LARA
A Design Concept for Lattice-based Encryption

Rachid El Bansarkhani

Technische Universität Darmstadt
Fachbereich Informatik
Kryptographie und Computeralgebra,
Hochschulstraße 10, 64289 Darmstadt, 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 characteristics 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. In particular, many constructions still encrypt data in an one-time-pad manner considering LWE instances as random vectors added to a message, most often encoded bit vectors. 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. Combinations of both concepts are also possible. In this paper we revisit this approach and propose amongst others a standard model variant of the scheme as well as several techniques in order to improve the message throughput per ciphertext. Furthermore, we introduce a new discrete Gaussian sampler, that is inherently induced by the encryption scheme itself, and 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 a security analysis as well as an efficient implementation of the scheme instantiated with the new trapdoor construction. In particular, we attest high message throughputs (message expansion factors close to 1-2) at running times comparable to the CPA-secure encryption scheme from Lindner and Peikert (CT-RSA 2011). 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. This feature is naturally induced by the characteristics of LWE.

Keywords: Lattice-Based Encryption, Lattice-Based Assumptions

1 Introduction

Lattice-based cryptography emerges as a promising candidate to replace classical systems in case powerful quantum computers are built. Besides of its conjectured quantum resistance, lattice problems have a long history in mathematics and gained, for instance in cryptography, a lot of attention in recent years due to a series of seminal works. For instance, Ajtai's work [2] on worst-case to average-case hardness of lattice problems represents a major cornerstone for lattice-based cryptography in general as it opens up the possibility to build provably secure schemes based on lattice-problems after some failed constructions such as GGH [27] and NTRU-Sign [29]. One of the main contributions of his work is a proof that average-case instances of lattice problems enjoy worst-case hardness. In particular for cryptography, the average-case problems SIS and LWE form the foundation for almost all of the well-known lattice-based cryptosystems. LWE is often used for primitives from Cryptomania such as provably secure encryption schemes [21, 33, 52] including chosen-ciphertext secure encryption [48, 24, 44, 1, 41], identity based encryption [24, 15, 1, 41] and fully homomorhic encryption [13, 14, 23, 25]. Furthermore, it is applied for building secure key exchange protocols [31, 30] and oblivious transfer [47]. The decision problem of LWE asks a challenger to distinguish polynomially many samples $(\mathbf{A}_i, \mathbf{b}_i^\top) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{A}_i \leftarrow_R \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_i \leftarrow_R \chi$ and $\mathbf{b}_i^\top = \mathbf{s}^\top \mathbf{A}_i + \mathbf{e}_i^\top \mod q$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and discrete Gaussian distribution χ, from uniform random samples in $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$. Regev showed in [51] that solving the search problem of LWE, where the challenger is supposed to find the secret, is at least as hard as quantumly approximating SIVP resp. GapSVP to factors $\tilde{O}(n/\alpha)$ in n-dimensional worst-case lattices for error vectors following the discrete Gaussian distribution with parameter $\alpha q \geq 2\sqrt{n}$. The SIS problem, however, usually serves to build provably secure signature schemes [8, 18, 28, 36, 24, 41], preimage sampleable trapdoor functions [24, 41, 5, 45, 52] and collision-resistant hash functions [37, 6]. The corresponding ring variants represent a further milestone for practical lattice-based cryptography as it allows for small key sizes while speeding up operations. As a consequence, new problems were formulated (e.g. [21, 35, 12]) allowing for fast instantiations of cryptographic schemes or solving open problems. The hardness of the new established problems (e.g. k-SIS, k-LWE or PLWE problem) mostly stem from either the SIS or the LWE problem.

Recently, a novel lattice-based encryption scheme [21] has been proposed that encrypts data in a way that differs from previous constructions following the one-time-pad approach. It is equipped with many nice 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 to secure the transmission of bundles of keys as required for the provisioning of remote attestation keys during manufacturing or data authentication, i.e. two keys in order to instantiate a symmetric key cipher and a MAC. Public key encryption schemes also represent important building blocks of sophisticated constructions such as group and ABS schemes. Thus, the argument of sole CPA-secure hybrid encryption schemes does not hold, especially in case it is desired to ensure CCA1- or CCA2-security. More specifically, the Augmented Learning with Errors problem (A-LWE) [21], 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 [41, 20], the owner of the trapdoor is empowered to recover the secret resp. error-term and hence reveal the injected data. By this, 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. The high data load encryption mode further allows for more flexibility as the message throughput can arbitrarily be increased by use of additional uniform random polynomials (e.g. using a seed) and thus an extended error term without generating new LWE secret vectors. This mechanism is indeed a property inherited from LWE as it is possible to generate polynomially many samples using the same secret vector. Following this approach, one benefits from the features of LWE and realize (R)CCA-secure encryption algorithms that are not built upon the one-time-pad approach as before. Beside of its presumed efficiency due to the resemblance of ciphertexts to plain LWE samples, the scheme further allows to be combined with the one-time-pad approach, hence, taking the best of both worlds.

1.1 Our Contributions

In this paper we revisit the A-LWE problem and the implied encryption schemes from [21]. 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 (**LA**ttice-based encryption of data embedded in **RA**ndomness):

- **Flexibility.** The encryptor of the scheme can arbitrarily increase the amount of encrypted data in two ways without invoking the encryption engine several times. First, he can increase the error size to at most $\|\mathbf{e}_i\|_2 < q/4$ depending on the parameters. Second, the encryptor can generate from a uniform random seed a desired number of uniform random polynomials and use them in order to encrypt/embed further data in the associated error terms while exploiting the same secret. The respective error size can even be larger as the decryptor is applying the trapdoor only once on the first 3 polynomials. Such a feature, that is inherited directly from ring-LWE, may be desired when transmitting many session keys at once, consuming one per session. In previous schemes, the encryptor has to invoke the encryption engine several times. Alternatively, the decryptor would have to extend its public key to the desired size (e.g. for [33]) resulting in a rather static scheme.
- **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 message/error term 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.
- **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 via the transformations from [21].

Standard Model. We efficiently convert the A-LWE problem and the related schemes into the standard model, hence, removing the need for random oracles (RO). By our new technique we can indeed replace ROs by PRNGs. All algorithms and schemes can immediately be transferred to this setting without affecting its efficiency and message throughput. More precisely, our technique is based on the leakage resilience of LWE. Due to its robustness LWE instances do not get significantly easier when disclosing some information of the secret or error vector. Following the extract-then-expand approach a fresh and uncorrelated seed is deduced, that is stretched via a PRNG to the desired random output length.

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 $\log_2(\alpha q/4.7)$ 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. Following this approach we can revise the parameters from [21] according to Table 1. When comparing our approach with the CPA-secure encryption scheme from Lindner and Peikert [33], we attest an improvement factor of at least $O(\log(\alpha q))$. Via our new discrete Gaussian sampler (see below) the performance does not suffer, when increasing the error size.

