LARA: A Design Concept for Lattice-Based Encryption

Rachid El Bansarkhani

Technische Universität Darmstadt
QuantiCor Security GmbH, Germany
elbansarkhani@cdc.informatik.tu-darmstadt.de

Abstract Lattice-based encryption schemes still suffer from a low message throughput per ciphertext and inefficient solutions towards realizing enhanced security properties such as CCA1- or CCA2-security. This is mainly due to the fact that the underlying schemes still follow a traditional design concept and do not tap the full potentials of LWE. Furthermore, the desired security features are also often achieved by costly approaches or less efficient generic transformations. Recently, a novel encryption scheme based on the A-LWE assumption (relying on the hardness of LWE) has been proposed, where data is embedded into the error term without changing its target distributions. By this novelty it is possible to encrypt much more data as compared to the classical approach. In this paper we revisit this approach and propose several techniques in order to improve the message throughput per ciphertext. Furthermore, we present a very efficient trapdoor construction of reduced storage size. More precisely, the secret and public key sizes are reduced to just 1 polynomial, as opposed to $O(\log q)$ polynomials following previous constructions. Finally, we give an efficient implementation of the scheme instantiated with the new trapdoor construction. In particular, we attest high message throughputs and low ciphertext expansion factors at efficient running times. Our scheme even ensures CCA (or RCCA) security, while entailing a great deal of flexibility to encrypt arbitrary large messages or signatures by use of the same secret key.

Keywords: Lattice-Based Encryption, Lattice-Based Assumptions

1 Introduction

In [EDB15], a novel lattice-based encryption scheme has been proposed that encrypts data in a way that differs from previous constructions [Reg05, GPV08, Pei09, Pei10, ABB10, LP11, SS11, MP12] following the one-time-pad approach. It is equipped with many features such as a high message throughput per ciphertext as compared to current state-of-the-art encryption schemes while simultaneously ensuring different security notions (e.g. CCA security) for many cryptographic applications, for instance utilized for sign-then-encrypt scenarios or to securely transmit bundles of keys as required for the provisioning of remote attestation keys during manufacturing. Public key encryption schemes also represent important building blocks of advanced primitives such as group signature and ABS schemes. In many application scenarios it is also desired to ensure CCA1- or CCA2-security. The Augmented Learning with Errors problem (A-LWE) [EDB15], a modified LWE variant, has been introduced that allows to inject auxiliary data into the error term without changing the target distributions. In fact, the A-LWE problem has been proven to be hard to solve in the random oracle model assuming the hardness of LWE. Using a suitable trapdoor function as a black-box such as [EB14, MP12], the owner of the trapdoor is empowered to recover the secret resp. error-term and hence reveal the injected data. By this novelty, it is possible to exploit the error term as a container for the message or further information such as lattice-based signatures following the distributions of the error-term. It further encompasses a great deal of flexibility and other important properties such as CCA-security.

1.1 Our Contributions

In this paper we revisit the A-LWE problem and the implied encryption schemes from [EDB15]. In particular, we provide several theoretical improvements, introduce new tools, and give an efficient software implementation of the scheme testifying its conjectured efficiency. Below, we give an overview of features that can be realized by our scheme LARA (LAttice-based encryption of data embedded in RAndomness):

1. *Flexibility.* The encryptor of the scheme can increase the amount of encrypted data without invoking the encryption engine several times. Since the message is embedded into the error term, increasing the error size (to at most $\|\mathbf{e}_i\|_2 < q/4$ depending on the parameters) results in a higher message throughput. Thus, we achieve very low ciphertext expansion factors as compared to recent schemes. Furthermore, using a trapdoor allows to retrieve the secret and error polynomials for inspection. The retrieved secret polynomial could also play the role of a uniform random key for a symmetric key cipher.
2. *Signature embedding.* Due to the coinciding distributions of the error term and lattice-based signatures, the encryptor can exploit the signature as the error term. For instance, $(\mathbf{c}_2, \mathbf{c}_3)$ contains the signature on the error or message encrypted in $\mathbf{c}_1$. This offers an CCA2 like flavour as the decryptor can verify that the ciphertext has not been altered during transmission and the source of the data is authenticated via the signature. In case the size of the signature is too large, the encryptor can further exploit its flexibility.
3. *Security.* An increase of the error size already enhances the security of the scheme. However, it is also possible to further lift the security from CPA or CCA1 to RCCA or CCA2 almost for free via the transformations from [EDB15].

4. *Efficiency.* Due to the resemblance of ciphertexts to plain LWE samples, the efficiency of the scheme is very close to that required to generate ring-LWE samples, which intuitively seems to be a lower bound for many encryption schemes that are based on ring-LWE.

Improved Message Throughput. We introduce new techniques in order to increase the message throughput per ciphertext. In fact, we are able to exploit almost the full min-entropy of the error term to embed arbitrary messages. Previously, only one bit of the message was injected into a coefficient of the error term. By our new method, we are able to inject about $\log_2(\alpha q/\omega(\sqrt{\log n}))$ bits per entry for an error vector sampled according to the discrete Gaussian distribution with parameter αq. Encoding and decoding of the message requires only to reduce the coefficients modulo some integer.

$m = c \cdot nk$ $k = \log q$	CCA [**MP12**]	CPA/CCA [**EDB15**]	CPA/CCA **This work**	CCA **This work** + [**MP12**]	CPA [**LP11**], **others**
Ciphertext size	$m \cdot k$	$m \cdot k$	$m \cdot k$	$m \cdot k$	$m \cdot k$
Signature size	nk	$c\log(\alpha q)nk$	$c\log(\alpha q)nk$	$(c\log(\alpha q)+1)nk$	$cnk - n$
Message size	nk	$c \cdot nk$	$c \cdot nk\log(\alpha q/4.7)$	$(c\log(\alpha q/4.7)+1)nk$	$cnk - n$
Message Exp.	$c \cdot k$	k	$\frac{k}{\log(\alpha q/4.7)}$	$\frac{k}{\log(\alpha q/4.7)+1/c}$	$k + \frac{k}{ck-1}$

Following this approach we can revise the parameters from [EDB15] according to the table above. When comparing our approach with the CPA-secure encryption scheme from Lindner and Peikert [LP11] and other recently proposed schemes, we attest an improvement factor of at least $O(\log(\alpha q))$.

Improved Trapdoors, Scheme Instantiation and Security. We give an improved construction of trapdoors in the random oracle model, which allows to significantly reduce the number of ring elements in the public key by a factor $O(\log q)$, hence moving trapdoor constructions towards practicality. More precisely, we give an improved construction of trapdoor algorithms ($\mathsf{TrapGen}$, LWEGen, LWEInv), in case the secret vector is sampled uniformly at random and can thus be selected $\mathbf{s} = F(\mathbf{r}, H(\mathbf{r}))$ involving a deterministic function F and a cryptographic hash function H modeled as a (quantum-) random oracle. This is a crucial ingredient of our construction and the resulting schemes. In particular, we achieve public and secret keys each consisting only of 1 polynomial. Hence, our construction improves upon previous proposals, where the public key contains at least $\lceil \log q \rceil$ polynomials (matrix dimension in [MP12] is $n \times n(1 + \lg q)$, see also [LPR13]), and is thus comparable with the public key size used in current state-of-the-art encryption schemes. This makes the usage of trapdoor based constructions more attractive for practice as it provides direct access to the secret and error vectors, which can be exploited in many different ways and at least for inspection.

Implementation and Analysis. In order to attest the conjectured efficiency of our scheme that we call $\mathsf{LARA}_{\mathsf{CPA}}$ or $\mathsf{LARA}_{\mathsf{CCA}}$, we implement the (quantum-) random oracle variants of our CPA- and CCA-secure schemes in software. This implementation is optimized with respect to the underlying architecture. To this end, we applied optimized techniques for discrete Gaussian sampling and FFT multiplication, the core elements governing the efficiency of the scheme. In particular, we adopt several optimizations for the polynomial representation and polynomial multiplication by use of efficient FFT/NTT operations. We implement our scheme and compare it with various schemes. For our reference implementation and $n = 1024$ (conservative parameters), we attest running times of 418 000 cycles for encryption and about 289 000 cycles for decryption in the CPA-secure setting. Thus, in comparison to the other schemes, we achieve by our improved trapdoor construction high message throughputs at low ciphertext expansion factors and at efficient running times and key sizes. The AVX-implementation is about twice as fast.

