputs. For our purpose, it suffices to consider an algorithm that samples a single χ_μ^n, as we give in Algorithm 8. We briefly introduce the idea: find a nearby two's power, i.e., $b_n \in \mathbb{N}$ such that $2^{b_n-1} < n \leq 2^{b_n}$, and recall that n was chosen to be as close to 2^{b_n} as possible. Thus, we divide the random bits into b_n-bit blocks $\mathbf{r}_1, \mathbf{r}_2, \ldots$, where each $\mathbf{r}_i$ refers to set the $(\mathbf{r}_i)$-th bit (of the string to be sampled) to be 1 and of course ignore those $\mathbf{r}_i$ beyond n or already collide with a previous $\mathbf{r}_j$ $(i > j)$. Note that with an infinitely small probability this process could run forever before reaching a desired weight $k = \mu n$. However, it suffices to repeat at most $c_f \cdot k$ times for some small constant $2 \leq c_f \leq 3$ such that χ_μ^n is perfectly sampled except for a failing probability of 2^{-130} or even less, which

[6] The only exception is that a pre-computed file (the "precomp9-30-256.c" in the source code) is used for fast BCH encoding and decoding, which has size 32.4KB and thus incurs small memory consumption.

is formally proved in Lemma 10. For our convenience we simply fix $c_f = 3$ for all parameter settings with concrete error probabilities given in Table 1.

Algorithm 7: $F(\mathbf{w}, \ell)$

$b_n := \lceil \log(n) \rceil$;
$k := \mu n$;
apply cSHAKE-128 on $\mathbf{w}$ to generate $\mathbf{t}$ of $3kb_n$ bits (and discard the rest);
parse $\mathbf{t}$ into ℓ blocks of $(3kb_n)$-bit substrings : $\mathbf{t} = \mathbf{t}_1 \| \dots \| \mathbf{t}_\ell$;
for $i := 1$ **to** ℓ **do**
  $\mathbf{v}_i := \mathsf{Sample}.\chi_\mu^{\mathsf{n}}(\mathbf{t}_i)$;
return $\mathbf{v}_1 \| \dots \| \mathbf{v}_\ell$;

Algorithm 8: $\mathsf{Sample}.\chi_\mu^{\mathsf{n}}(\mathbf{r} \in \mathbb{F}_2^{3kb_n})$

$b_n := \lceil \log(n) \rceil$;
$k := \mu n$;
parse $\mathbf{r}$ into $3k$ blocks of b_n-bit substrings: $\mathbf{r} = \mathbf{r}_1 \| \mathbf{r}_2 \| \dots \| \mathbf{r}_{3k}$;
$\mathbf{u} := \mathbf{0} \in \mathbb{F}_2^n$;
$i := 1$;
while $|\mathbf{u}| < k$ *and* $i \leq 3k$ **do**
  $j := \mathsf{Integer}(\mathbf{r}_i) + 1$;
  if $j \leq n$ **then**
    set the j-th bit of $\mathbf{u}$ to be 1 ;
  else
    ignore ;
  $i := i + 1$;
return $\mathbf{u}$;

Table 1. The probabilities that Algorithm 8 (for $c_f = 3$) fails to sample χ_μ^n in different settings.

	n	m	μn	Err. Prob. of Samp. Failure
Light I	8100	9	16	$< 2^{-132}$
Light II	8100	9	20	$< 2^{-168}$
Moderate I	16153	10	19	$< 2^{-130}$
Moderate II	16153	10	24	$< 2^{-170}$
Moderate III	16153	10	28	$< 2^{-203}$
Moderate IV	16153	10	37	$< 2^{-278}$
Paranoid I	32767	1	35	$< 2^{-634}$
Paranoid II	32767	1	40	$< 2^{-710}$

6.2 Error Correction Codes and Error Rates

Instantiations of ECC. Let $\mathsf{ECC} : \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^\ell$ be an error correction code with an efficient decoding algorithm ECC^{-1}. As analyzed in the proof of Theorem 1, the error vector (before truncating) $\mathbf{ex} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2$ has a upper bound on its weight, and thus we could even use a long error correction code to correct all possible errors. For example, in order to achieve perfect correctness for the Moderate I parameter set, one can use a BCH code of length 2^{14} with error correcting ability up to 1481 bits, which we implemented and verified in [52]. This incurs a relatively high computational overhead: the decoding itself takes about 10^8 cycles [52]. Taking a closer look at the error vector $\mathbf{ex} - \mathbf{e}_1\mathbf{s} + \mathbf{e}_2$, we observe that for a fixed public key (and thus fixed $\mathbf{s}$ and $\mathbf{e}$) the distribution is induced by $\mathbf{x}$, $\mathbf{e}_1$ and $\mathbf{e}_2$ for independent ciphertexts,

and would therefore behave like a random distribution of bounded weight and allows tradeoff between correctness and efficiency. We use a two-layered coding scheme, namely, first encode the message of $\kappa = 256$ bits by a BCH code $\mathsf{BCH} : \mathbb{F}_2^{\kappa=256} \longrightarrow \mathbb{F}_2^{508}$ capable of correcting any 30-bit errors and followed by a repetition code REP of parameter RCN (correcting up to $\mathsf{RCN}/2$-bit error), namely,

$$\mathsf{ECC}(\mathbf{m}) = \mathsf{REP}\big(\mathsf{BCH}(\mathbf{m}), \mathsf{RCN}\big)$$

where the resulting codeword size is $\ell = 508 \cdot \mathsf{RCN}$ bits. We use the BCH code implementation from [16] with their parameters[7] (n, m, t) set to $(508, 9, 30)$ and we make some further efficiency optimizations for the parameters of our choice (code provided as part of the submission).

Correctness of ECC. In the decryption algorithm, we reverse the process of encoding, i.e., first decoding repetition code by a majority voting and then decoding with BCH^{-1}. To analyze the probability of decryption failure, we apply a theoretical estimation based on empirical data. Concretely, for each parameter set, we first use a relatively small RCN and counts the number of decryption failures occurred in a total 2^{21} times runs of our program (we actually test our program up to 2^{32} times for some parameter set). Under the assumption that the error independently occurs at each coefficient of a degree $n-1$ polynomials with certain probability p (which is roughly confirmed by our empirical results), we can calculate a concrete p such that the overall decryption failure probability matches the empirical result. Then, fixing the probability p that each coefficient has an error, we increase RCN to reduce the overall decryption failure probability to a negligible one. For example, by using a repetition code with a small $\mathsf{RCN} = 3$, the number of decryption failures counted is less than 10 in a total 2^{21} times for the Moderate I parameter set. In other words, the empirical decryption failure probability is at most $10/2^{21}$ (in fact, we use a more conservative $20/2^{21}$ to upper bound the decryption failure probability). Then, we calculate the probability that each coefficient occurs error is roughly 0.0865, i.e., $p \approx 0.0865$, which corresponds to an expected total errors about 1398. Note that the theoretical upper bound for such parameter set is 1481 by Theorem 1, this means that our estimation is promising and somewhat accurate. Finally, fixing the probability $p = 0.0865$, we increase the length of the repetition code to 7 (i.e., $\mathsf{RCN} = 7$), and obtain an estimated probability about 2^{-114} in our analysis model. We do the same analysis for other parameter sets and give the decryption error rates in Table 2 and Table 3.