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. Prior to a presentation of the concrete details, we first instantiate the CCA1-secure encryption scheme in the ring setting for public keys being both statistically and computationally indistinguishable from uniform. This is obtained by use of an efficient ring variant [20] of

$m = c \cdot nk$	CCA	CCA	CCA	CCA	CPA
$k = \log q$	**[41]**	**[21]**	**This work**	**This work** + **[41]**	**[33]**
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	$\frac{k}{\log(\alpha q/4.7)}$	$\frac{k}{\log(\alpha q/4.7)+1/c}$	$k + \frac{k}{ck-1}$

Table 1. Parameters

the most recent trapdoor candidate [41] (see also Toolkit [39]) ensuring CCA1-security almost for free. Second and more importantly, we also give based on [20] an improved construction of trapdoor algorithms (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 random oracle. This is a crucial ingredient of our construction and the resulting schemes. In particular, we achieve public and secret keys 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 [41] is $n \times n(1 + \lg q)$, see also [39]), and is thus comparable with the public key size of [34]. This makes the usage of trapdoor based constructions more attractive for practice as it provides direct access to the secret and error vector.
In addition to the possibility to increase the error size in order to encrypt more data, the proposed high data load encryption mode (extension by uniform random polynomials) following [21] further allows to encrypt large blocks of data (via an increased number of error vector entries) using the same secret and at a minimal increase of the running time for encryption and decryption. We, thus, obtain a flexible encryption engine. The system under scrutiny will, moreover, be analyzed in terms of security using state-of-the-art tools from cryptanalysis such as the embedding approach from [4] and [8].

Improved LWE Inversion For Arbitrary Modulus. We introduce a new LWE inversion subroutine for arbitrary moduli and hence generalize the algorithm from Micciancio and Peikert [41] using a different approach. There exist many application scenarios, where it is preferred to have a modulus of a specific shape such as a prime modulus satisfying $q \equiv 1 \bmod 2n$ in the ring setting offering the possibility to apply the fast NTT transformation. The efficient inversion subroutine given in [41] to recover the secret vector from LWE instances is only suitable for moduli of the form $q = 2^k$. For moduli different to this shape, generic approaches such as [7, 32, 24] have to be applied, which are less efficient for practice. Our algorithm, however, can efficiently be applied to LWE instances employing arbitrary moduli.

Efficient Tag Generation and Inversion of Ring Elements For generating tags used to ensure CCA security (see [41]), we give a fast inversion algorithm for ring elements in special classes of rings $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, where ring elements correspond to polynomials with $n = 2^l$ coefficients and prime satisfying $q \equiv 1 \bmod 2n$. This setting is commonly used in lattice-based crypto as it allows to use the fast NTT transformation when multiplying polynomials. Our inversion algorithm, which relies on the CRT via the NTT, can check arbitrary ring elements with regard to invertibility by one NTT transformation. Thus, we can directly use the structure of the unit group and provide explicit generators of tags without performing costly checks. When working in the NTT representation, we are even not required to apply the NTT forward and backward transformations. This will particularly be important in order to lift the encryption scheme to CCA, where a tag is chosen uniformly at random from the ring of units and is subsequently applied to the public key when encrypting messages. On the basis of recent results we also use another efficient ring alternative with $q = 3^k$ where a proper subset of binary polynomials serves as the tag space. Inverting tags can also efficiently be accomplished in this setting.

Novel and Efficient Discrete Gaussian Sampler (FastCDT). For the proposed encryption schemes, we need to embed data into the error term of (A)LWE such that it still follows the discrete Gaussian distribution. However, this method gives also rise to a new disrete Gaussian sampler that is very efficient in practice. Current state-of-the-art discrete Gaussian samplers such as [24, 44, 19, 36] get less efficient once the error size is increased. This is very crucial since lattice-based schemes get more secure, if the error-size is increased, e.g. $\alpha q > 2\sqrt{n}$ for worst-case to average-case hardness. In fact, a new approach towards building discrete Gaussian samplers is desired such that sampling of discrete Gaussians with large parameters is essentially as efficient as with small ones. We therefore designed a new and powerful discrete Gaussian sampler, called $\mathsf{FastCDT}$, that is more efficient than previous samplers. The sampling procedure is almost independent from the parameter and uses at runtime a CDT table of constant size containing at most 44 entries with overwhelming probability for all $\beta q = p \cdot \omega(\sqrt{\log n})$ and integers $p > 0$. We will show that almost the whole probability mass is concentrated on the $10 - 11$ mid elements of the table. As a result, we need to consider only $5 - 6$ entries most of the time. At the same time $\mathsf{FastCDT}$ is capable of sampling from any desired partition $\Lambda_i^{\perp} = i + p\mathbb{Z}$ without any modifications and even faster than from $\mathbb{Z}$. In fact, this sampler is used to efficiently embed data into the error term of ALWE instances.

Implementation and Analysis. In order to attest the conjectured efficiency of our scheme that we call $\mathsf{LARA}_{\mathsf{CPA}}$, $\mathsf{LARA}_{\mathsf{CCA1}}$ or $\mathsf{LARA}_{\mathsf{CCA2}}$, we implement the random oracle variantes of our CPA- and CCA-secure schemes in software for $n = 512$. This implementation is optimized with respect to the underlying architecture using the AVX/AVX2 instruction set. To this end, we applied adapted variants of the techniques introduced in [26] to our setting. In particular, we adopt several optimizations for the polynomial representation and polynomial multiplication by use of efficient NTT/FFT operations. We implement our scheme and compare it with the efficient encryption scheme due to Lindner and Peikert [34]. First, we notice that applying the $\mathsf{FastCDT}$ sampler to all considered schemes already leads to a significant performance boost. We observe improvement factors ranging between 1.3 and 2.6 as compared to the usage of current state-of-the-art samplers such as the standard CDT sampler for the same set of parameters. Moreover, we attest running times of $15.1 - 50.1$ cycles per message bit for encryption and about $9.1 - 42$ cycles per bit for decryption in the CPA- and CCA-secure setting. This represents an improvement factor up to 26.3 and 7.8 for encryption and decryption when comparing with [34] and even faster in comparison to the OpenSSL implementation of RSA. The absolute running times are comparable to those of [34].

1.2 Organization

This paper is structured as follows.

Section 2 Provides the relevant background of our work.
Section 3 Introduces the A-LWE problem from [21] and presents our improvements to enhance the message throughput.
Section 4 The A-LWE problem and the related algorithms are converted into the standard model.
Section 5 A description of new trapdoor algorithms and the resulting encryption schemes is proposed.

Section 6 Contains a security analysis of the scheme.
Section 7 Introduces a novel and efficient discrete Gaussian sampler, that we call FastCDT.
Section 8 Presents 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 $\mathcal{D}_{\Lambda+\mathbf{c},s}(\mathbf{x}) = \rho_s(\mathbf{x})/\rho_s(\Lambda+c)$.