1.2 Acknowledgements

The work presented in this paper was performed within the context of the project P1 within the CRC 1119 CROSSING. We thank Douglas Stebila for his useful comments.

1.3 Organization

This paper is structured as follows. Section 2 provides the relevant background of our work. In Section 3 we introduce the A-LWE problem from [EDB15] and present our improvements to enhance the message throughput. In Section 4 a description of new trapdoor algorithms is proposed. The resulting encryption schemes are detailed in Section 5. In Section 6 we present our software implementation and experimental results.

2 Preliminaries

Notation We will mainly focus on polynomial rings $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ for integers $q > 0$ and n being a power of two. We denote ring elements by boldface lower-case letters e.g. $\mathbf{p}$, whereas for vectors of ring elements we use $\hat{\mathbf{p}}$ and upper-case bold letters for matrices (e.g., $\mathbf{A}$). By $\oplus$ we denote the XOR operator.

Discrete Gaussian Distribution We define by $\rho : \mathbb{R}^n \to (0, 1]$ the n-dimensional Gaussian function

$$\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\cdot\frac{\|\mathbf{x}-\mathbf{c}\|_2^2}{s^2}}, \ \forall \mathbf{x}, \mathbf{c} \in \mathbb{R}^n .$$

The discrete Gaussian distribution $\mathcal{D}_{\Lambda+\mathbf{c},s}$ is defined to have support $\Lambda + \mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^n$ and $\Lambda \subset \mathbb{R}^n$ is a lattice. For $\mathbf{x} \in \Lambda + c$, it basically assigns the probability $\mathcal{D}_{\Lambda+\mathbf{c},s}(\mathbf{x}) = \rho_s(\mathbf{x})/\rho_s(\Lambda + c)$.

Lattices. Throughout this paper we are mostly concerned with q-ary lattices $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$, where $q = poly(n)$ denotes a polynomially bounded modulus and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is an arbitrary matrix. $\Lambda_q^\perp(\mathbf{A})$ resp. $\Lambda_q(\mathbf{A})$ are defined by

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv \mathbf{0} \mod q\}$$
$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}^m \text{ s.t. } \mathbf{x} = \mathbf{A}^\top \mathbf{s} \mod q\}\,.$$

Definition 1. *For any n-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \backslash \{0\}) \le \epsilon$.*

Lemma 1 ([Ban95, Lemma 2.4]). *For any real $s > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have*

$$P[|\langle \mathbf{x}, \mathcal{D}_{\mathbb{Z}^n,s}\rangle| \ge T \cdot s\,\|\mathbf{x}\|] < 2exp(-\pi \cdot T^2)\,.$$

Lemma 2 ([GPV08, Theorem 3.1]). *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{S}$, and let $\epsilon > 0$. We have $\eta_\epsilon(\Lambda) \le \| \tilde{\mathbf{S}} \| \cdot \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$. In particular, for any function $\omega(\sqrt{\log n})$, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \le \| \tilde{\mathbf{S}} \| \cdot \omega(\sqrt{\log n})$.*

Corollary 1 ([DM14, Corollary 4]). *Let $n \ge 4$ be a power of two, $q \ge 3$ a power of 3, and set $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, then any nonzero polynomial $\mathbf{t} \in \mathcal{R}_q$ of degree $d < n/2$ and coefficients in $\{0, \pm 1\}$ is invertible in $\mathcal{R}_q$.*

Definition 2 (LWE Distribution). *Let n, m, q be integers and χ_e be distribution over $\mathbb{Z}$. By $L^{\mathsf{LWE}}_{n,m,\alpha q}$ we denote the LWE distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, which draws $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n\times m}$ uniformly at random, samples $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ and returns $(\mathbf{A}, \mathbf{b}^\top) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$.*

Definition 3 (LWE Problem). *Let $\mathbf{u} \in$ be uniformly sampled from $\mathbb{Z}_q^m$.*
- *The decision problem of LWE asks to distinguish between $(\mathbf{A}, \mathbf{b}^\top) \leftarrow L^{\mathsf{LWE}}_{n,m,\alpha q}$ and $(\mathbf{A}, \mathbf{u}^\top)$ for a uniformly sampled secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*
- *The search problem of LWE asks to return the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ given an LWE sample $(\mathbf{A}, \mathbf{b}) \leftarrow L^{\mathsf{LWE}}_{n,m,\alpha q}$ for a uniformly sampled secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*

3 Augmented Learning with Errors

In this section, we give a description of the message embedding approach as proposed in [EDB15] and how it is used in order to inject auxiliary data into the error term of LWE samples. This feature represents the main building block of the generic encryption scheme from [EDB15], which allows to encrypt huge amounts of data without increasing the ciphertext size. In fact, it is even possible to combine this concept with the traditional one-time-pad approach in order to take the best from both worlds and hence increase the message size per ciphertext at almost no cost.

Lemma 3 ([EDB15, Statistical]). *Let $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$ be an arbitrary full-rank matrix and $\epsilon = \mathsf{negl}(n)$. The statistical distance $\Delta(\mathcal{D}_{\mathbb{Z}^m,r}, \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r})$ for uniform $\mathbf{v} \leftarrow_R \mathbb{Z}_p^n$ and $r \ge \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ is negligible.*

Lemma 4 ([EDB15, Computational]). *Let $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$ be an arbitrary full-rank matrix. If the distribution of $\mathbf{v} \in \mathbb{Z}_p^n$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_p^n$, then $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ is computationally indistinguishable from $\mathcal{D}_{\mathbb{Z}^m,r}$ for $r \ge \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$.*

3.1 Message Embedding

The proposed technique aims at embedding auxiliary data into the error term $\mathbf{e}$ such that it still follows the required error distibution. In particular, Lemma 3 and 4 are used, which essentially state that a discrete Gaussian over the integers can be sampled by first picking a coset $\Lambda_{\mathbf{c}}^{\perp}(\mathbf{B}) = \mathbf{c} + \Lambda_{p}^{\perp}(\mathbf{B})$ uniformly at random for any full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n \times m}$ and then invoking a discrete Gaussian sampler outputting a preimage $\mathbf{x}$ for $\mathbf{c}$ such that $\mathbf{B} \cdot \mathbf{x} \equiv \mathbf{c} \bmod p$ However, this requires the knowledge of a suitable basis for $\Lambda_q^{\perp}(\mathbf{B})$. In fact, the random coset selection can be made deterministic by means of a random oracle H taking a random seed with enough entropy as input.
The fact that xoring a message $\mathbf{m}$ to the output of H does not change the distribution, allows to hide the message within the error vector without changing its target distribution. As a result, we obtain $\mathbf{e} \leftarrow D_{\Lambda^{\perp}_{H(\mu)\oplus \mathbf{m}}(\mathbf{B}),r}$, which is indistinguishable from $D_{\mathbb{Z}^m,r}$ for a random seed μ and properly chosen parameters (see Lemma 3 and Lemma 4). Subsequently, based on the message embedding approach the Augmented LWE problem (A-LWE) has been introduced, where A-LWE samples resemble ordinary LWE instances except for the modified error vectors. In particular, the A-LWE problem is specified with respect to a specific matrix $\mathbf{G}$, which allows to efficiently sample very short vectors according to the discrete Gaussian distribution. We note that other choices are also possible as long as the parameter of the error vectors exceed the smoothing parameter of the associated lattice. We now give a generalized description of the A-LWE distribution using any preimage sampleable public matrix $\mathbf{B}$.