6.3 Build and Run the Program

In the Lepton directory of our submission package, we include all the source codes for implementing Lepton.CPA and Lepton.CCA. We also provide a readme.txt file for explaining how to use the software. Our software is written in pure C language. One can also find our source code at `https://github.com/whntgxtia/Lepton`. To compile and build the program, one can use the following commands (this only works when the openssl library was installed in the directory /usr/local/include for header files and /usr/local/lib for binary library):

```
1 make
2 make test
3 make kat
```

The first line command will compile and build software, after which two executable program: kem_test and kem_kat, will be generated. The second line will run the kem_test program, and output the running time in the command terminal. The third line will create two files (i.e., the REQUEST and RESPONSE files) for the known answer tests.

If the openssl library was not installed in the directories /usr/local/include and /usr/local/lib, one can set pass the concrete paths to the "make" command:

```
1 make INC=-I/path_to_openssl_header    LIB=-L/path_to_openssl_lib
```

[7] Here parameters n, m and t are not those defined in this document but used in the source code of [16]. In fact, the actual BCH codeword in use consists of 511 bits, which includes 259 message bits and 252 parity check bits, where 3 message bits are dropped since our message is of length $\kappa = 256$.

By default, the "make" procedure will initialize the program to run the Lepton.CCA scheme under the choice of Moderate I parameter set. One can change this by passing the PARAM and ALGOR arguments to "make", e.g.,

```
1 make PARAM=-DLIGHT_I    ALGOR=-DLEPTON_CPA
```

The above command will initialize the program to run the Lepton.CPA scheme under the choice of Light I parameter set. The following tables gives all the argument values and the respective meanings.

Flag	Value	Meaning
PARAM	-DLIGHT_I	using Light I parameter set
	-DLIGHT_II	using Light II parameter set
	-DMODER_I	using Moderate I parameter set
	-DMODER_II	using Moderate II parameter set
	-DMODER_III	using Moderate III parameter set
	-DMODER_IV	using Moderate IV parameter set
	-DPARAN_I	using Paranoid I parameter set
	-DPARAN_II	using Paranoid II parameter set
ALGOR	-DLEPTON_CPA	using Lepton.CPA scheme
	-DLEPTON_CCA	using Lepton.CCA scheme

6.4 Computational Efficiency

We list the performance parameters for Lepton.CPA and Lepton.CCA in Table 2 and Table 3 respectively. The size of the public key roughly corresponds to the value of n, and the ciphertext size ranges from $1.25n$ to nearly $2n$, which accounts for the lengths of the full-length $\mathbf{c}_1$ and truncated $\mathbf{c}_2$ (whose size in turn depends on the parameter of the repetition code RCN). The running time for key generation, encapsulation and decapsulation is in terms of the averaged number of CPU cycles. In general, there are no substantial differences in key and ciphertext sizes, key generation and encapsulation running time, between Lepton.CPA and Lepton.CCA, except that decapsulation time of Lepton.CCA nearly triples that of Lepton.CPA due to the additional invocation of the encapsulation process in its decapsulation algorithm. Overall, both Lepton.CPA and Lepton.CCA exhibit quite desirable computational efficiency and low memory consumptions even compared with other known post-quantum candidates. Finally, the choices of the parameter RCN code are reflected in the error rates of decapsulation.

Remark 1. Table 2 and Table 3 may look somewhat counterintuitive in that the decapsulation time for Lepton.CCA is much smaller than the sum of the encryption time and decryption time for Lepton.CPA. This is due to the fact most operations in our implementations are so cheap (i.e., mostly bit operations) that even the hash function (i.e., SHA3) becomes the bottleneck of the overall performance which accounts for 58% $\sim$ 72% (resp, 58% $\sim$ 70%) of the time for Lepton.CPA.KeyGen (resp, Lepton.CCA.KeyGen), 65% $\sim$ 67% (resp, 66% $\sim$ 70%) for Lepton.CPA.Encaps (resp, Lepton.CCA.Encaps) and 49% $\sim$ 57% (resp, 59% $\sim$ 64%) for Lepton.CPA.Decaps (resp, Lepton.CCA.Decaps). Compared to the sum of Lepton.CPA.Decaps and Lepton.CPA.Encaps , the implementation of Lepton.CCA.Decaps only has to compute the shared session key once which allows us to save an expensive hash operation (i.e., $K = G(K'\|\mathbf{C})$). All of the above also indicate that our schemes can be much faster in hardware implementations.

6.5 Memory Requirements

To measure the memory consumption of the software implementations, we implemented a dynamic memory profiler with a widely used dynamic binary instrumentation (DBI) framework, namely Intel PIN [41], on a Debian Linux 8.5 (32-bit CPU) platform. With the support of DBI, we can monitor the program to record the used memory pages and evaluate the maximum memory usage at runtime. We tested a few typical parameters settings, ranging from the minimal Lepton.CPA Light I, Lepton.CCA Moderate I in the middle, and the maximal Lepton.CCA Paranoid II, which are measured to consume a memory of sizes 192KB 256KB and 320KB respectively.