Let $\mathcal{X} = \{\mathcal{X}_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_n\}_{n\in\mathbb{N}}$ be two distribution ensembles. We say $\mathcal{X}$ and $\mathcal{Y}$ are (computationally) indistinguishable, if for every polynomial time distinguisher $\mathcal{A}$ we have $|\Pr[\mathcal{A}(\mathcal{X}) = 1] - \Pr[\mathcal{A}(\mathcal{Y}) = 1]| = \mathsf{negl}(n)$, and we write $\mathcal{X} \approx_c \mathcal{Y}$ (resp. $\mathcal{X} \approx_s \mathcal{Y}$ if we allow $\mathcal{A}$ to be unbounded).

Lattices. A k-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$ containing all integer linear combinations of k linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ with $k \leq m$ and $m \geq 0$. More formally, we have $\Lambda = \{ \ \mathbf{B}\cdot\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^k \ \}$. 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

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

By $\lambda_i(\Lambda)$ we denote the *i-th successive minimum*, which is the smallest radius r such there exist i linearly independent vectors of norm at most r (typically l_2 norm) in Λ. For instance, $\lambda_1(\Lambda) = \min_{\mathbf{x}\neq\mathbf{0}} \|\mathbf{x}\|_2$ denotes the minimum distance of a lattice determined by the length of its shortest nonzero vector.

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 $\rho_{1/s}(\Lambda^*\backslash\{0\}) \leq \epsilon$.*

Lemma 1. ***([24, Theorem 3.1]).*** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. We have*

$$\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{B}} \| \cdot\sqrt{\ln(2n(1+1/\epsilon))/\pi},$$

where $\tilde{\mathbf{B}}$ denotes the orthogonalized basis.

Specifically, we have $\eta_\epsilon(\Lambda) \leq b\cdot\sqrt{\ln(2n(1+1/\epsilon))/\pi}$ for basis $\mathbf{B} = b\cdot\mathbf{I}$ of Λ.

Lemma 2 (Statistical,[21]). *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 $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ is negligible.*

Lemma 3 (Computational,[21]). *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 $r \geq \eta_\epsilon(\Lambda^{\perp}(\mathbf{B}))$.*

Lemma 4. *([42, Lemma 4.4]). Let Λ be any n-dimensional lattice. Then for any $\epsilon \in (0,1)$, $s \geq \eta_\epsilon(\Lambda)$, and $\mathbf{c} \in \mathbb{R}^n$, we have*

$$\rho_{s,\mathbf{c}}(\Lambda) \in [\frac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_s(\Lambda)\,.$$

Lemma 5 ([10, Lemma 2.4]). *For any real $s > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have*

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

Lemma 6 ([24, Theorem 3.1]). *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{S}$, and let $\epsilon > 0$. We have $\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{S}} \| \cdot \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$. In particular, for any function $\omega(\sqrt{\log n})$, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{S}} \| \cdot\omega(\sqrt{\log n})$.*

Below we give a description of the LWE distribution and the related problems of the matrix variant.

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 [21] 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 [21], 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. In Section 4 we convert the A-LWE distribution and the related problems into the standard model and thus get rid of the need for ROs. To this end, we make use of the leakage resilience of LWE following [3, Theorem 3] and destroy correlations even without ROs. All schemes remain as efficient as with ROs.

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 2 and 3 are used, which essentially state that a discrete Gaussian over the integers can be simulated by sampling 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 deterministically 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.

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 sample very short vectors efficiently 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 [41]). 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:*

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

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 [21] go through without any modifications, since the security proofs are not based on the choice of $\mathbf{B}$.

Theorem 1 (adapted [21]). *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* $\mathsf{search\ LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}s\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*
2. *If* $\mathsf{decision\ LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{decision\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*
3. *If* $\mathsf{decision\ LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}m\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ *is hard.*

One easily notes, that these hardness results also hold for the ring variant. 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 6.

In the following section 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 [11]. 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 [11] 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 $\Lambda_1 \cap \Lambda_2 + \mathbf{t}$. For instance, one can efficiently instantiate the scheme when choosing $\Lambda_1 = p\mathbb{Z}^m$ and $\Lambda_2 = \Lambda_q^{\perp}(\mathbf{A})$ for a matrix $\mathbf{A} \in \mathbb{Z}_q^{l \times m}$ with a short basis $\mathbf{T}$ and p coprime to q. Doing this, the message spaces are given by $\mathbf{m}_1 \in \mathbb{Z}^m / \Lambda_1 \cong \mathbb{Z}_p^m$ and $\mathbf{m}_2 \in \mathbb{Z}^m / \Lambda_2 \cong \mathbb{Z}_q^l$, where the isomorphisms are given by $\mathbf{x} \mapsto (\mathbf{x} \bmod p)$ and $\mathbf{x} \mapsto (\mathbf{A} \cdot \mathbf{x} \bmod q)$. Due to the simple choice of Λ_1, we obtain a short basis $\mathbf{S} = p \cdot \mathbf{T}$ for $\Lambda_1 \cap \Lambda_2 = p \cdot \Lambda_2$, where $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot \eta_\epsilon(\Lambda_2)$. So, if $\mathbf{A}$ corresponds to $\mathbf{G} \in \mathbb{Z}_q^{m/k \times m}$ for $k = \log q$, we have $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n})$. In our schemes, however, we have to sample a short vector $\mathbf{e}$ from $(H(\mathbf{r}) \oplus \mathbf{t}) + \Lambda_1 \cap \Lambda_2$ with parameter $\alpha q \geq \eta_\epsilon(\Lambda_1 \cap \Lambda_2)$, where $\mathbf{t}$ is computed as above and the (simplified) description $H : \{0,1\}^* \to \mathbb{Z}_q^m$ defines a random function taking a random string $\mathbf{r} \in \{0,1\}^*$ with sufficient entropy as input. The error vector is then given by $\mathbf{e} \leftarrow D_{\mathbf{b}+\Lambda_1 \cap \Lambda_2, \alpha q}$ with $\mathbf{b} = H(\mathbf{r}) \oplus \mathbf{t}$. Due to $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n})$ (e.g., $\alpha q = p \cdot 2 \cdot \omega(\sqrt{\log n})$), the error vector is indistinguishable from $D_{\mathbb{Z}^m, \alpha q}$ following Lemma 2 and Lemma 3. This technique allows to embed $m \log p + m$ bits of messages into the error term.

Lattices of the Form $\mathbf{p}\mathbb{Z}^{\mathbf{m}}$. 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 $m \log 2p = m \cdot (\log p + 1)$. In Lemma 7 we prove that a message throughput of size $\log p$ bits per entry is optimal for a given parameter $\alpha q = p \cdot \omega(\sqrt{\log n})$. 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 the more efficient $\mathsf{FastCDT}$ sampler that we present in Section 7. 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.