Definition 4 (Augmented LWE Distribution). *Let $n, n', m, m_1, m_2, k, q, p$ be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^{\perp}(\mathbf{B}))$. Let $H : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^{n' \cdot \log(p)}$ be a cryptographic hash function modeled as random oracle. Let $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable full-rank matrix (such as $\mathbf{B} = \mathbf{G}$ from [MP12]). For $\mathbf{s} \in \mathbb{Z}_q^n$, define the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ with $\mathbf{m} \in \{0,1\}^{n' \log p}$ to be the distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ obtained as follows:*

1. *Sample $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.*
2. *Set $\mathbf{v} = \mathsf{encode}(H(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$.*
3. *Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),\alpha q}$.*
4. *Return $(\mathbf{A}, \mathbf{b}^\top)$ where $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.*

We note that the Step 3 returns a discrete Gaussian that is distributed as $\mathcal{D}_{\mathbb{Z}^{m_2},\alpha q}$ follwoing [EDB15, Computational]. In principal, for A-LWE one differentiates the decision problem $\mathsf{decision\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ from the corresponding search problem $\mathsf{search\text{-}s\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$, as known from LWE. Furthermore, there exists a second search problem $\mathsf{search\text{-}m\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$, where a challenger is asked upon polynomially many A-LWE samples to find in polynomial time the message $\mathbf{m}$ injected into the error vector. Note that the error distribution could also differ from the discrete Gaussian distribution. For instance, one could use the uniform distribution, for which one obtains similar results. All the proofs from [EDB15] go through without any modifications, since the security proofs are not based on the choice of $\mathbf{B}$.

Theorem 1 (adapted [EDB15]). *Let n, n', m, m_1, m_2, q, p be integers with $m = m_1 + m_2$. Let H be a random oracle. Let $\alpha q \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{B}))$ for a real $\epsilon = \mathsf{negl}(\lambda) > 0$ and preimage sampleable public matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$. Furthermore, denote by χ_s and χ_{e_1} the distributions of the random vectors $\mathbf{s}$ and $\mathbf{e}_1$ involved in each A-LWE sample. If $\mathbb{H}_\infty(\mathbf{s}, \mathbf{e}_1) > \lambda$, then the following statements hold.*

1. *If* search $\mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* search-s $\mathsf{A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*
2. *If* decision $\mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* decision $\mathsf{A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*
3. *If* decision $\mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* search-m $\mathsf{A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*

One easily notes, that these hardness results also hold for the ring variant (see [ABBK17]). We remark that for encryption schemes the secret $\mathbf{s}$ is always resampled such that $H(\mathbf{s})$ suffices to output a random vector and the complete bandwidth of $\mathbf{e}$ is exploited for data to be encrypted.

3.2 Improved Message Embedding

For the sake of generality, we used in all our statements an abstract matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ for integers p, n', and m. This is used to embed a message into the error term via $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),\alpha q}$, where $\mathbf{v} = \mathsf{encode}(H(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$ is uniform random. However, we can specify concrete matrices that optimize the amount of information per entry with respect to the bound given in Lemma 2. We propose several techniques in order to enhance the message throughput per discrete Gaussian vector. These techniques could also be applied to the error vector involved in the A-LWE distribution. In other words, we aim at choosing an appropriate preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ such that $n' \cdot \log p$ is maximized. For now, we will focus on how to apply this technique to the different encryption schemes and omit the term $\mathbf{e}_1$ when invoking the random oracle, since the secret $\mathbf{s} \in \mathbb{Z}_q^n$ is always resampled in encryption schemes and hence provides enough entropy for each fresh encryption query. The first approach is based on a method used to construct homomorphic signatures in [BF11]. We also propose a simpler approach that avoids such complex procedures while entailing the same message throughput.

Intersection Method. The intersection method as proposed in [BF11] considers two m-dimensional integer lattices Λ_1 and Λ_2 such that $\Lambda_1 + \Lambda_2 = \mathbb{Z}^m$, where addition is defined to be element-wise. Therefore, let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two messages, where $\mathbf{m}_1$ and $\mathbf{m}_2$ define a coset of Λ_1 and Λ_2 in $\mathbb{Z}^m$, respectively. As a result, the vector $(\mathbf{m}_1, \mathbf{m}_2)$ defines a unique coset of the intersection set $\Lambda_1 \cap \Lambda_2$ in $\mathbb{Z}^m$. By the Chinese Remainder theorem one can compute a short vector $\mathbf{t}$ such that $\mathbf{t} = \mathbf{m}_1 \bmod \Lambda_1$ and $\mathbf{t} = \mathbf{m}_2 \bmod \Lambda_2$ using a short basis for $\Lambda_1 \cap \Lambda_2$. In fact, it is easy to compute any vector $\mathbf{t}$ that satisfies the congruence relations. Subsequently, by invoking a preimage sampler one obtains a short vector from $\Lambda_1 \cap \Lambda_2 + \mathbf{t}$.

Lattices of the Form $\mathbf{p\mathbb{Z}^m}$. One realizes that for a given parameter αq for the distribution of the error vector one can be much more efficient, if one considers only the lattice $\Lambda_p^{\perp}(\mathbf{I}) = p\mathbb{Z}^m$. In this case, the message space is simply defined by the set $\mathcal{M} = \mathbb{Z}^m / \Lambda_p^{\perp}(\mathbf{I}) \cong \mathbb{Z}_p^m$. When comparing with the previous approach, for instance, it is only required to increase p by a factor of 2 in order to obtain the same message throughput $m \log 2p = m \cdot (\log p + 1)$. Furthermore the decoding and encoding phase is much faster, since encoding requires only to sample $\mathbf{e} \leftarrow D_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$ for $\mathbf{b} = H(\mathbf{r}) \oplus \mathbf{m}$ using fast discrete Gaussian samplers such as the Knuth-Yao algorithm or efficient lookup tables. Decoding is performed via $H(\mathbf{r}) \oplus (\mathbf{e} \bmod p)$. Optimizing the message throughput requires to increase p such that $\eta_\epsilon(\Lambda) \leq p \cdot \mathsf{const} \leq \alpha q$ still holds for $\mathsf{const} = \sqrt{\ln(2(1+1/\epsilon))/\pi}$. Doing this, one can embed approximately $m \cdot \log p$ bits of data, which almost coincides with the min-entropy of a discrete Gaussian with parameter αq, since $\mathsf{const} \approx 4.7$. Therefore, one prefers to choose a parameter $\alpha q = p \cdot \mathsf{const}$ with $p = 2^i$ and integer $i > 0$ in order to embed i bits of data into the error term.

Uniform Error. For uniformly distributed errors one can directly employ the output of the random function $H(\cdot)$ as the error term. More specifically, suppose $\mathbf{e} \in ([-p,p] \cap \mathbb{Z})^m$, then let $H(\cdot) : \{0,1\}^* \to ([-p,p] \cap \mathbb{Z})^m$ be a random function (e.g. RO) such that $\mathbf{e} \leftarrow \mathsf{encode}(H(\mathbf{r}) \oplus \mathbf{m})$ for $\mathbf{m} \in \{0,1\}^{m \log_2(2p)}$. As a result, one can use the full bandwidth of the error term and inject $m \log_2(2p)$ message bits.

4 New Trapdoor Algorithms for Ideal-Lattices

In [EB14] a generic approach of how to instantiate the trapdoor construction is given that allows to retrieve the error term and the secret vector from A-LWE instances. However, the number of public key polynomials is with $\bar{m} + k$ polynomials where $k = \lceil \log q \rceil$ rather large and hence not suitable for practice. In fact, the trapdoor constructions [EB14, MP12] require at least 2 public key polynomials in order to generate signatures. For encryption, one requires even more as the LWE inversion algorithm has to efficiently recover the correct secret. Thus, a new approach is needed in order to tackle this issue.