Table 2. The key/ciphertext sizes, running times and error probabilities for Lepton.CPA

	RCN	bytes			CPU cycles			err. prob.
		$\|pk\|$	$\|sk\|$	$\|c\|$	Keygen	Enc	Dec	
Light I	9	1045	32	1585	33625	78808	33400	2^{-87}
Light II	15	1045	40	1966	34912	85347	42462	2^{-121}
Moderate I	7	2052	38	2465	48932	117275	45519	2^{-114}
Moderate II	11	2052	48	2719	51519	125178	51353	2^{-123}
Moderate III	15	2052	56	2973	51508	130057	60289	2^{-120}
Moderate IV	31	2052	74	3989	57861	152431	72564	2^{-129}
Paranoid I	19	4128	70	5303	96602	237722	97757	2^{-119}
Paranoid II	23	4128	80	5557	97884	247932	105200	2^{-148}

Table 3. The key/ciphertext sizes, running times and error probabilities for Lepton.CCA

	RCN	bytes			CPU cycles			err. prob
		$\|pk\|$	$\|sk\|$	$\|c\|$	Keygen	Enc	Dec	
Light I	9	1045	1077	1617	34308	79152	87043	2^{-87}
Light II	15	1045	1085	1998	34536	86584	100141	2^{-121}
Moderate I	7	2052	2090	2497	49943	121564	132708	2^{-114}
Moderate II	11	2052	2100	2751	51658	124426	141988	2^{-123}
Moderate III	15	2052	2108	3005	52699	130631	151185	2^{-120}
Moderate IV	31	2052	2126	4021	59450	154473	179520	2^{-129}
Paranoid I	19	4128	4198	5335	94454	234441	264881	2^{-122}
Paranoid II	23	4128	4208	5589	97569	244706	282199	2^{-148}

7 Known Answer Test Values

In the KAT directory of our submission package, we include all the KAT files for our implementations of Lepton.CPA and Lepton.CCA under the choices of all the eight parameter sets. Specifically, the KAT directory contains two sub directories Lepton.CPA and Lepton.CCA, which in turn contains eight sub directories Light I to II, Moderate I to Moderate IV, Paranoid I to II, to accommodate the corresponding KAT files. For example, the KAT files under directory \KAT\Lepton.CPA\Light_I correspond to Lepton.CPA with security strength Light I. One can also use our provided kem_kat program (which will be generated when one builds our software packages) to generate the KAT files automatically by using the command:

```
1 make kat
```

The kem_kat program will create two files: the REQUEST and RESPONSE files. By default, the kem_kat program was set to generate the REQUEST and RESPONSE files for Lepton.CCA with the Moderate I parameter set. One can change the parameters and algorithms for the kem_kat program by passing PARAM and ALGOR flags to the make procedure, e.g.,

```
1 make PARAM = -DLIGHT_I    ALGOR = -DLEPTON_CPA
2 make kat
```

Since the kem_kat program will use some subroutines from the openssl library. If the openssl library was not installed in /usr/local/include and /usr/local/lib, one can specify the concrete path to openssl by passing the INC and LIB flags to the make procedure, e.g.,

```
1 make PARAM = -DLIGHT_I    ALGOR = -DLEPTON_CPA INC=-I/usr/local/include
2         LIB = -L/usr/local/lib
3 make kat
```

By doing this, the REQUEST and RESPONSE files will be generated in the same directory of the program. The REQUEST file for each of these KAT tests contains a series of data sets consisting of a randomness seed, a public key pk, a secret key sk, a ciphertext ct and a shared session key ss. The following is a sample data set (for Lepton.CPA with Light I parameter set),

where the pk, sk, ct and ss were **left blank on purpose**.

seed = 061550234D158C5EC95595FE04EF7A25767F2E24CC2BC479D09D86DC9
ABCFDE7056A8C266F9EF97ED08541DBD2E1FFA1

pk =

sk =

ct =

ss =

The RESPONSE file for the KAT tests contains the same data as the REQUEST file. The following is a sample data set (for Lepton.CPA with Light I parameter set):

seed = 061550234D158C5EC95595FE04EF7A25767F2E24CC2BC479D09D86DC9
ABCFDE7056A8C266F9EF97ED08541DBD2E1FFA1

pk =D43193D14CED0DBE25AE5F248FA769C7FA73D4C8247F9318261D1AF59
72E1C8E948AF96155B37AF7ABCF26082E06BCE13EDD4A7CA2ED195BF970E
5405435FCC9618851EF14A3B2EC5A506334BB880EB2305B698D43F0318F57559
6DB588E3F96746BB7735E9B2D8DEB897FCDC2E3DA5B7A97B2DE4EB9DCE2
43FF1414CAE8A89266B9E40C7F4FC00C3BC836EAA71DE660ED3EA0E45881F
09E45E666ADF85893CEE0439E5C078932512D9E5132ECE431AF83A74CF9EF3
F57D458462B6F327CF6C1B5CD1E14670BE765C7C797D616FC48A7D6DBC4FA
28C67FC8D6BEC90FEEA95856E569BFCA8208794CE2ABD884FDA40C27D038
119EBA7790AE12EC7724BF216F0BEC6367169CB685FF7A59F7E3544C32D551
8B9D58B7E4D8B754412F2E7553424DEF6590B47810399A444D90ED6D0CBA3B
7308FFABBB48F2C0B2B5CC1D8373F14CBBCBF80322DFF43AD2028BD09540
FEFA21279267064AE3AB80C0D0B0DC78E4E11D1235994FC1989CFA060B68B8
F45AE652A5D226CC779BDBD7E8361CCBA19A9F73FC7E69C81A3E641B91F2
2BEFE3EDA4D016038559C797DA8C7B0017A58372AE18EDFD0CACC230BA91
62AEF472FCA9646B4FEA6D5BE3C357F54E6AA9A1880C23DEEDE2F3DADD
3A23171F043E4CBC370336D68403EFD2C8F2512D083F9A6C3DB60CAEAFF65
EADBCD2D50F2555CF3A22B2EFB80064580A4162D7FAA2A1C9C2D3382E0B1
5628AB9B47F9A02F33B6C1B31DFB7AB5538224FEB34032D3747F3E6CEDFCA
D5B623E45AB039C350AB5B64E7A5B54DF24AFFFC337FD5E17F6E16B572B9
DEC8055F89553F954BE931BEE0761E4922DCDEA895FAAF00728DCF05F39746
D638E26D39FEC85618D90DC325815985B08780389522F2A662E3E4469625671C4
CC46A949D3B43F1690B8CF3842C7812926E54BF99B1FB267F03D16E1747FDC
CBEEC6287D485C60309D73C2735B93910B9BF4649896F02EF110ABB628685FF
2B4275F291ACD573F203C205827ED0CC645180CE89D8D9F25F5920E461F0877
C55269546D10B29F6723AF5035976E651503113C9FF2D4907432C3DEF63349224
01801A470366EB7C54FCF3983B10CA7FABDF58CE27D0F768F6AA917087EB2
FAE953364CFD87DA2A7D4AEE79CDE14B9582D75FEAD4FD7F0CBB073F4A
1CCA6E46B0DC2FE77428A52AC85155BD0833D0F937EFA6CE5BDD7712EFC7
C0C18C09D599D4EDA4F7F8A761BBCB4BABBFA378ECC74AE8654397A3F59
8F1BA206B09D2CA398066671CCC0844F333AA7AB1D74D5D481426D225BAD3
AC193DA99A6FACE979AFB9DC885B5A11F3447EB7131F0C7880BCED944F1B
8699B158201EF736A81A8530CFE335EFE97FF40C8BACCB9DFD46F8A4A0B6E
F54B92A86FEA7BA79E96922AF80B825869A81FBBCF12AD533B5EFBA477845
17EEFF5618DDFEEC60FF444F0C3E058FE0D33670C506D29CC6B3E6A96BE6
B2B4EE6CA64EC508550