Lemma 7 (Optimal Message Bound). *Let $r = p \cdot \omega(\sqrt{\log m})$ for integers $p, m > 0$. Furthermore, let a discrete Gaussian over $\mathbb{Z}^m$ be sampled by selecting $\mathbf{v} \in \mathbb{Z}_{p'}^{n'}$ uniformly at random and then sampling $\mathcal{D}_{\mathbf{v}+\Lambda_{p'}^{\perp}(\mathbf{B}),r}$ for a preimage sampleable matrix $\mathbf{B} \in \mathbb{Z}_{p'}^{n' \times m}$ with $p', n' > 0$. An optimal bound for the maximum number of bits, that can be injected into a discrete Gaussian (or equivalently the size of $\mathbf{v}$), is given by $m \cdot \log p$ bits.*

Proof. The proof is essentially based on the bound given in Lemma 6. First, in order to sample a discrete Gaussian vector over $\mathbb{Z}^m$, the condition $r \geq \eta_\epsilon(\Lambda_{p'}^{\perp}(\mathbf{B}))$ has to be satisfied for a negligible ϵ following Lemma 2 and Lemma 3. Based on Lemma 6 we have $p \geq \| \tilde{\mathbf{S}} \|$, where $\mathbf{S}$ denotes a basis of $\mathbf{B}$ with $\mathbf{B} \cdot \mathbf{S} \equiv \mathbf{0} \bmod q$ and $\tilde{\mathbf{S}}$ its orthogonolization. We note, that it suffices to consider $m = 1$, since each component of a discrete Gaussian vector over $\mathbb{Z}^m$ is sampled independently containing the same amount of information and randomness. Following this, $n' = 1$ and subsequently $\tilde{\mathbf{S}} = \mathbf{S} \leq p$. Hence, for $\mathbf{S} = p = p'$ we obtain the maximum bit size amounting to $\log p$ bits for v such that $\mathcal{D}_{v+\Lambda_p^{\perp}(\mathbf{B}),r} = \mathcal{D}_{v+p\mathbb{Z},r}$ is identically distributed to $\mathcal{D}_{\mathbb{Z},r}$. For $m > 1$ one takes, for instance, $\tilde{\mathbf{S}} = \mathbf{S} = p \cdot \mathbf{I}_m$ resulting in $m \cdot \log p$ bits for $\mathbf{v} \in \mathbb{Z}_p^m$. $\square$

In fact, based on the bound given in Lemma 6, for any $\alpha q > 0$ and $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ the maximum number of bits that can be embedded into a component of the error term is bounded by $\log(\alpha q/\omega(\sqrt{\log n}))$. This means that $p' = \lfloor \alpha q/\omega(\sqrt{\log n}) \rfloor$ is the largest integer such that $\mathbf{e}_2 \bmod p'$ is guaranteed with overwhelming probability to be uniform random (see Lemma 7). Hence, we can choose $\mathbf{B} = \mathbf{I} \in \mathbb{Z}^{m \times m}$ with $\alpha q = p \cdot \mathsf{const}$ for $p = 2^k$ allowing for k-bits of information. The data is recovered via the efficient operation $\mathbf{v} = \mathbf{e}_2 \bmod p$. For the sake of these arguments, we will use $\mathbf{B} = \mathbf{I}$ throughout this work.

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 A-LWE in the Standard Model

In this section, we introduce an A-LWE variant that is hard under standard assumptions. Previously, the A-LWE problem was defined by use of a random oracle, that helped to destroy correlations between the secret and the output of $H(\cdot)$ and thus ensured the required distributions. Therefore, a straightforward approach to instantiate $H(\cdot)$ by a PRNG or a pseudorandom function (PRF) is not obvious. In particular, by means of a random oracle the secret $\mathbf{s}$ and the output of $H(\cdot)$ are both uniformly random and hence allow the distributions of the error term and the secret behave following the basic LWE distributions such that it is not possible to distinguish between A-LWE and original LWE samples. In this section we show how to get rid of the random oracle leading to a standard model instantiation of the A-LWE problem. We formulate the A-LWE problem, where H is replaced by a PRNG, hence avoiding the need for a cryptographic hash function modeled as random oracle. Following this, we can prove security in the standard model for many of the A-LWE based encryption schemes which are previously shown to be secure in the random oracle model.

4.1 Tools

Prior to defining the A-LWE distribution and proving the related hardness statements, an efficient mechanism is needed that allows to replace the random oracle by tools based on standard assumptions but in such a way that the message can efficiently be recovered from the error term. Our approach is based on the idea of computational extractors following the extract-then-expand design [16], where a statistical extractor is used to produce a seed as input to a PRNG expanding the seed to the desired output length. We will prove that we can deterministically produce a seed using the ingredients of LWE samples. In fact, we use a small part of the error term in order to derive a seed statistically close to uniform given LWE samples. That is, we sample a random matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times n}$ for $t < n$ and prove that $\mathbf{C} \cdot \mathbf{e}_1 \equiv \mathbf{b} \bmod q$ is indistinguishable from a random vector in $\mathbb{Z}_q^t$ given $\mathbf{A}^\top \mathbf{s} + \mathbf{e} \bmod q$ for $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)^\top \leftarrow \mathcal{D}_{\mathbb{Z}_q^m, \alpha q}$ and random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

Lemma 8. *([3, Theorem 3]) For any integer $n > 0$, integer $q \geq 2$, an error-distribution $\chi^n = \mathcal{D}_{\mathbb{Z}^n, \alpha q}$ and any subset $S \subseteq \{0, \ldots, n\}$, the two distributions $(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, (s_i)_{i \in S})$ and $(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, \mathcal{U}(\mathbb{Z}_q^{|S|}))$ are computationally indistinguishable assuming the hardness of* decision $\mathsf{LWE}_{n-|S|, m, \alpha q}$.

As a result, it follows

$$\begin{aligned}(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, (s_i)_{i \in S}) &\approx_c (\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, \mathcal{U}(\mathbb{Z}_q^{|S|})) \\ &\approx_c (\mathbf{A}, \mathcal{U}(\mathbb{Z}_q^m), \mathcal{U}(\mathbb{Z}_q^{|S|})),\end{aligned}$$

which proves the independence of $(s_i)_{i \in S}$ from the remaining parts of $\mathbf{s} = (s_i)_{i \in [n]}$ for $S \subseteq [n]$. As an immediate consequence, we obtain similar results, if $\mathbf{s}$ is sampled from the error distribution χ^n.

Corollary 1. *For any integer $n > 0$, integer $q \geq 2$, an error-distribution $\chi^n = \mathcal{D}_{\mathbb{Z}^n, \alpha q}$ and any subset $S \subseteq \{0, \ldots, n\}$, the two distributions $(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, (s_i)_{i \in S})$ and $(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, \mathcal{D}_{\mathbb{Z}_q^{|S|}, \alpha q})$ are computationally indistinguishable assuming the hardness of* decision $\mathsf{LWE}_{n-|S|, m, \alpha q}$.