In this section, we give new trapdoor algorithms and show how to reduce the size of the public key to just 1 polynomial. This is due to the fact that we can select the secret vector in A-LWE instances to be of the form $\mathbf{s} = F(\mathbf{r}, H_1(\mathbf{r}))$ for a deterministic function $F(\cdot)$, where $\mathbf{r}$ is a random bit string and H_1 is a cryptographic hash function modeled as RO. Remarkably, the secret key consists only of 1 polynomial, which improves upon the construction from [MP12, EB14]. We start with a description of our new trapdoor algorithms in the ring setting $\mathcal{K} = (\mathsf{TrapGen}, \mathsf{LWEGen}, \mathsf{LWEInv})$. Lemma 5 shows that $\mathsf{TrapGen}$ outputs a public key that is computationally indistinguishable from uniform random. In order to use tags for CCA-secure constructions, we need to modify the way, in which tags are applied.

4.1 Construction of Efficient Trapdoors for A-LWE

We present new trapdoor algorithms for public key generation ($\mathsf{TrapGen}$), ring-LWE generation (LWEGen) and inversion (LWEInv). These algorithms will serve to instantiate our new encryption scheme from ring A-LWE. For the sake of simplicity, we only consider the case where $q = p^k$, where p is any positive (prime) integer.

1. $\underline{\mathsf{TrapGen}(1^n)}$: Let $q = p^k$ for a prime integer $p > 0$. Let further $\mathbf{g} = p^{k-1}$. The system parameters are two uniform random polynomials $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q$ (e.g. sampled from a seed). Sample 2 random polynomials $\mathbf{r}_i$ according to $\mathcal{D}_{\mathbb{Z}^n, r_{sec}}$ for $i \in \{1, 2\}$. The public and secret keys are given by $\mathsf{pk} := \mathbf{a}_3,\ \mathsf{sk} = [\mathbf{r}_1, \mathbf{r}_2]$ with
$$\mathbf{A} = \left[\mathbf{a}_1, \mathbf{a}_2, \underbrace{\mathbf{g} - (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2)}_{\mathbf{a}_3}\right] \in \mathcal{R}_q^3.$$
If a tag $\mathbf{t}_u$ is applied, we obtain $\mathbf{A}_u$ via $\mathbf{t}_u \cdot \mathbf{g}$ (see below).

2. $\underline{\mathsf{LWEGen}(1^n)}$: In order to generate an (A-)LWE instance, we let H_1 be a cryptographic hash function modeled as a random oracle. For $\mathbf{A}$ we generate ring-LWE instances
$$[\mathbf{b}_1,\ \mathbf{b}_2,\ \mathbf{b}_3] = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{g} - (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2)] \cdot \mathbf{s} + \hat{\mathbf{e}} \in \mathcal{R}_q^3\,.$$
Each coefficient of $\mathbf{s} \in \mathcal{R}_q$ is of the form
$$s_i = c_{i,0} + c_{i,1} \cdot p + ... + c_{i,k-1} \cdot p^{k-1}$$
for $c_{i,j} \in \{0, \ldots, p-1\}$ and $i \in \{1, \ldots, n\}$, where $c_{i,0} \leftarrow_R \{0, \ldots, p-1\}$ is sampled uniformly at random. Then, invoke $\mathbf{d} = H(c_{1,0}, \ldots, c_{n,0}) \to \mathbb{Z}^n_{p^{k-1}}$ and set $s_i = c_{i,0} + p \cdot d_i$.
   - For the special case $\mathbf{q = 2^k}$, the binary number $c_{i,0}$ corresponds to the least significant bit of the coefficient s_i. That is $\mathsf{LSB}(s_i) \leftarrow_R \{0, 1\}$, where LSB denotes the least significant bit. Then, in order to set the remaining bits of s_i invoke $\mathbf{d} = H_1(\mathsf{LSB}(s_1), \ldots, \mathsf{LSB}(s_n)) \in \mathbb{Z}^n_{2^{k-1}}$. Finally, determine $s_i = \mathsf{LSB}(s_i) + 2 \cdot d_i \in \mathbb{Z}_q$ by appending the bit $c_{i,0}$ to d_i.

   The error polynomials $\mathbf{e}_i$ can now be sampled from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n, \alpha q}$, where $1 \leq i \leq 3$ and $\alpha q > 0$.

3. $\underline{\mathsf{LWEInv}(\hat{\mathbf{b}}, \mathsf{sk})}$: We first compute
$$\mathbf{v} = \mathbf{g} \cdot \mathbf{s} + \mathbf{t} = \mathbf{b}_3 + \mathbf{b}_1 \cdot \mathbf{r}_1 + \mathbf{b}_2 \cdot \mathbf{r}_2,$$
where $\mathbf{t}$ is a some small error.
The closest integer $c_{i,0} \cdot p^{k-1}$ to each coefficient v_i is recovered. This is possible if $|t_i| < p^{k-1}/2$. In particular, we recover $c_{i,0}$ via
$$c_{i,0} = \lfloor v_i / p^{k-1} \rceil \bmod p \text{ for } 0 \leq i < n$$

- For $\mathbf{q} = 2^k$, we have $c_{i,0} = \mathsf{LSB}(s_i) = \lfloor v_i/2^{k-1} \rceil \bmod 2\,$.
  Once having recovered all $c_{i,0}$, the hash function is invoked $\mathbf{d} = H_1(c_{1,0}, \ldots, c_{n-1,0}) \in \mathbb{Z}^n_{p^{k-1}}$ such that $s_i = c_{i,0} + d_i \cdot p$. The error vector is subsequently retrieved via $\hat{\mathbf{e}} = \hat{\mathbf{b}} - \mathbf{A} \cdot \mathbf{s}$.

Remark 1. For odd q and small secrets, we can instead set $\mathbf{g} = (q-1)/2\,$. The most significant bits do not vanish but wrap around modulo q. We note, that the case $q = 2^k$ is very efficient due to cheap sampling and modulo operations.

Lemma 5. *Let $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q$ be uniform random polynomials and $\mathbf{r}_1, \mathbf{r}_2$ be sampled according to $\mathcal{D}_{\mathcal{R},\alpha q} = \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ (via the coefficient embedding) for $\alpha q > 2\sqrt{n}$. The public key*

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2]$$

is computationally indistinguishable from uniform.

Proof. For simplicity, we can assume that $\mathbf{a}_1$ is a unit in $\mathcal{R}_q$, since the ring of units $\mathcal{R}_q^\times$ represents a non-negligible subset of $\mathcal{R}_q$ for the rings in consideration. Then

$$\mathbf{A} = \mathbf{a}_1 \cdot [\mathbf{1}, \bar{\mathbf{a}}, \bar{\mathbf{a}} \cdot \mathbf{r}_2 + \mathbf{r}_1]\,,$$

where $[\bar{\mathbf{a}}, \bar{\mathbf{a}} \cdot \mathbf{r}_2 + \mathbf{r}_1]$ is a ring-LWE instance with a uniform random polynomial $\bar{\mathbf{a}} = \mathbf{a}_1^{-1}\mathbf{a}_2$, since $\mathbf{a}_2$ is uniform random. As a result and due to the independence of $\mathbf{a}_1$ from $\mathbf{a}_2$ the claim follows. $\square$

Lemma 6 (Correctness). *For $q = p^k$, error polynomials $\mathbf{e}_i$ and secret key polynomials $\mathbf{r}_j$, the algorithm $\mathsf{LWEInv}(\hat{\mathbf{b}}, \mathit{sk})$ correctly inverts the (A-)LWE instance, if*

$$\|\mathbf{e}_3 + \mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2\|_\infty < p^{k-1}/2\,.$$

Proof. The inversion algorithm computes

$$\begin{aligned}
\mathbf{b}_3 + \mathbf{b}_1 \cdot \mathbf{r}_1 + \mathbf{b}_2 \cdot \mathbf{r}_2 \bmod q &= \mathbf{g} \cdot \mathbf{s} + \mathbf{e}_3 + \mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2 \bmod q \\
&= p^{k-1} \cdot \mathbf{s} + \mathbf{e}_3 + \mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2 \bmod q \\
&= p^{k-1} \cdot \begin{bmatrix} c_{1,0} \\ \vdots \\ c_{n,0} \end{bmatrix} + \mathbf{e}_3 + \mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2 \bmod q
\end{aligned}$$