sk = 09CF0F5E1A381C9705DD1F380BDD0B7E15BE0E020EAD0185067A160617
341D2C

ct =16518489AEBD37FDF41A2283D18BF165FAA11D19C188F3FCA40F456A0F
17DD2466A991F1DD36BC0BE3162E3A5106D56F744EE1B760A2A2A664728356
42873473AF029AA2702F46CD0E40171171C2F65E92E70E1D828D2DBB77D658E
5FD0FAC1CB93B2A0A5554F4116383DDC7EEC92EA74C6C17F91C9FFACBC1
068A2C6B1E89439BCE868C97548546E9ECB640BF592283913B70D80E6F84F648
C62AABDFDA677F7ABE790928DC4D59C3AEE857C4FD918B80FCC5A5775CB
2ED40DD2ED9F7452387E27D1C22FA2C7A82B7F50060EA4708C73C9EF8771E6
B58B16651427772B0495C179AB28003C6D4CECAD8A2A42BADAAC57F1E95C
F75A8470ED5B858372838FB70C2E6D88E4A15F4451DD708662B5E212806C47E
E8C8D0408B3A75D29140EE78A261CB8D53C54418C865E719EB7C8BD93BDDA
31420AC4659BA73ED8B70192E9C0957195BC9F877B1C60B7E45838BF4893C05
86060315C71C915BA56AD65AF0C36FA90F51D0494AFB64467FD8E117D2DD31
7914021791C6559718080FFE70783A647F093F30669C2CF6CC4E970F2EE778DA
FB28C54D5885ED746BF139B94EC867A2EFE511DB23D2AC1B7C44E606161168
499187A0A56DB11826FEB72846E5B20F2C7AA5C710C4695901F98CF79A9BC01
ECA5C5B4453290FCA4F22CF9278CF6452E5D100A54073CF5BB2149A6458DF0
BE41B2ADDDCCAC059F5E2C598CC9FE6A2898304846811CD187FE89C847C89
45882EC210F28AA301F79A3928D0337A2CBDFDF62125859C430FD842ECA31A
D147D4BC4629BAAD0E988A99270605BD68AD633E395C47EB8F6D464F05F71
C16E846CD27119433A99A50256508A62A34875F386052C5B7234A2C1752660B33
4CC13AC2DCA05F865091BF10EE91514362A0C3FB6A7662AF76EE05C9AB207
0C8B6CBB03A903D5F3CD05C4CB02DD6FA65E9ABA00C050732F6BF480F2BF
CA615453D18D9A003B2F35B008C3BF8CC9105DE7B04C6FE65405416FBF24B0
69E79E1B7E6A0E30DD8A9F7F6846421BE7B915ADF8A450D6C7AD83829CA13
D03FE56A4ECE6BC140E491FD1B8B2CA4F73D587D93D5128BAC1DC615F92F
C05DBCF0867C4724205FF99F13671626DAA731A45D0DD62F9E8B2319B4171B
ED81359B894616CA06958C47516DDDA719FD271C230C53D2BC54290E5AC442
F06BCAD6A3F55AD6E29EDDDC24B6EC3B536155F12484D5D4A4F214D10280
E30E8D3353D3067C2383696298088D787F731E8908C81ECFAECF0995CB64B193
2B97F15DDAE87CA855486B1775F53F7FB2527E6AE5F33EBFB32271183610385
3656CA59777F08BB1782C542CC62C411B9F30EC7CD6A744AE5D9CFEE577F5
678D52EA164A4D59AEE3A1FC9EF453C051976398E246D59948EF6F16B1FAC5
4B4D6102D82CBC3637D25339AD3520C955456D057773D2773070C4F9666429C6
45660F414E80C62FEDCCA3473CC7812AD30509D49376AFA060C1F606D53E2C
E990F3C8EB970F3EA10B5C0606E6B6BF00B044F0E6ACE826C13FD9BF3E771
D52F3D93BE6CF5FFAE0FD209FE2A0F8B7A677ECA8047E9283183AB50C7901
48B859D34D6110E0420C42FBEA40EF0EF3E0B0473FD7239DBE8DD4D9482631
AAB99697D0642E66486BF08123A9043A11E27AE7233831550134726D00A09AC7
6FEFC08E2EC42778B5C5EA76D5023AFBEA2F5C06CD4223DC078EDA2B1834
5D14A497096F36BB546A633B7BC10ABD482594B92D92BE1F2E437402CB3EF2
30648065CCD2D18FD273F52F93030F6D5C95324D3CC002EF0F92B0FDC437D6
B72D3E8AD8A8055F82BAB27E373553400B4CA776A92F18071F6B5F3F6D832E
694D016A306C73B79BADF66CFA1F90F99A53EC84DE4A913D75FE556473F964
13638C58C657C70312D82264EF7D6769E2529992D0EB8F72C4FFAF8A1E6F76B
A8ED43B8B61C620E9196E40607686E7347806A4208904617C2047F3E221591A31
B1D459E4730A00A85C619D928FFD18B85E3D21393CEEC454ECFD25EFE17C3
2E0BF0DBD2FE7EF8DBF4A5A907EB1B9B1598C632EFCC65982B01DFFA520
D9251E8A042EB72C2D8E15CC5C8A53590AE28E87D0CC11BD2CAB243B6C05
7FA19994223A135D0222068C2DD5975FC1908AF97F1169B2744B08B387629056A
2FBFA32A7ED64C3E0A82FC75F03D0ECA18CF6C299CB516A3E5059FC546648
E310CE5CE7A8D8E1CA980B3CFD1B805147A8F3DA4E48EF8856D7BFBE007C

D85492546245FECEA800

ss = C8846FBE83F88ABEE829A4D11A39B55611BCBC3578B220FA3C79BC384
15C06C7

8 Advantages and Limitations

In summary, we present practical post-quantum public-key crypto-systems based on a variant of the well-known learning parity with noise assumption that represents quantum hard problems from learning and coding theory. The LPN problem and its variants have been well studied but previously they were believed to be more of theoretical interests than to be relevant to any practical public-key cryptographic applications. We introduce new variants with structured noise distributions such that the hardness of the problem is at least preserved in a provable manner (by a reduction from standard LPN) and even boosted by log factor in the exponent (in respect of the best known cryptanalysis results). Supported by some non-trivial connections between LPN and Ring-LPN discovered along the way, we use ring heuristics to get further improved efficiency for the proposed crypto-systems.