This is proven in a straightforward manner following the same proof steps as in Lemma 8. However, in this case we have

$$\begin{aligned}(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, (s_i)_{i \in S}) &\approx_c (\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}, \mathcal{D}_{\mathbb{Z}_q^{|S|}, \alpha q}) \\ &\approx_c (\mathbf{A}, \mathcal{U}(\mathbb{Z}_q^m), \mathcal{D}_{\mathbb{Z}_q^{|S|}, \alpha q}),\end{aligned}$$

which is based on the hardness of decision $\mathsf{LWE}_{n-|S|, m, \alpha q}$. In order to account for leakage of coefficients we give an alternative definition of the LWE distribution allowing for leakage of t coefficients either of the error term or secret vector. Such a notation is indeed required to sample n-dimensional LWE instances based on the hardness of decision $\mathsf{LWE}_{n-t, m, \alpha q}$. In particular, when proving the hardness of the A-LWE problem it simplifies the representation of such instances.

Definition 5 (LWE Distribution with Leakage). *Let n, m, q be integers and χ_e be the error distribution over $\mathbb{Z}$. By $L_{n,m,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ 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 $(\mathbf{A}, \mathbf{b}^\top, (s_i)_{i \in S}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{|S|}$, where $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ and $t = |S| > 0$ coefficients of $\mathbf{s} \in \mathbb{Z}_q^n$ are leaked.*

We use the convention that the first t coefficients are leaked if it is not explicitly specified. From Lemma 8 (or Corollary 1) it follows that it is hard to distinguish uniform random vectors from samples following the $L_{n,m,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ distribution assuming the hardness of $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$. The next theorem provides a description of how to deterministically deduce a vector statistically close to uniform from LWE samples. This vector will serve as a seed in order to instantiate the A-LWE distribution below.

Theorem 2. *Let m, n be integers and $\mathbf{s} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$. Furthermore, let $\mathbf{A}_2 \leftarrow_R \mathbb{Z}_q^{m\times n}$, $\mathbf{A}_1 \leftarrow_R \mathbb{Z}_q^{t\times n}$ and $\mathbf{A}_1' \leftarrow_R \mathbb{Z}_q^{t\times m}$ be uniform random matrices with $t \leq (d-2\lambda)/\log q$, where $d = \mathbb{H}_\infty(\mathbf{s})$ (resp. $d = \mathbb{H}_\infty(\mathbf{e})$). Suppose that the $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ assumption holds, then*

1. $(\mathbf{A}_1\mathbf{s}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$
2. $(\mathbf{A}_1'\mathbf{e}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$

for $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$. Moreover, $\mathbf{A}_1'\mathbf{e} \bmod q$ (resp. $\mathbf{A}_1\mathbf{s} \bmod q$) is a statistical extractor.

Proof. We prove the statement $(\mathbf{A}_1\mathbf{s}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$ by contradiction. Suppose there exists a PPT distinguisher D that distinguishes between the aforementioned distributions, then we construct a PPT algorithm $\mathcal{A}$ that breaks $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$.
We note here that it suffices to prove the statements for any m as long as it is polynomially bounded. In fact, if the underlying problem is hard for $m = n$ samples, the same hardness statement must also hold for less samples. And for $m > n$, one can easily reduce the problem to $m = n$.
The input to $\mathcal{A}$ is an instance $(\mathbf{A}, \mathbf{A}\cdot\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix}+\mathbf{s})$ of $L_{n,n,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ due to leakage of the first t zero entries, where $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1\\\mathbf{A}_2\end{bmatrix} \in \mathbb{Z}_q^{n\times n}$ composed by the matrices $\mathbf{A}_1 \in \mathbb{Z}_q^{t\times n}$ and $\mathbf{A}_2 \in \mathbb{Z}_q^{n-t\times n}$ is a uniform random matrix that is invertible over $\mathbb{Z}_q^{n\times n}$ with non-negligible probability. For instance, the probability to sample a full rank matrix from $\mathbb{Z}_q^{l\times n}$ is equal to $\prod_{i=0}^{l-1}\frac{q^n-q^i}{q^{ln}} \geq \frac{(q^n-q^{l-1})^l}{q^{ln}} = (\frac{q^{n-l+1}-1}{q^{(n-l+1)}})^l$ for prime modulus and $l \leq n$. If $l = n$, the matrix is invertible with probability at least $(\frac{q-1}{q})^n \geq 1/e$ for $q = O(n)$. Now, $\mathcal{A}$ computes

$$\mathbf{A}^{-1}\cdot(\mathbf{A}\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix}+\mathbf{s}) = \mathbf{A}^{-1}\cdot\mathbf{s}+\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix},$$

which can be represented as $(\mathbf{Bs}, \mathbf{Cs}+\mathbf{e})$ for $\mathbf{A}^{-1} = \begin{bmatrix}\mathbf{B}\\\mathbf{C}\end{bmatrix}$. Subsequently, D is invoked which distinguishes the input $(\mathbf{Bs}, \mathbf{Cs}+\mathbf{e})$ and hence $\mathbf{A}\cdot\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix}+\mathbf{s}$ from uniform, thus solving $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ as per Lemma 8 and Corollary 1. For the second statement, one observes that

$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$
$$\Longleftrightarrow$$
$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_2^{-1}\mathbf{e}+\mathbf{s}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$

for $m = n$ and invertible matrix $\mathbf{A}_2$, which exists with non-negligible probability as shown before. This particularly also proves the statement for $m \leq n$.
As for $m > n$, suppose the upper n rows of $\mathbf{A}_2$ are linearly independent, otherwise we can find n out of $m > n$ rows with high probability and bundle them together in the upper part. The matrix $\mathbf{A}_2$ can subsequently be extended to an invertible matrix $\mathbf{A}_e = \begin{bmatrix}\mathbf{B}\ \mathbf{A}_2\end{bmatrix} \in \mathbb{Z}_q^{m\times m}$ with

$\mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{m \times m-n}$ by appending ones on the diagonal such that $\mathbf{A}_e \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} + \mathbf{e} = \mathbf{A}_2\mathbf{s} + \mathbf{e} \bmod q$.

$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_2\mathbf{s} + \mathbf{e}) = (\mathbf{A}_1\mathbf{e}, \mathbf{A}_e \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} + \mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$

$$\Longleftrightarrow$$

$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_e^{-1}\mathbf{e} + \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$

Hence, our claim follows from the first case. We note that $\mathbf{A}_1\mathbf{e} \bmod q$ is a statistical extractor by the Leftover Hash Lemma, if $t \leq (d - 2\lambda)/\log q$ for $d = \mathbb{H}_\infty(\mathbf{e})$. □

Remark. The theorem above mainly states, that it is even possible to reveal the first t entries of the error-term from LWE samples. This is equivalent to sampling a random matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{t \times n}$ and outputting the vector $\mathbf{A}_1\mathbf{s} \bmod q$, which is statistically close to the uniform distribution for $t \leq (d - 2\lambda)/\log q$ according to the Leftover Hash Lemma. Applying the same argument, the complete vector $\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \mathbf{s} + \begin{bmatrix} \mathbf{0} \\ \mathbf{e} \end{bmatrix}$ is statistically close to uniform if $m + t < n$.