So, if $\|\mathbf{e}_3 + \mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2\|_\infty < p^{k-1}/2$, then we cleary can recover $c_{i,0}$ of each coefficient. From the coefficients we can recover $\mathbf{s}$ and $\mathbf{e}_j$ and thus the message. $\square$

Tagging the public key (in order to achieve CCA security) in the ring setting is accomplished similar to [MP12, EDB15], but with some practical obstacles to be solved for decryption. This is due to the random oracle instantiation, which prevents from recovering the tag $\mathbf{t}_u$ in a straightforward way, because the inversion algorithm only recovers $c_{i,0}$ (for $q = p^k$) of the coefficients from $\mathbf{t}_u \cdot \mathbf{s}$. However, via a trick we can circumvent this obstacle in a computationally indistinguishable way. This is mainly possible, since $\mathbf{t}_u$ is a unit and multiplication with a uniform random polynomial is again uniform. Thus, we can instead generate $\mathbf{t}_u \cdot \mathbf{s} := c_{i,0} + p \cdot d_i$ in LWEGen and cancel out $\mathbf{t}_u$ from it via its inverse when $\mathbf{s}$ is required. Here, we denote $\mathbf{A}_u = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{t}_u \cdot \mathbf{g} - (\mathbf{a}_1\mathbf{r}_1 + \mathbf{a}_2\mathbf{r}_2)]$ in accordance to Section 4.

5 Public Key Encryption

In order to build a public key encryption scheme we need to combine the trapdoor construction described in Section 4 with the message embedding approach from Section 3. The main idea is to inject data to be encrypted into the error polynomials from LWEGen. To this end, we need the error terms to be partially deterministic and simultaneously look random by use of a random oracle.

```
$\mathsf{LARA.KGen}(1^\lambda)$
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{TrapGen}(1^\lambda)$
2: We replace $\mathbf{a}_1, \mathbf{a}_2$ in the public key with the $\mathsf{seed}$ generating those elements,
   i.e. $\mathsf{pk} = (\mathsf{seed}, \mathbf{a}_3)$.

$\mathsf{LARA.Enc}(\mathsf{pk}, (\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3) \in \{0,1\}^{3\mathsf{n}\log \mathsf{w}})$
1: $\mathbf{a}_1, \mathbf{a}_2 \leftarrow \mathsf{G}(\mathsf{seed})$
2: $\mathbf{c}_0 = (c_{1,0}, \ldots, c_{n,0}) \leftarrow \mathbb{Z}_p^n$
3: $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3), \mathbf{d} \leftarrow \mathsf{H}(\mathbf{c}_0) \in \mathbb{Z}_{\mathsf{w}}^{3\mathsf{n}} \times \mathbb{Z}_{\mathsf{p}^{\mathsf{k}-1}}^{\mathsf{n}}$
4: $\mathbf{s} = \mathbf{c}_0 + p \cdot \mathbf{d}$
5: $\mathbf{t}_i = \mathsf{Encode}(\mathsf{m}_\mathsf{i}) + \mathbf{v}_\mathsf{i} \bmod \mathsf{w}$ for $1 \leq \mathsf{i} \leq 3$
6: $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbf{t}_i + \mathsf{w}\cdot\mathbb{Z}^\mathsf{n}, \mathsf{s}}$ for $1 \leq i \leq 3$
7: Output $\mathbf{b}_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i$ for $1 \leq i \leq 3$

$\mathsf{LARA.Dec}(\mathsf{sk}, \mathbf{b} \in \mathcal{R}_\mathsf{q}^3)$
1: $\mathbf{a}_1, \mathbf{a}_2 \leftarrow \mathsf{G}(\mathsf{seed})$
2: $(\mathbf{s} := \mathbf{c}_0 + p \cdot \mathbf{d}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \leftarrow \mathsf{LWEInv}(\mathsf{sk}, \mathbf{b})$
3: $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3), \mathbf{d} \leftarrow \mathsf{H}(\mathbf{c}_0) \in \{0,1\}^{3\mathsf{n}\log \mathsf{w}} \times \mathbb{Z}_{\mathsf{p}^{\mathsf{k}-1}}^{\mathsf{n}}$
4: $\mathbf{t}_i = \mathbf{e}_i \bmod \mathsf{w}$ for $1 \leq i \leq 3$
5: $\mathsf{m}_\mathsf{i} = \mathsf{Decode}(\mathbf{t}_\mathsf{i} - \mathbf{v}_\mathsf{i} \bmod \mathsf{w})$
6: Output $(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3)$
```

Fig. 1: Description of the CPA-secure Encryption Scheme.

We now give a description of our new CPA-secure public key encryption scheme. Thus, let $s = \mathsf{w} \cdot \sqrt{\ln\left(2\mathsf{n}\left(1 + \frac{1}{\epsilon}\right)\right)/\pi}$ for an integer $\mathsf{w} > 2$. Hence, we embed the message into the cosets of the lattice $\mathsf{w} \cdot \mathbb{Z}^\mathsf{n}$ (see Section 3.2). For key generation, the CPA-secure scheme just invokes $\mathsf{TrapGen}$. For a compact scheme, we let the uniform random polynomials $\mathbf{a}_1, \mathbf{a}_2$ be generated from a large enough $\mathsf{seed} \in \{0,1\}^\lambda$ ensuring $\lambda(n)$ classical bits. Here G is

instantiated as a random oracle, which in practice can be replaced by pseudorandom generators such as Shake.
The encryption function works similar to LWEGen with the main difference that H outputs the additional random vectors $\mathbf{v}_i$ used to mask the message and to generate the error polynomials via Lemma 3. The Encode and Decode routines are used to translate between bit strings and vectors/polynomials. The decryption routine invokes LWEInv to recover the error polynomials and the secret $\mathbf{s}$. Finally, all steps from the encryption function are reversed such that the message is unmasked again.
Remark 2. For $q = 2^k$, $\mathbf{t}_i = \mathsf{Encode}(\mathsf{m_i}) + \mathbf{v}_\mathsf{i} \bmod 2$ is equivalent to $\mathbf{t}_i = \mathsf{Encode}(\mathsf{m_i} \oplus \mathsf{h_i})$, where $\mathsf{h_i} := \mathsf{Decode}(\mathbf{v}_\mathsf{i})$. This complies with the representation of Section 3, when defining the ALWE problem. We can also directly generate $\mathbf{v}_i$ as a bit string during encryption without the need for conversion.
In the standard IND-CPA security game the adversary is challenged to correctly guess the bit b with non-negligible advantage given two distinct messages of his/her choice.

```
Experiment $\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathrm{ind-CPA}}(n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^k)$
$(\mu_0, \mu_1) \leftarrow \mathcal{A}(\textsc{choose}, \mathsf{pk})$
$\mathbf{c}_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_b)$ for $b \leftarrow_R \{0,1\}$
$b' \leftarrow \mathcal{A}(\textsc{guess}, \mathbf{c}_b)$
Output 1 iff
    1. $b' = b$
    2. $|\mu_0| = |\mu_1|$
```

We now state the main theorem of this section, which can be extended to the quantum random oracle case. By $\mathsf{lb-RLWE}$ we define the problem of finding low order bits in ring-LWE instances.