Advantages. The proposed crypto-systems enjoy the following advantages:

- **Computationally efficient.** Similar to the large modulus analogue LWE, LPN and its variants possess good properties such as simplicity and parallelizability. Our Ring-CLPN based schemes appear to be comparably (and even slightly more) efficient than the Ring-LWE based counterparts, as already demonstrated by the software implementation (see Table 2 and Table 3). Further, thanks to the simplification to the binary field, the additions and multiplications directly translate to basic logical XOR and AND operations, and thus we expect our schemes to be more efficient when implemented in hardware and they should be quite friendly to low-power devices such as smart cards, IoT and even RFID devices.

- **Provably (quantum) secure.** The underlying problem (Ring-)CLPN is reducible from (Ring-)LPN with many connections to other variants such as LSPN and exact LPN, which have been well studied by the theoretical computer science community and are among the promising candidates resistant to quantum algorithms [19].

- **Physically secure.** (Ring-)(C)LPN has a linear structure and is therefore friendly to masking schemes as countermeasures against differential power analysis and other forms of side-channel attacks. We refer to [22] for the advantages of masking Ring-LPN based crypto-systems over AES.

Limitations. We mention also the following aspects in which lepton is less desirable than some post-quantum candidates.

- **Key/ciphertext sizes.** Depending on the target security level, Lepton needs public keys of $1-4$ KB and ciphertexts of $1.5-5.5$KB, which are roughly a couple of times larger than the counterparts of certain candidates [11].
- **Randomness.** The key generation and encryption processes need quite some amount of uniform random bits. A considerable percentage (more than half for our current implementation) of running time is spent on generating randomness (using SHAKE-128 and cSHAKE-128) and sampling noise vectors.

9 Acknowledgements

We are grateful to our co-authors in [52] for a collaboration in a CLPN-based PKE, and we also thank Juanru Li for implementing the memory consumption measurement platform.

References

1. Lightweight protocols: HB and its variations. `http://www.ecrypt.eu.org/ecrypt2/documents/D.SYM.5.pdf` (Section 3.1).
2. Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, Massachusetts, October 2003. IEEE.
3. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.
4. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009*, pages 595–618, 2009.
5. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In *Advances in Cryptology - CRYPTO 2007*, pages 92–110, 2007. Full version available at `http://www.eng.tau.ac.il/~bennyap/pubs/input-locality-full-revised-1.pdf`.
6. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, pages 520–536, 2012.
7. Elwyn Berlekamp, Robert J. McEliece, and H.C.A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
8. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In *Advances in Cryptology - CRYPTO 2011*, pages 743–760, 2011.
9. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *LNCS*, pages 278–291. Springer-Verlag, 22–26 August 1993.
10. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.
11. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. Crystals – kyber: a cca-secure module-lattice-based kem. Cryptology ePrint Archive, Report 2017/634, 2017. `http://eprint.iacr.org/2017/634`.
12. Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pages 1006–1018, 2016.
13. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy (SP 2015)*, pages 553–570, 2015.
14. David Cash, Eike Kiltz, and Stefano Tessaro. Two-round man-in-the-middle security from LPN. In *Proceedings of the 13th Theory of Cryptography (TCC 2016-A)*, pages 225–248, 2016.
15. Ivan Damgård and Sunoo Park. How practical is public-key encryption based on lpn and ring-lpn? Cryptology ePrint Archive, Report 2012/699, 2012. `http://eprint.iacr.org/2012/699`.
16. Ivan Djelic and Mark Borgerding. User bch encode/decode library based on bch module from linux kernel. `https://github.com/mborgerding/bch_codec`.
17. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)*, pages 355–374, 2012.
18. Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In *Advances in Cryptology – ASIACRYPT 2012*, pages 485–503, 2012.
19. Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In *Advances in Cryptology - CRYPTO 2017 - Part II*, pages 486–514, 2017.
20. Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *47th Symposium on Foundations of Computer Science*, pages 563–574, Berkeley, CA, USA, October 21–24 2006. IEEE.
21. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO 1999*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.
22. Lubos Gaspar, Gaëtan Leurent, and François-Xavier Standaert. Hardware implementation and side-channel analysis of lapin. In *The Cryptographer's Track at the RSA Conference 2014 (CT-RSA 2014)*, pages 206–226, 2014.

23. Elena Grigorescu, Lev Reyzin, and Santosh Vempala. On noise-tolerant learning of sparse parities and related problems. In *22nd International Conference on Algorithmic Learning Theory (ALT 2011)*, pages 413–424, 2011.
24. Qian Guo, Thomas Johansson, and Carl Löndahl. A new algorithm for solving ring-lpn with a reducible polynomial. *IEEE Trans. Information Theory*, 61(11):6204–6212, 2015.
25. Stefan Heyse. *Post Quantum Cryptography: Implementing Alternative Public Key Schemes on Embedded Devices*. PhD thesis, Ruhr-University Bochum, 2013. `https://www.emsec.rub.de/media/attachments/files/2014/03/thesis-stefan-heyse.pdf`.
26. Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-lpn. In *Fast Software Encryption - 19th International Workshop, FSE 2012*, pages 346–365, 2012.
27. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. *IACR Cryptology ePrint Archive*, 2017.
28. Dennis Hofheinz, Kathrin Hvelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. Cryptology ePrint Archive, Report 2017/604, 2017. `https://eprint.iacr.org/2017/604`.
29. Thomas Holenstein. Key agreement from weak bit agreement. In *STOC*, pages 664–673, Baltimore, Maryland, 22–24 May 2005.
30. Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In *Proceedings of the 3rd Theory of Cryptography Conference (TCC 2006)*, 2006.
31. Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*, pages 52–66, 2001.
32. Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147, 1995.
33. Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *Advances in Cryptology – ASIACRYPT 2012*, pages 663–680, 2012.
34. Shu jen Chang John Kelsey and Ray Perlner. Sha-3 derived functions: cshake, kmac, tuplehash and parallelhash. FIPS Special Publication 800-185, 2016. `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf`.
35. Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *Advances in Cryptology—CRYPTO 2005*, volume 3621 of *LNCS*, pages 293–308. Springer-Verlag, 14–18 August 2005.
36. Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the hb and hb^+ protocols. In Serge Vaudenay, editor, *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 73–87. Springer-Verlag, 2006.
37. Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise lpn. In Hugo Krawczyk, editor, *Public-Key Cryptography PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2014.
38. Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2011)*, pages 7–26, 2011.
39. Paul Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `http://eprint.iacr.org/2011/377`.
40. Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 43–62, 2015. full version at `https://eprint.iacr.org/2015/552.pdf`.
41. C.K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, and K. Hazelwood. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. In *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2005.
42. Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Proceedings of the 9th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM 2005)*, pages 378–389, 2005.
43. Vadim Lyubashevsky and Daniel Masny. Man-in-the-middle secure authentication schemes from lpn and weak prfs. In *Advances in Cryptology - CRYPTO 2013*, pages 308–325, 2013.
44. National Institute of Standards and Technology. Sha-3 standard: Permutation-based hash and extendable-output functions. FIPS PUB 202, 2015. `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf`.