4.2 A-LWE Distribution

The tools introduced in the previous section will allow us to deterministically derive a uniform random seed by means of the error term from LWE samples. Using this framework we start defining the A-LWE distribution $L^{\text{A-LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ in the standard model. It looks very similar to the random oracle variant introduced in the previous section.

Definition 6 (Augmented LWE Distribution). *Let $n, n', t, m, m_1, m_2, q, p$ be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ for a preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ and a real $\epsilon = \mathsf{negl}(\lambda) > 0$. Let $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ be a random matrix and $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{n' \cdot \log p}$ be a secure pseudorandom generator. For $\mathbf{s} \in \mathbb{Z}_q^n$, define the A-LWE distribution $L^{\text{A-LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ with $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$ to be the distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ obtained as follows:*
- *Sample $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.*
- *Set $\mathsf{seed} = \mathbf{C} \cdot \mathbf{e}_1 \bmod q$.*
- *Set $\mathbf{v} = \mathsf{encode}(\mathsf{PRNG}(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$.*
- *Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_\mathbf{v}^\perp(\mathbf{B}),\alpha q}$.*
- *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 we introduced many parameters in order to keep the problem as general as possible. But for applications considered in the previous sections, it suffices to set $n' = m_2$, $p = \lfloor \alpha q/\sqrt{\ln(2(1 + 1/\epsilon))/\pi} \rfloor$ and $\Lambda^\perp(\mathbf{B}) = p\mathbb{Z}^{m_2}$ for $\mathbf{B} = \mathbf{I} \in \mathbb{Z}_p^{m_2}$ as discussed in Section 3.2. For our constructions to work, it is possible to select small values for t such as $t = 15$ in order to ensure a seed of size $15 \cdot \log q$ bits. In fact, the seed is statistically close to uniform for $t \leq (d - 2\lambda)/\log q$ and $d = \mathbb{H}_\infty(\mathbf{e}_1)$. In this case $\mathsf{PRNG}(\mathsf{seed})$ represents a computational extractor providing enough pseudorandom bits in order to conceal the message.

4.3 A-LWE Hardness in the Standard Model

The following theorem shows that LWE samples are computationally indistinguishable from A-LWE samples, when instantiated with a PRNG. As a result, this construction also depends on the underlying computational problem of the PRNG.

Theorem 3. *Denote by λ the security parameter and let n, t, m, m_1, m_2, q, p be integers with $m = m_1 + m_2$, where $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi} \geq \eta_\epsilon(\Lambda^\perp(p\mathbb{Z}^{m_2}))$ and a real $\epsilon = \mathsf{negl}(\lambda) > 0$. Let $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ be a random matrix and $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{m_2 \cdot \log p}$ be a secure pseudorandom number generator with $t \leq (d - 2\lambda)/\log q$ for $d = \mathbb{H}_\infty(\mathbf{e}_1)$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$. Then, assuming the hardness of $\mathsf{decision}\ \mathsf{LWE}_{n-t,m,\alpha q}$ the distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ is computationally indistinguishable from uniform for arbitrary $\mathbf{m} \in \{0,1\}^{m_2 \cdot \log p}$.*

Proof. Via a series of hybrids we will prove that samples from $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ are indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times m}$ assuming that the $\mathsf{decision}\ \mathsf{LWE}_{n-t,m,\alpha,q}$ problem is hard to solve in polynomial time and the output distribution of the PRNG is indistinguishable from uniform for PPT adversaries, where $t \leq (d - 2\lambda)/\log q$ with $d = \mathbb{H}_\infty(\mathbf{e}_1)$. This yields

$$L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m}) \approx_c L^{\mathsf{LWE}_{\ell(t)}}_{n,m,\alpha q}$$

for arbitrary $\mathbf{m} \in \{0,1\}^{m_2 \cdot \log p}$. In the first hybrid, we modify the A-LWE samples in such a way that we replace $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$ with a uniformly sampled value $\mathbf{u}_1$. This follows from the fact that $(\mathbf{C}\mathbf{e}_1, \mathbf{A}_1^\top \mathbf{s} + \mathbf{e}_1)$ is indistinguishable from uniform according to Theorem 2 and $\mathbf{C}\mathbf{e}_1$ is statistically close to uniform following the Leftover Hash Lemma for t chosen as above. In the next hybrid we replace the output of the PRNG with a uniform random value $\mathbf{u}_2$ as a result of the uniform random input $\mathbf{u}_1$ to the PRNG. Following this, the vector $\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$ becomes uniformly distributed. The final hybrid replaces $\mathbf{e}_2$ by a vector $\mathbf{e}_2^*$, which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$ as per Lemma 3. As a result, we obtain instances being identically distributed as the original LWE distribution.

Hybrid$_1$. In the first hybrid, in each A-LWE sample we replace the value $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$ by a uniformly sampled value $\mathbf{u}_1 \in \mathbb{Z}_q^t$. This mainly follows from the Leftover Hash Lemma for $t \leq (d - 2\lambda)/\log q$ with $d = H_\infty(\mathbf{e}_1)$ and the fact that $(\mathbf{C}\mathbf{e}_1, \mathbf{A}_1^\top \mathbf{s} + \mathbf{e}_1)$ is indistinguishable from uniform assuming the hardness of $\mathsf{decision}\ \mathsf{LWE}_{n-t,m,\alpha q}$ as per Theorem 2. We note here, that the adversary never gets to see seed. Due to the high entropy of the seed a distinguisher will guess the correct seed only with negligible probability or at most $2^{-O(\lambda)}$. Also by the same argument, the same seed will not be sampled except with negligible probability due to the high min-entropy of $\mathbf{e}_1$.

Hybrid$_2$. In the second hybrid, we replace the output $\mathsf{PRNG}(\mathbf{u}_1)$ of the PRNG by a uniform random string $\mathbf{u}_2 \in \{0,1\}^{m_2 \cdot \log p}$. As a result,$\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$ becomes uniformly distributed as well. A potential (polynomial-time) adversary notices the difference between the uniform distribution and the output of the PRNG only if he queries the PRNG with the correct seed $\mathbf{u}_1$ or breaks the underlying hard computational problem. But in both cases, the distinguisher succeeds only with negligible probability. Hence, this also holds, if many samples are given to the distinguisher.
We comment on a distinguisher which queries the PRNG at a certain point on $(\mathbf{s}, \mathbf{e}_1)$ below in the proof, and assume for now, that no such distinguisher exists.