Theorem 2. *Let $\mathsf{lb-RLWE}$ be defined as in Lemma 7. In the random oracle model, assume that there exists a PPT-adversary $\mathcal{A}$ against the scheme with $s \geq \mathsf{w} \cdot \sqrt{\ln\left(2\mathsf{n}\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$, then there exists a reduction $\mathcal{M}$ that breaks ring-LWE/ring-ALWE such that*

$$\mathsf{Adv}_{\mathsf{LARA}}^{\mathsf{CPA}}(\mathcal{A}) \leq 3\mathsf{Adv}_{\mathsf{n},3}^{\mathsf{dec-RLWE}}(\mathcal{M}) + \mathsf{Adv}_{\mathsf{n},3}^{\mathsf{lb-RLWE}}(\mathcal{M}) + \mathsf{q_H}/\mathsf{p^n}\,.$$

Proof. We proceed via a sequence of hybrids and show that the ciphertext is pseudorandom under any of the computational assumptions, namely ring-LWE or ring-ALWE, where latter is itself based on ring-LWE. Let $\mathcal{H}_0$ be the real $\mathsf{IND-CPA}$ game. In the first hybrid $\mathcal{H}_1$, we replace $\mathbf{a}_3$ by a uniform random polynomial. If there exists a distinguisher that can distinguish $\mathcal{H}_0$ from $\mathcal{H}_1$, then there exists a reduction $\mathcal{M}_0$ that breaks decision ring-LWE ($\mathsf{dec-RLWE}$). Thus, $\mathsf{Adv}_{\mathcal{H}_0,\mathcal{H}_1}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{n},1}^{\mathsf{dec-RLWE}}(\mathcal{M}_0)$. In the second hybrid $\mathcal{H}_2$, we change the random oracle output $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3), \mathbf{d}$ of $\mathsf{H}(\mathbf{c}_0)$ by uniform random values and thus also $\mathbf{t}_i$ for $1 \leq i \leq 3$. A PPT adversary can only distinguish $\mathcal{H}_1$ from $\mathcal{H}_2$, if it queries H on $\mathbf{c}_0$ (see below). But then, a reduction $\mathcal{M}_1$ exists (Lemma 7) that breaks $\mathsf{lb-RLWE}$. Thus, we have $\mathsf{Adv}_{\mathcal{H}_1,\mathcal{H}_2}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{lb-RLWE}}_{\mathsf{n,3}}(\mathcal{M}_1) + \mathsf{q_H/p^n}$. Latter term represents the probability of a correct guess with at most q_{H} queries to H. In the third hybrid $\mathcal{H}_3$, we replace $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbf{t}_i+p\mathbb{Z}^n,s}$ by $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z}^n,s}$ (coefficient embedding) via Lemma 3. Here, $\mathbf{e}_i$ is distributed statistically close to the discrete Gaussian distribution. Thus, $\mathsf{Adv}_{\mathcal{H}_2,\mathcal{H}_3}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{dec-RLWE}}_{\mathsf{n,3}}(\mathcal{M}_2)$ for appropriate parameters. Note that $\mathsf{Adv}_{\mathcal{H}_2,\mathcal{H}_3}(\mathcal{A})$ is bounded by the statistical distance. We note that samples from $\mathcal{H}_2$ are ring-LWE instances (except with negligible statistical distance). In the last hybrid $\mathcal{H}_4$, we let the ciphertexts $\mathbf{b}_i$ for $1 \leq i \leq 3$ be generated uniformly at random rather than as ring-LWE instances. Thus, $\mathsf{Adv}_{\mathcal{H}_3,\mathcal{H}_4}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{dec-RLWE}}_{\mathsf{n,3}}(\mathcal{M}_3)$. The claim follows from

$$\begin{aligned}\mathsf{Adv}_{\mathcal{H}_0,\mathcal{H}_4}(\mathcal{A}) &\leq \mathsf{Adv}_{\mathcal{H}_0,\mathcal{H}_1}(\mathcal{A}) + \mathsf{Adv}_{\mathcal{H}_1,\mathcal{H}_2}(\mathcal{A}) + \mathsf{Adv}_{\mathcal{H}_2,\mathcal{H}_3}(\mathcal{A}) + \mathsf{Adv}_{\mathcal{H}_3,\mathcal{H}_4}(\mathcal{A}) \\ &\leq 3\mathsf{Adv}^{\mathsf{dec-RLWE}}_{\mathsf{n,3}}(\mathcal{M}) + \mathsf{Adv}^{\mathsf{lb-RLWE}}_{\mathsf{n,3}}(\mathcal{M}) + \mathsf{q_H/p^n},\end{aligned}$$

We stress that the adversary cannot tell apart samples from $\mathcal{H}_1$ and $\mathcal{H}_2$ unless he queries the RO on $\mathbf{c}_0$, i.e., before querying the RO on $\mathbf{c}_0$ samples from $\mathcal{H}_1$ are indistinguishable from ones in $\mathcal{H}_2$ (ring-LWE samples) in the adversary's view. Thus, with the same probability as in $\mathcal{H}_1$ the adversary queries the RO on $\mathbf{c}_0$ when he is only given ring-LWE samples from $\mathcal{H}_2$. In $\mathcal{H}_2$ the only information the adversary gets about $\mathbf{c}_0$ is the ring-LWE instance with least significant bits $\mathbf{c}_0$ of the secret. Thus, if it queries H on $\mathbf{c}_0$ with non-negligible probability, it breaks ring-LWE as per Lemma 7. □

Lemma 7. *Let $q = O(n)$ and ℓ the error size. Suppose there exists a* PPT *algorithm $\mathcal{S}$ that can output the low order bits of the secret in ring-LWE instances (*$\mathsf{lb-RLWE}$ *problem), then there exists a* PPT *algorithm $\mathcal{B}$ that breaks the search version of ring-LWE.*

Proof. Suppose there exists such an algorithm. For simplicity, let p be coprime to q. The ring-LWE samples $\{(\mathbf{a}_i, \mathbf{b}_i := \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i \bmod q)\}_i$ define the problem instance. $\mathcal{B}$ is challenged to find $\mathbf{s}$. With high probability, there exists an invertible element $\mathbf{a}_j \in \mathcal{R}_q$ (see e.g. [Pei15]). Taking any other sample, e.g. $(\mathbf{a}_1, \mathbf{b}_1 := \mathbf{a}_1 \cdot \mathbf{s} + \mathbf{e}_1 \bmod q)$, we can construct samples $\mathbf{b}_i^{(0)} := (\mathbf{a}_j^{-1}\mathbf{a}_i)\cdot(\mathbf{e}_j+\mathbf{d})+\mathbf{e}_i = \mathbf{a}_j^{-1}\mathbf{a}_i\cdot(\mathbf{b}_j+\mathbf{d})-\mathbf{b}_i \bmod q$ with $\mathbf{e}_j+\mathbf{d}$ as the secret and $i \neq j$ [ACPS09]. Here, the term $\mathbf{d}$ is filled with the tail bound at each coefficient such that all coefficients of $\mathbf{e}_j + \mathbf{d}$ are positive. Finding $\mathbf{s}$ is equivalent to recovering $(\mathbf{e}_j + \mathbf{d}) := \mathbf{e}_j^{(0)} = \mathbf{c}_0 + p\cdot\mathbf{c}_1 + \ldots + p^{\ell-1}\mathbf{c}_{\ell-1}$ with $\mathbf{c}_i \in \mathbb{Z}_p^n$ for some small $\ell < k$. The first input to $\mathcal{S}$ is therefore $(\mathbf{a}_i^{(0)} := \mathbf{a}_j^{-1}\mathbf{a}_i, \mathbf{b}_i^{(0)})$ which outputs $\mathbf{c}_0$ in polynomial time by assumption. In the second iteration the input is modified to $(\mathbf{a}_i^{(1)}, \mathbf{b}_i^{(1)})$ with $\mathbf{a}_i^{(1)} := p \cdot \mathbf{a}_i^{(0)}$,

$$\begin{aligned}\mathbf{b}_i^{(1)} &:= \mathbf{b}_i^{(0)} - \mathbf{a}_i^{(0)} \cdot \mathbf{c}^{(0)} \bmod q = \mathbf{a}_i^{(0)} \cdot (p \cdot \mathbf{c}_1 + \ldots + p^{\ell-1}\mathbf{c}_{\ell-1}) + \mathbf{e}_i \bmod q \\ &= (p \cdot \mathbf{a}_i^{(0)}) \cdot (\mathbf{c}_1 + \ldots + p^{\ell-2}\mathbf{c}_{\ell-1}) + \mathbf{e}_i \bmod q = \mathbf{a}_i^{(1)} \cdot \mathbf{e}_j^{(1)} + \mathbf{e}_i \bmod q\end{aligned}$$