45. Krzysztof Pietrzak. Cryptography from learning parity with noise. In *Proceedings of the Theory and Practice of Computer Science (SOFTSEM 2012)*, pages 99–114, 2012.
46. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
47. Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the fujisaki-okamoto and OAEP transforms. In *14th International Conference on Theory of Cryptography (TCC 2016-B)*, pages 192–216, 2016.
48. Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 11–20, 2012.
49. Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015.
50. Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise lpn. In *Advances in Cryptology – CRYPTO 2016*, pages 214–243, 2016.
51. Yu Yu and Jiang Zhang. The Lepton post-quantum crypto-system, 2017. More information at `http://yuyu.hk/pqc/` and source code at `https://github.com/whntgxtia/Lepton`.
52. Yu Yu, Jiang Zhang, Han Shuai, Weijia Wang, and Shuoyao Zhao. Practical post-quantum cryptography from a variant of learning parity with noise, 2017. `http://yuyu.hk/pqc/pqc-lpn.pdf`.

A Inequalities, Lemmata and Theorems

Lemma 5 (LPN $\leq$ HNF-LPN [4]). *Any algorithm $\mathcal{A}$ that runs in time T and breaks the computational (resp., decisional) (n,μ,q)-HNF-LPN with advantage ϵ implies another algorithm $\mathcal{A}'$ that runs in time $T' \approx T$ and breaks computational (resp., decisional) (n,μ,$q' = q + n$)-LPN problem with advantage $\Omega(\epsilon)$.*

Lemma 6 (Piling-up lemma). *For $0 < \mu < 1/2$ and random variables E_1, E_2, $\cdots$, E_ℓ that are i.i.d. to $\mathcal{B}_\mu$ we have*

$$\Pr\Big[\bigoplus_{i=1}^{\ell} E_i = 0\Big] = \frac{1}{2}(1 + (1-2\mu)^\ell) \ .$$

where $(1-2\mu)^\ell \approx e^{-2\mu\ell}$ for $\mu = o(1)$.

Lemma 7 ([42]). *If a bucket contains n balls, $(1/2 + p)n$ of which are colored white, and the rest colored black, and we select k balls at random without replacement, then the probability that we selected an even number of black balls is at least*

$$1/2 + \frac{1}{2}(\frac{2np - k + 1}{n - k + 1})^k \ .$$

Lemma 8. *For any $0 < \mu < 1/2$, and all sufficiently large $n \in \mathbb{N}$*

$$\sum_{i=0}^{n\mu} \binom{n}{i} = 2^{n\mathbf{H}(\mu) - \frac{\log n}{2} + O(1)} \ .$$

Lemma 9. *The following holds for all sufficiently large x*

$$1 - 2x^{-2} \ > \ \mathbf{H}(1/2 - x^{-1}) \ > \ 1 - 3x^{-2} \ .$$

Proof. It follows from Taylor expansion that

$$1 - 2 \cdot x^{-2} > \Big(\mathbf{H}(1/2 - x^{-1}) = 1 - (2\log e + o(1)) \cdot x^{-2}\Big) > 1 - 3 \cdot x^{-2} \ .$$

Lemma 10. *For $n \in \mathbb{N}$, let $b_n = \lceil \log(n) \rceil$, $n' = 2^{b_n}$ and $\delta = n' - n$. Consider a repeated experiment in which each time an element is selected from set $\{1, 2, \ldots, n'\}$ uniformly at random and such selection (with replacement) is repeated for $c_f \cdot k$ times. Let $\mathcal{S}$ be the set of elements that are no greater than n and are selected during the $c_f \cdot k$ times experiment. Then, we have*

$$\Pr\big[|\mathcal{S}| < k\big] < 2^{k\left(2\log(k+\delta) - b_n\right) - \log\left((k-1)!\right)} \ .$$

Proof. We have the following by a union bound:

$$\Pr\left[|\mathcal{S}| < k\right] < k \cdot \Pr\left[|\mathcal{S}| = k\right] < \underbrace{k \cdot \binom{n}{k}}_{=\frac{n(n-1)\cdots(n-k+1)}{(k-1)!} < \frac{n^k}{(k-1)!}} (\frac{k+\delta}{n'})^{c_f \cdot k} \ < \ \frac{n^k}{(k-1)!} \cdot (\frac{k+\delta}{n'})^{c_f \cdot k}$$
$$< 2^{k\left(c_f \log(k+\delta) - (c_f-1)b_n\right) - \log\left((k-1)!\right)} \ .$$

B Proofs Omitted in the Main Body

Proof of Lemma 1. It is easy to see that $r = p \cdot q$ is of degree up to $2n-2$ and has weight $|r| \leq |p| \cdot |q|$, so it suffices to show $|r \mod g| \leq \min(3|r|, 2|r| + m - 1)$. To this end, we define set $\mathcal{I} \subseteq \{0, 1, \cdots, 2n-2\}$ that corresponds to the non-zero terms in r, i.e., $r = \sum_{i \in \mathcal{I}} X^i$, and further write $r_i \stackrel{\text{def}}{=} X^i \mod g$. For $i \geq 2n - m$ we call X^i expansion-3 term, whose residue r_i is as follows (recall '+' denotes modulo-2 addition):

$$X^i = (X^{i-n} + X^{i+m-2n}) \cdot g + \underbrace{X^{i+2m-2n} + X^{i-n} + X^{i+m-2n}}_{r_i} \ ,$$

and for $n \leq i < 2n - m$ we call X^i expansion-2 term

$$X^i = X^{i-n} \cdot g + \underbrace{X^{i+m-n} + X^{i-n}}_{r_i}$$

and for $0 \leq i < n$ we have expansion-1 term X^i equal to residue r_i. We have the following general bound:

$$|r \mod g| = |\sum_{i \in \mathcal{I}} r_i| \leq \sum_{i \in \mathcal{I}} |r_i| \leq |\mathcal{I}| \max_{i \in \mathcal{I}}\{|r_i|\} \leq 3|r| \leq 3|p| \cdot |q| \ ,$$

where tightness is seen for $p = q = X^{n-1}$ as X^{2n-2} is a expansion-3 term. Next we proceed to proving the other bound. There is at most $(m-1)$ expansion-3 terms in $r = p \cdot q$, i.e., from degree $2n - m$ to $2n - 2$. Thus,

$$|r \mod g| = |\sum_{i \in \mathcal{I}} r_i| \leq 2(|r| - m + 1) + 3(m-1) \leq 2|r| + m - 1 \leq 2|p| \cdot |q| + m - 1 \ .$$