Hybrid$_3$. In the last hybrid, the error term $\mathbf{e}_2$ is replaced by $\mathbf{e}_2^*$ which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$. This follows from Lemma 3 stating that $\mathcal{D}_{\Lambda^\perp_{\mathbf{v}}(\mathbf{B}),\alpha q}$ is statistically close to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$ for a uniform random vector $\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$, where $\mathbf{u}_2$ is uniform random and $\alpha q \geq \eta_\epsilon(\Lambda^\perp(p\mathbb{Z}^{m_2}))$.

We stress that A-LWE samples from **Hybrid**$_3$ are indistinguishable from LWE samples. The only difference between A-LWE and LWE samples is the way the error term was constructed, which we proved via the hybrids to be identically distributed. Hence, a distinguisher can only invoke the PRNG with the correct seed either by guessing $\mathbf{C}\mathbf{e}_1 \bmod q$, which is $2^{-O(\lambda)}$-hard by the Leftover Hash Lemma, or breaking A-LWE samples which is via the relation $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m}) \approx_c L^{\mathsf{LWE}_{\ell(t)}}_{n,m,\alpha q}$

equivalent to breaking LWE samples or solving search $\mathsf{LWE}_{n-t,m,\alpha,q}$ and decision $\mathsf{LWE}_{n-t,m,\alpha q}$, respectively. By assumption such a distinguisher does not exist.

We conclude that the step from the original A-LWE samples to $\mathbf{Hybrid}_1$ will be unnoticeable to a distinguisher, if search $\mathsf{LWE}_{n-t,m,\alpha,q}$ and decision $\mathsf{LWE}_{n-t,m,\alpha q}$ are hard, and both distributions $L^{\mathsf{LWE}}_{n-t,m,\alpha,q}$ and $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ are computationally indistinguishable. □

Theorem 4. *Let n, m, m_1, m_2, q, p be integers with $m = m_1 + m_2$. Let $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{m_2 \cdot \log p}$ be a PRNG taking* $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$ *as input for a random matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ with $t \leq (d - 2\lambda)/\log q$ and $d = \mathbb{H}_\infty(\mathbf{e}_1)$. Let $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq \eta_\epsilon(\Lambda_q^\perp(p\mathbb{Z}^{m_2}))$ for a real $\epsilon = \mathsf{negl}(n) > 0$. Furthermore, denote by χ_{e_1} the distribution of the error vectors $(\mathbf{e}_1)_i$ involved in each A-LWE sample i. If $\mathbb{H}_\infty(\mathbf{e}_1) > \lambda$, then the following statements hold.*

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

The proof for the second statement follows from 3. As for the remaining statements we refer to the proof of Theorem 1, which proves the validity of the first and last statements using the same argumentation line. We note here, that the seed is always kept hidden and hence believe that the hardness of A-LWE is even stronger based on search $\mathsf{LWE}_{n,m,\alpha,q}$ and decision $\mathsf{LWE}_{n,m,\alpha q}$.

5 Encryption Scheme based on Trapdoors

5.1 Setting

Prior to starting with the key ingredients of the encryption scheme, we define the setting in which we operate and explain how it contributes to the performance and usability of the scheme. We give customized algorithms that make use of the features accompanying the scheme in consideration. In particular, we operate in the ring setting, where lattices correspond to ideals in the corresponding rings. This allows for more efficient algorithms as compared to the unstructured counterparts in $\mathbb{Z}_q^n$. More specifically, we will focus on cyclotomic rings $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ for integers $q > 0$ and n being a power of two, where $\Phi_{2n}(X) = X^n + 1$ is a cyclotomic polynomial that is irreducible over $\mathbb{Z}[X]$. Cyclotomic rings have very nice structures that allow for efficient and specialized algorithms [49] and furthermore provide similar worst-case to average-case hardness results [38]. In some constructions one may wishes to operate with a power of two modulus $q = 2^l$, when considering the trapdoor construction and the corresponding LWE inversion algorithm from [41, 20]. As already stated in many works, it might be more advantageous in certain settings to select a prime q such that $q = 1 \bmod 2n$. In this case $\Phi_{2n}(X) = X^n + 1$ splits into n linear factors over $\mathbb{Z}_q[X]$ such that $\mathcal{R}_q \cong \mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$, where $g_i(X)$ denote linear polynomials in $\mathbb{Z}_q[X]$. Due to this fact, there exists an element $\omega \in \mathbb{Z}_q$ of order $2n$ that satisfies $\Phi_{2n}(\omega) = 0 \bmod q$ since $2n | q - 1$ and $\mathbb{Z}_q^\times = \mathbb{Z}_q \backslash \{0\}$ is a cyclic group. Therefore, we can write $g_i(X) = X - \xi_i$ for some element $\xi_i \in \mathbb{Z}_q$ and use the NTT [26] in order to efficiently perform polynomial multiplication. Thus, let ξ be an element of order n and $\psi^2 = \xi \bmod q$. Then two polynomials $\mathbf{r}, \mathbf{u} \in \mathcal{R}_q$ are multiplied by first transforming $\mathbf{r} = (r_0, \ldots, r_{n-1})$ resp. $\mathbf{u}$ to $T(\mathbf{r}) = (r_0, \psi r_1, \ldots, \psi^{n-1} r_{n-1})$ resp. $T(\mathbf{u})$ via the bijective map $T : \mathcal{R}_q \to \mathcal{R}_q$ and subsequently computing $T(\mathbf{c}) = \mathsf{NTT}_\xi^{-1}[\mathsf{NTT}_\xi(T(\mathbf{r})) \circ \mathsf{NTT}_\xi(T(\mathbf{u}))]$. Following this approach, it is not required to double the input length to the NTT [54] and there is no need to use the less efficient FFT on the complex numbers. Moreover, from the representation of $\mathcal{R}_q$, we deduce that the number of units in $\mathcal{R}_q$ is given by $(q-1)^n = q^n(1 - 1/q)^n$ or simply by the ratio $(1 - 1/q)^n$, which is non-negligible for large enough values of q. In fact, selecting $q = 8383489$ and $n = 512$ results in a ratio close to 1. Beyond that, we also introduce a fast way to generate ring elements and the associated inverses required for tagged public keys in order to ensure CCA security. This method is mainly possible due to the existence of the NTT. In addition, we present in Section 5.3 an efficient subroutine for the LWE inversion algorithm specified in [41].