and secret $\mathbf{e}_j^{(1)} := (\mathbf{c}_1 + \ldots + p^{\ell-2}\mathbf{c}_{\ell-1})$. Then, $\mathcal{S}$ outputs $\mathbf{c}_1$ by assumption. Analogously, via $(\mathbf{a}_i^{(t)} := p^t \cdot \mathbf{a}_i^{(0)}, \mathbf{b}_i^{(t)} := \mathbf{b}_i^{(t-1)} - \mathbf{a}_i^{(t-1)} \cdot \mathbf{c}^{(t-1)})$ as input instances to $\mathcal{S}$ the algorithm $\mathcal{B}$ obtains all $\mathbf{c}_t$ for $0 \leq t \leq \ell - 1$, recovers $\mathbf{e}_j + \mathbf{d} = \mathbf{c}_0 + p \cdot \mathbf{c}_1 + \ldots + p^{\ell-1}\mathbf{c}_{\ell-1}$ and thus $\mathbf{s}$ (after ℓ iterations) solving $\mathsf{search} - \mathsf{RLWE}$. Adding small errors to $\mathbf{a}_i^{(t)}$ generalizes the proof to all p, q.

5.1 CCA-secure Encryption

In order to obtain CCA-security, there exist 2 approaches. The first approach just requires to turn a CPA-secure public key encryption scheme via the Fujisako-Okamoto transform [FO99] into a CCA-secure hybrid encryption scheme. This can indeed be made very efficiently, where the symmetric key cipher could be instantiated by a random oracle or pseudorandom function (such as Shake). The other approach is realized based on the so-called tag approach, where a random tag [MP12, ABB10, PV08] is applied to the public key prior to encryption, i.e. we have $\mathbf{A}_u = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{t}_u \cdot \mathbf{g} - (\mathbf{a}_1\mathbf{r}_1 + \mathbf{a}_2\mathbf{r}_2)]$. This has been realized in several works such as [MP12, EDB15]. To this end, a large tag space $\mathcal{T}$ has to be defined, out of which the tag is drawn uniformly at random. An element is called a tag, if it is a unit in the ring and satisfies the unit difference property. That is, for two units $\mathbf{u}, \mathbf{v} \in \mathcal{T}$ the difference $\mathbf{u} - \mathbf{v}$ is again a unit. Beside of these properties, a further objective is to specify efficient algorithms that allow to sample elements from $\mathcal{T}$ uniformly at random. In fact, for $q = 3^k$ the tag space may be defined to consist of binary polynomials of degree smaller than $n/2$ such that it satisfies the unit difference property as per Corollary 2. Thus, it suffices to sample binary strings of length $n/2$ bits and map them to the corresponding binary polynomial of degree smaller than $n/2$. In Section 4.1 we explained how to generate $\mathbf{t}_u \cdot \mathbf{s}$ such that we can recover $\mathbf{s}$. Using the framework from [EDB15] we give a CCA-secure scheme in Appendix A.

Corollary 2 (Unit Difference Property). *Let the tag space be defined as* $\mathcal{T} = \{a_0 + a_1 \cdot x + \ldots + a_{n/2-1} \cdot x^{n/2-1} \mid$ *for* $a_i \in \{0,1\}\}\backslash\mathbf{0}$. *Then, any tag* $\mathbf{u} \in \mathcal{T}$ *satisfies the unit difference property.*

Proof. Any two elements $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{T}$ are invertible as per Corollary 1. Since both tags of degree at most $n/2 - 1$ have coefficients in $\{0, 1\}$, the difference $\mathbf{u}_1 - \mathbf{u}_2$ has coefficients in $\{0, \pm 1\}$; thus invertible as per Corollary 1. □

6 Software Implementation and Performance Analysis

At the implementation front we consider several optimizations and present an overview of the main ingredients. The error polynomials are generated as $\mathbf{e}_i \leftarrow \mathcal{D}_{p\mathbb{Z}^n+\mathbf{v}_i,\alpha q}$ for uniform random cosets $\mathbf{v}_i$ following Lemma 3. This is realized with the aid of lookup tables, where almost 0.99 of the probability mass is concentrated on the 10 mid elements. Furthermore, we can use buckets for the 10 mid elements such that one call suffices to obtain a sample in 0.99 of all cases. In general, we find the right element after around 2 table lookups. By this technique we can build an efficient discrete Gaussian sampler. We instantiate the random oracle $H(\cdot)$, when encrypting messages, by an efficient and secure pseudo-random generator such as Salsa20 or Shake[1].

[1] KeccakCodeProject: `github.com/gvanas/KeccakCodePackage/`

The secret key might consist of uniform random elements deduced from a seed and pk. We refer to the table below for a description parameters in use.

Parameter	**Description**
n	Dimension
q	Modulus
λ	Size of seed generating $\mathbf{a}_1, \mathbf{a}_2$
w	Message range
s	Error distribution $\mathcal{D}_{\mathbb{Z}^n,s}$
r_{sec}	Distribution of secret keys: $\mathcal{D}_{\mathbb{Z}^n,r_{sec}}$ or uniform with integer coefficients from $(-r_{sec}, r_{sec}]$
Message size	$3n\log_2 \mathsf{w}$
Ciphertext size	$3n\log_2(q)$
Public key size	$\lambda + n\lceil\log_2(q)\rceil$
Secret key size	$2\lambda + n\lceil\log_2(q)\rceil$

6.1 Performance Analysis and Implementation Results

We implemented both our CPA/CCA secure schemes for $n = 1024$ on a machine that is specified by an Intel Core i5-6200U processor operating at 2.3GHz and 8GB of RAM. We used a gcc-5.4 compiler with compilation flags Ofast. Table 2 compares different schemes at a security level of 256 bits,

Scheme (**Ref-Impl**)	$\mathsf{LARA}_{\mathsf{CPA}}$ This work	$\mathsf{LARA}_{\mathsf{CCA}}$ This work	Kyber1024 – KEM [BDK+]	$\mathsf{spLWE}_{\mathsf{CCA}}$ [CHK+17]
q	2^{15}	3^9	7681	520
Enc (in cycles)	414 586	497 239	481 042	813 800
Dec (in cycles)	289 463	418 147	558 740	787 800
Message size (in bits)	6 144	6 144	256	256
Ciphertext exp	7.5	7.5	>33	25
PK size (in bytes)	1 984	1 984	1 081	-
SK size (in bytes)	2 048	2 048	2 400	-

Fig. 2: Experimental results from our reference implementation.

where $\mathsf{spLWE}_{\mathsf{CCA}}$ provides only 128 bits of security. LARA has very small ciphertext expansion factors represented by a very low ratio of the ciphertext size per message bit. The number of cycles per encrypted message bit as well as its absolute performance and key sizes are very competitive for the CPA and CCA secure schemes. For instance, we are able to encrypt 2 bits per entry for $q = 2^{15}$ or $q = 2^{14}$ resulting in 414 586 cycles for encryption or 67 cycles per message bit. In order to estimate the security we used the LWE estimator[2].