□

Proof of Lemma 3. The idea is to show that if Assumption 3 does not hold then a key agreement protocol depicted in Figure 3 is implied. Under assumption 2, i.e., $(\mathbf{B}, \mathbf{Bs}+\mathbf{e}_1)$ is computationally indistinguishable from $(\mathbf{B}, \mathbf{r} \stackrel{\$}{\leftarrow} \mathbb{F}_2^{nq})$ and thus passive security of the protocol is guaranteed. Further, we observe that

$$\mathbf{Bs} + \mathbf{e}_1 = (\mathbf{AX} + \mathbf{E})\mathbf{s} + \mathbf{e}_1 = \mathbf{A}(\underbrace{\mathbf{Xs}}_{\sim \mathcal{B}_\delta^n}) + (\underbrace{\mathbf{Es} + \mathbf{e}_1}_{\sim \mathcal{B}_\delta^{qn}}) \ ,$$

where $0.99\mu n \leq |\mathbf{s}| \leq 1.01\mu n$ except with probability $2^{-\Omega(\mu n)}$, and conditioned on any fixed s of weight roughly μn, $\mathbf{Xs}$ and $\mathbf{Es}+\mathbf{e}_1$ independently follow $\mathcal{B}_\delta^n$ and $\mathcal{B}_\delta^{qn}$ respectively by the definition of δ. In other words, $(\mathbf{A}, \mathbf{Bs} + \mathbf{e}_1)$ is a random instance of the "Hermite normal form" of (n,δ,q)-Ring-LPN. If Assumption 3 does not hold, there exist a PPT $\mathcal{D}$ and polynomial $p = \mathsf{poly}(n)$ such that the following holds for at least infinitely many n's:

$$\Pr[\mathcal{D}(\mathbf{A}, \mathbf{Bs} + \mathbf{e}_1) - 1] - \Pr[\mathcal{D}(\mathbf{A}, \mathbf{r} \stackrel{\$}{\leftarrow} \mathbb{F}_2^{nq}) = 1] \geq \frac{1}{p} \ .$$

This implies that Alice is able to decrypt m with at least noticeable probability:

$$\begin{aligned}\Pr[m'=m] &= \underbrace{\Pr[m=1]}_{1/2}\cdot\Pr[m'=1|m=1] + \underbrace{\Pr[m=0]}_{1/2}\cdot\underbrace{\Pr[m'=0|m=0]}_{1-\Pr[m'=1|m=0]} \\ &= 1/2 + (\Pr[m'=1|m=1] - \Pr[m'=1|m=0])/2 \\ &= 1/2 + \frac{\Pr[\mathcal{D}(\mathbf{A},\mathbf{Bs}+\mathbf{e}_1)-1] - \Pr[\mathcal{D}(\mathbf{A},\mathbf{r}\xleftarrow{\$}\mathbb{F}_2^{nq})=1]}{2} \\ &\geq 1/2 + \frac{1}{2p} \quad .\end{aligned}$$

Using parallel repetition and privacy amplification, it is known [29,30] that any protocol which achieves bit-agreement with noticeable correlation can be turned into a full-fledged key-agreement protocol (without increasing the number of rounds), which further implies a IND-CPA secure public-key encryption. □

Proof of Corollary 1. CLPN differs to LSPN in that CLPN uses a non-Bernoulli noise $\mathbf{e}\xleftarrow{\$}\chi_\mu^n$. But we argue that the substrings of χ_μ^n behaves quite close to Bernoulli distribution in terms of the bounded number of 1's, as formalized in Definition 3 and Fact 1. Recall that the LSPN solver [23] (see Theorem 3) takes sample complexity $q \ll n$, and any q-bit substring of $\mathbf{e}\xleftarrow{\$}\chi_\mu^n$ follows a Bernoulli-like distribution $\tilde{\mathcal{B}}_\eta^q$ for $\eta \leq \mu n/(n-q+1) \approx \mu$ (see Fact 2). Therefore, for $\mu = O(1/\sqrt{n})$ we just invoke the $(n,\mu,\eta\approx\mu)$-LSPN solver on the first (or any) q samples of CLPN which solves the CLPN problem with probability more than $1/2$ and time complexity

$$\mathsf{poly}(\frac{1}{1-2\eta})\cdot n^{\frac{(1+(2\eta)^2+o(1))\mu n}{2}} \approx n^{\frac{(1+o(1))\mu n}{2}} \quad .$$

□

Definition 3 (Bernoulli-like distribution). *A distribution of length ℓ and noise rate $0<\eta<1/2$, is called a (η,ℓ)-Bernoulli-like distribution, denoted by $\tilde{\mathcal{B}}_\eta^\ell$, if for any permutation π over $\{1,\cdots,\ell\}$, any $0\leq i\leq \ell-1$ and any value $v_i\in\mathbb{F}_2^i$ (v_i is empty when $i=0$) we have*

$$\Pr\left[\pi(\tilde{\mathcal{B}}_\eta^\ell)[i+1]=1 \middle| \pi(\tilde{\mathcal{B}}_\eta^\ell)[1,\cdots,i]=v_i\right] \leq \eta \ .$$

Fact 1 states that $\tilde{\mathcal{B}}_\eta^\ell$ behaves very similar to $\mathcal{B}_\eta^\ell$ in terms of bounded errors.

Fact 1 *For any $0\leq L\leq\ell$, it holds that $\Pr[|\tilde{\mathcal{B}}_\eta^\ell|\leq L]\geq\Pr[|\mathcal{B}_\eta^\ell|\leq L]$.*

Fact 2 (χ_μ^m is Bernoulli-like) *For any $m\geq\ell$, any ℓ-bit substring of χ_μ^m follows $\tilde{\mathcal{B}}_\eta^\ell$ for $\eta\leq\mu m/(m-\ell+1)$.*

Proof. Conditioned on any i-bit prefix (or substring) of χ_μ^m, the next bit satisfies:

$$\Pr\left[\chi_\mu^m[i+1]=1\middle|\chi_\mu^m[1,\cdots,i]=v_i\right]\leq\mu m/(m-i+1)\leq\mu m/(m-\ell+1) \ .$$

C Definitions

Definition 4 (Learning Parity with Noise). *The decisional LPN problem with secret length n and noise rate $0<\mu<1/2$, denoted by decisional (n,μ)-LPN, is hard if for every $q=\mathsf{poly}(n)$ and every PPT algorithm $\mathcal{D}$ we have*

$$\left|\Pr[\mathcal{D}(\mathbf{A},\ \mathbf{A}\cdot\mathbf{x}+\mathbf{e})=1]-\Pr[\mathcal{D}(\mathbf{A},\mathbf{r})=1]\right|=\mathsf{negl}(n) \ , \tag{3}$$

and the computational LPN problem with the same n and μ, denoted by computational (n,μ)-LPN, is hard if for every $q=\mathsf{poly}(n)$ and every PPT algorithm $\mathcal{D}$ we have

$$\Pr[\ \mathcal{D}(\mathbf{A},\ \mathbf{A}\cdot\mathbf{x}+\mathbf{e})=\mathbf{x}\ ] \ = \ \mathsf{negl}(n) \ , \tag{4}$$

where $q \times n$ matrix $\mathbf{A} \xleftarrow{\$} \mathbb{F}_2^{q\times n}$, $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^n$, $\mathbf{e} \xleftarrow{\$} \mathcal{B}_\mu^q$ and $\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^q$.
In certain scenarios such as PKE, the problem further puts a constraint on the number of samples, denoted by q, and thus the decisional and computational (n,μ,q)-LPN problems are similarly defined as above.

CONCRETE HARDNESS. For $T = T(n)$, we say that the decisional/computational (n,μ,q)-LPN is T-hard if every probabilistic adversary of running time T the distinguishing (resp., inverting) advantage in (3) (resp., (4)) is upper bounded by $1/T$. For (n,μ)-DLPN (whose q is unspecified) an implicit constraint is $q \leq T$.

Definition 5 (HNF-LPN). *The Hermite normal form of LPN is defined analogous to Definition 4 except that the secret $\mathbf{x}$ is also sampled from $\mathcal{B}_\mu^n$ instead of from uniform.*

Definition 6 (Ring-LPN [26]). *The ring version of the Learning Parity with Noise (Ring-LPN) problem is defined similar to Definition 1 except that matrix $\mathbf{A}$ is replaced with a number of ring elements. The decisional (n, μ)-LPN problem is hard if for every $q = \mathsf{poly}(n)$ and every PPT algorithm $\mathcal{D}$ it holds that*

$$\left| \Pr[\mathcal{D}(\mathbf{a}_1, \ldots, \mathbf{a}_q,\ \mathbf{a}_1\cdot\mathbf{x}+\mathbf{e}_1, \ldots, \mathbf{a}_q\cdot\mathbf{x}+\mathbf{e}_q) = 1] - \Pr[\mathcal{D}(\mathbf{a}_1, \ldots, \mathbf{a}_q, \mathbf{r}_1, \ldots, \mathbf{r}_q) = 1] \right| = \mathsf{negl}(n) \ , \tag{5}$$

and the computational (n, μ)-LPN problem is hard if for every $q = \mathsf{poly}(n)$ and every PPT algorithm $\mathcal{D}$ we have

$$\Pr[\ \mathcal{D}(\mathbf{a}_1, \ldots, \mathbf{a}_q,\ \mathbf{a}_1\cdot\mathbf{x}+\mathbf{e}_1, \ldots, \mathbf{a}_q\cdot\mathbf{x}+\mathbf{e}_q) = \mathbf{x}\ ] \ = \ \mathsf{negl}(n) \ , \tag{6}$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_q, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_q \xleftarrow{\$} R$, $\mathbf{x}, \mathbf{e}_1, \ldots, \mathbf{e}_q \xleftarrow{\$} \chi_\mu^R$. Throughout we consider only rings whose g is an irreducible trinomial over $\mathbb{F}_2$.

We recall the standard IND-CPA and IND-CCA security notions for PKE.

Definition 7 (IND-CPA Security for PKE). *A PKE scheme PKE is IND-CPA secure, if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in n:*

1. *On $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)$, $\mathcal{A}$ is given $(1^n, pk)$.*
2. *$\mathcal{A}$ outputs a pair of messages M_0 and M_1 of the same length. A random challenge bit $b \xleftarrow{\$} \{0, 1\}$ is sampled, and a challenge ciphertext $C \xleftarrow{\$} \mathsf{Enc}(pk, M_b)$ is generated and given to $\mathcal{A}$.*
3. *Finally, $\mathcal{A}$ outputs a guessing bit $b' \in \{0, 1\}$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{ind\text{-}cpa}_{\mathsf{PKE},\mathcal{A}}(n) := \left| \Pr[b' = b] - \frac{1}{2} \right|$.*

Definition 8 (IND-CCA Security for PKE). *A PKE scheme PKE is IND-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in n:*

1. *On $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)$, $\mathcal{A}$ is given $(1^n, pk)$, and has oracle access to $\mathsf{Dec}(sk, \cdot)$.*
2. *$\mathcal{A}$ outputs a pair of messages M_0 and M_1 of the same length. A random challenge bit $b \xleftarrow{\$} \{0, 1\}$ is sampled, and a challenge ciphertext $C \xleftarrow{\$} \mathsf{Enc}(pk, M_b)$ is generated and given to $\mathcal{A}$.*
3. *$\mathcal{A}$ continues to have oracle access to $\mathsf{Dec}(sk, \cdot)$, but is not allowed to query the challenge ciphertext C. Finally, $\mathcal{A}$ outputs a guessing bit $b' \in \{0, 1\}$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{ind\text{-}cca}_{\mathsf{PKE},\mathcal{A}}(n) := \left| \Pr[b' = b] - \frac{1}{2} \right|$.*

We also reproduce the formal definition of IND-CCA security for KEM.

Definition 9 (IND-CCA Security for KEM). *A KEM scheme KEM is IND-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in n:*

1. *On $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^n)$, $\mathcal{A}$ is given $(1^n, pk)$, and has oracle access to $\mathsf{Decaps}(sk, \cdot)$.*

2. *A random challenge bit* $b \xleftarrow{\$} \{0,1\}$ *is sampled. A challenge* $(C, K_0) \xleftarrow{\$} \mathsf{Encaps}(pk)$ *and* $K_1 \xleftarrow{\$} \mathcal{K}$ *are generated.* $\mathcal{A}$ *is given* (C, K_b) .
3. $\mathcal{A}$ *continues to have oracle access to* $\mathsf{Decaps}(sk, \cdot)$*, but is not allowed to query the challenge ciphertext* C*. Finally,* $\mathcal{A}$ *outputs a guessing bit* $b' \in \{0,1\}$*. The advantage of* $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{ind\text{-}cca}_{\mathsf{KEM},\mathcal{A}}(n) := \left| \Pr[b' = b] - \frac{1}{2} \right|$.