[2] https://bitbucket.org/malb/lwe-estimator

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, May 2010. 1, 14

ABBK17. Nabil Alkeilani Alkadri, Johannes Buchmann, Rachid El Bansarkhani, and Juliane Krämer. A framework to select parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2017/615, 2017. `http://eprint.iacr.org/2017/615`. 7

ACPS09. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, pages 595–618, 2009. 13

Ban95. W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in r^n. *Discrete Computational Geometry*, 13(1):217–231, 1995. 5

BDK$^+$. Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals – kyber: a cca-secure module-lattice-based kem. 15

BF11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, May 2011. 7

CHK$^+$17. Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. *A Practical Post-Quantum Public-Key Cryptosystem Based on spLWE*. Springer, 2017. 15

DM14. Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014*, pages 335–352. Springer, 2014. 5

EB14. Rachid El Bansarkhani and Johannes Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *SAC 2013*, LNCS. Springer, 2014. 2, 8

EDB15. Rachid El Bansarkhani, Özgür Dagdelen, and Johannes Buchmann. Augmented learning with errors: The untapped potential of the error term. In *Financial Crypto 2015*, LNCS. Springer, 2015. `http://eprint.iacr.org/2015/042`. 1, 2, 3, 4, 5, 6, 7, 10, 14, 19

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, August 1999. 14

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008. 1, 5

LP11. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA*, LNCS, pages 319–339. Springer, 2011. 1, 3

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *A Toolkit for Ring-LWE Cryptography*, pages 35–54. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 3

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, April 2012. 1, 2, 3, 6, 8, 10, 14

Pei09. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009. 1

Pei10. Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, August 2010. 1

Pei15. Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. https://eprint.iacr.org/2015/939. 13

PV08. Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 536–553. Springer, August 2008. 14

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005. 1

SS11. Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47. Springer, 2011. 1

A CCA-secure Encryption with Tags

Let $q = 3^k$ and $\mathcal{T}$ define the tag space containing binary polynomials of degree less than $n/2$.

```
$\underline{\mathsf{LARA}_{\mathsf{CCA}}.\mathsf{KGen}\ :(1^{\ell})}$
1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{TrapGen}(1^{\mathsf{n}})$
2: We replace $\mathbf{a}_1, \mathbf{a}_2$ in the public key with the $\mathsf{seed}$ generating those elements,
   i.e. $\mathsf{pk} = (\mathsf{seed}, \mathbf{a}_3)$, where $\mathbf{a}_3 = (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2)$.

$\underline{\mathsf{LARA}_{\mathsf{CCA}}.\mathsf{Enc}(\mathsf{pk}, (\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3) \in \{0,1\}^{3\mathsf{n} \log \mathsf{w}})}$
1: $\mathbf{a}_1, \mathbf{a}_2 \leftarrow \mathsf{G}(\mathsf{seed})$
2: $\mathbf{t}_u \in \mathcal{T}$
3: $\mathbf{c}_0 = (c_{1,0}, \ldots, c_{n-1,0}) \leftarrow \mathbb{Z}_p^n$
4: $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3), \mathbf{d} \leftarrow \mathsf{H}(\mathbf{c}_0) \in \mathbb{Z}_{\mathsf{w}}^{3\mathsf{n}} \times \mathbb{Z}_{\mathsf{p}^{\mathsf{k}-1}}^{\mathsf{n}}$
5: $\mathbf{t}_u \cdot \mathbf{s} = \mathbf{c}_0 + p \cdot \mathbf{d}$
6: $\mathbf{s} := \mathbf{t}_u^{-1} \cdot (\mathbf{c}_0 + p \cdot \mathbf{d})$
7: $\mathbf{t}_i = \mathsf{Encode}(\mathsf{m}_\mathsf{i}) + \mathbf{v}_\mathsf{i} \bmod \mathsf{w}$ for $1 \leq \mathsf{i} \leq 3$
8: $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbf{t}_i + \mathsf{w} \cdot \mathbb{Z}^{\mathsf{n}}, \mathsf{s}}$ for $1 \leq i \leq 3$
9: $\mathbf{b}_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i$ for $i = 1, 2$
10: $\mathbf{b}_3 = \mathbf{t}_u \mathbf{s} \cdot \mathbf{g} - \mathbf{a}_3 \cdot \mathbf{s} + \mathbf{e}_3$
11: $\mathsf{h} = \mathsf{H}(\mathbf{s}, \mathbf{t}_\mathsf{u}, \hat{\mathbf{e}})$
12: Output $u, \mathsf{h}, \hat{\mathbf{b}} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$.

$\underline{\mathsf{LARA}_{\mathsf{CCA}}.\mathsf{Dec}(\mathsf{sk}, (\mathsf{u}, \mathsf{h}, \hat{\mathbf{b}}) \in \{0,1\}^{\mathsf{n}/2} \times \{0,1\}^{\lambda} \times \mathcal{R}_{\mathsf{q}}^3)}$
1: $\mathbf{a}_1, \mathbf{a}_2 \leftarrow \mathsf{G}(\mathsf{seed})$
2: $(\mathbf{t}_u \mathbf{s} := \mathbf{c}_0 + p \cdot \mathbf{d}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \leftarrow \mathsf{LWEInv}'(\mathsf{sk}, \hat{\mathbf{b}})$
3: $(\mathsf{v}_1, \mathsf{v}_2, \mathbf{v}_3), \mathbf{d} \leftarrow \mathsf{H}(\mathbf{c}_0) \in \{0,1\}^{3\mathsf{n} \log \mathsf{w}} \times \mathbb{Z}_{\mathsf{p}^{\mathsf{k}-1}}^{\mathsf{n}}$
4: if $\mathsf{h} = \mathsf{H}(\mathbf{s}, \mathbf{t}_\mathsf{u}, \hat{\mathbf{e}}) \wedge \| \hat{\mathbf{e}} \| \leq \sqrt{3\mathsf{n}} \cdot \mathsf{s}$
5:     $\mathbf{t}_i = \mathbf{e}_i \bmod w$ for $1 \leq i \leq 3$
6:     $\mathsf{m}_\mathsf{i} = \mathsf{Decode}(\mathbf{t}_\mathsf{i} - \mathbf{v}_\mathsf{i} \bmod \mathsf{w})$
7:     Output $(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3)$
```

Fig. 3: Description of the CCA-secure Encryption Scheme.

Remark 3. We note that in the encryption routine we have $(\mathbf{t}_u \cdot \mathbf{g} - \mathbf{a}_3) \cdot \mathbf{s} + \mathbf{e}_3 = \mathbf{t}_u \mathbf{s} \cdot \mathbf{g} - \mathbf{a}_3 \cdot \mathbf{s} + \mathbf{e}_3$. Furthermore,the trapdoor inversion algorithm LWEInv' computes the same quantities as LWEInv with the difference that it also deduces $\mathbf{t}_u$ from u via the coefficient embedding. Once $\mathbf{t}_u \cdot \mathbf{s}$ is recovered, one can compute $\mathbf{s}$ and thus $\hat{\mathbf{e}} = \hat{\mathbf{b}} - \mathbf{A}_u \cdot \mathbf{s}$ (see Section 4).

A.1 Chosen Ciphertext Security and Variants

We recall the definitions of (replayable) chosen ciphertext security of encryption schemes. Let $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme and consider the following experiments for $\mathsf{atk} \in \{\mathsf{cca1}, \mathsf{cca2}, \mathsf{rcca}\}$:

```
Experiment $\mathbf{Exp}^{\mathrm{ind-}\mathsf{atk}}_{\mathcal{E},\mathcal{A}}(n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^k)$
$(\mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(\cdot)}(\textsc{choose}, \mathsf{pk})$
$\mathbf{c}_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_b)$ for $b \leftarrow_R \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot)}(\textsc{guess}, \mathbf{c}_b)$
Output 1 iff
    1. $b' = b$
    2. $|\mu_0| = |\mu_1|$
    3. $\mathbf{c}_b$ was not queried to $\mathcal{O}_2$
```

If $\mathcal{A}$ queries $\mathcal{O}_2(\mathbf{c})$, and

- if $\mathsf{atk} = \mathsf{cca1}$, then return $\perp$.
- if $\mathsf{atk} = \mathsf{cca2}$, then return $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$.
- if $\mathsf{atk} = \mathsf{rcca}$ and $\mathsf{Dec}(\mathsf{sk}, \mathbf{c}) \notin \{\mu_0, \mu_1\}$, then return $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$.
- Otherwise, return $\perp$.

The security of the scheme directly follows from the framework as described in [EDB15].