5.2 Generic Instantiation from Trapdoors for Ideal-Lattices

In general, trapdoor constructions for LWE represent powerful tools, as it allows the owner of the trapdoor to get access to the error vector and the secret in LWE instances and analyze the respective elements. In this paragraph we shortly recap the efficient ring setting [20] of the trapdoor generation algorithm [41] which is used in order to construct a public key that is indistinguishable from uniform and allows to invert LWE instances. An improved trapdoor construction is introduced in Section 5.4. We will restrict to the efficient ring variant [20]. Thus, let $k = \lceil \log q \rceil$ and $\bar{m} > 0$ be integers. The trapdoor generation algorithm takes as input an additional flag that is set either to $\mathsf{t} := \mathsf{statistical}$ or $\mathsf{t} := \mathsf{computational}$ invoking the respective trapdoor generation algorithms for public keys being either statistically or computationally close to uniform.

Trapdoor Generation

– $\underline{\mathsf{TrapGen}(1^n, \mathsf{t} := \mathsf{computational})}$ Sample a single polynomial $\mathbf{a}_1 \in \mathcal{R}_q$ uniformly at random ($\bar{m} = 1$). Let $f_{\mathbf{a}_1}(\mathbf{x}, \mathbf{y}) = \mathbf{a}_1 \cdot \mathbf{x} + \mathbf{y} \in \mathcal{R}_q$ be a function. Sample $2k$ random polynomials $\mathbf{r}_{i,j}$ according to $\mathcal{D}_{\mathbb{Z}^n, \alpha q}$ viewing polynomials as coefficient vectors with parameter αq (e.g., $\alpha q \geq 2\sqrt{n}$) for $1 \leq i \leq k$ and $j \in \{1, 2\}$. The secret key is given by $\mathsf{sk} = [\mathbf{r}_{1,1}, \ldots, \mathbf{r}_{k,1}]$ with the corresponding public key $\mathsf{pk} := \mathbf{A}$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{g}_1 - f_{\mathbf{a}_1}(\mathbf{r}_{1,1}, \mathbf{r}_{1,2}), \ \ldots \ , \mathbf{g}_k - f_{\mathbf{a}_1}(\mathbf{r}_{k,1}, \mathbf{r}_{k,2})] \ .$$

Analogously, if a tag $\mathbf{t}_u$ is used, we obtain $\mathbf{A}_u$ by multiplication of $\mathbf{t}_u$ with $\mathbf{g}_i$ for $1 \leq i \leq k$.

– $\underline{\mathsf{TrapGen}(1^n, \mathsf{t} := \mathsf{statistical})}$ Sample $\bar{m}$ polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in \mathcal{R}_q^{\bar{m}}$ uniformly at random. By $h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) = \sum_{i=1}^{\bar{m}} \mathbf{a}_i \mathbf{x}_i$ we define a generalized compact knapsack parametrized by the elements in $\hat{\mathbf{a}}$. Sample k vectors of polynomials $\hat{\mathbf{r}}_i = [\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\bar{m}}] \in \mathcal{R}^{\bar{m}}$ according to some distribution $\mathcal{D}$ and define $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ for $1 \leq i \leq k$. The public key is then given by $\mathsf{pk} := \mathbf{A}$ with

$$\mathbf{A} = [\mathbf{a}_1, \ \ldots \ , \mathbf{a}_{\bar{m}}, \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \, ,$$

where $\mathbf{g}_i$ denotes the constant polynomial consisting only of zero coefficients except for the constant term 2^{i-1}. The trapdoor polynomials define the secret key $\mathsf{sk} = [\hat{\mathbf{r}}_1, \ldots, \hat{\mathbf{r}}_{\bar{m}}]$. If a tag $\mathbf{t}_u$ is taken into account, we have

$$\mathbf{A}_u = [\mathbf{a}_1, \ \ldots \ , \mathbf{a}_{\bar{m}}, \mathbf{t}_u \cdot \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \mathbf{t}_u \cdot \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \ .$$

There exist many choices of how to select the parameter $\bar{m}$ and the distribution $\mathcal{D}$ such that the polynomials $\mathbf{a}_{\bar{m}+i}$ are statistically close to uniform over $\mathcal{R}_q$. In general, there exists an inherent relationship, where a large number $\bar{m}$ of random polynomials allows to select trapdoor polynomials $\mathbf{r}_{i,j}$ with small entries. Conversely, a small number of random polynomials leads to larger values. In fact, one can apply the regularity bound from [53, Lemma 6], which essentially states that the statistical distance of $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ from the uniform distribution over $\mathcal{R}_q$ is upper bounded by

$$\epsilon \leq \frac{1}{2}\sqrt{\left(1+\frac{q}{B^{\bar{m}}}\right)^n - 1},$$

where $\mathcal{D}$ corresponds to the uniform distribution over a set $[-b,b]^n \cap \mathbb{Z}^n$ with $B = 2b+1$. For instance, if one reconsiders the parameters $q = 8383489$ and $n = 512$ from above, it is possible to set $\bar{m} = 46$ and $b = 2^4$ in order to ensure a statistical distance of about 2^{-100}. This bound is impractical for a low value of $\bar{m}$. A better regularity bound is given by [52, Lemma 3.1] resp. [40, Corollary 7.5] for this case. We note that the parameters have to be chosen small enough in order to efficiently recover the secret when inverting LWE instances. For a computational instantiation of the public key, only one single polynomial $\mathbf{a}_1$ is sampled uniformly at random. The other polynomials are obtained via the ring-LWE distribution using $\mathbf{a}_1$. Applying this approach requires to sample the secret vector according to the discrete Gaussian distribution in order to correctly recover the error term. In the next section we present a new LWE inversion algorithm that efficiently works for arbitrarily selected moduli.

5.3 LWE Inversion for Arbitrary Modulus

In [41] an efficient LWE inversion algorithm has been proposed. However, the subroutine recovering the secret vector works very efficient only for moduli q being a power of two. For other choices one has to apply generic solutions [7, 32, 24] following [41], which are less efficient and hence not suitable for practice. Roughly speaking, the algorithm recovers the components of the secret $\mathbf{s}$ bit-wise starting from the last polynomial. Shifting $\mathbf{s}$, i.e. multiplication by powers of two, to the left deletes the most significant bits mod q. However, if the modulus is not of this form, the scaled secret is wrapped around and this approach does not work any more. We propose a new approach that works differently. The main steps are given in the table below. Let $\mathbf{A}$ be a public key instantiated as above and $\hat{\mathbf{c}} = [\mathbf{c}_1, \ldots, \mathbf{c}_{\bar{m}+k}] = \mathbf{A}\mathbf{s} + [\mathbf{e}_1, \ldots, \mathbf{e}_{\bar{m}+k}]$ be an (A-)LWE instance. The choice of the error size can be derived with the help of the following bound in order to correctly recover the secret with overwhelming probability.

Lemma 9. *Suppose that the secret key is sampled according to the discrete Gaussian distribution or uniformly at random with parameter r_{sec}. In order to correctly invert the LWE instance $\hat{\mathbf{c}}$, parameters for the error term are given by*

$$\alpha q \leq \frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}\frac{1}{\sqrt{n}}.$$

Proof. Since $\mathbf{p}_i = (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{r}_{ij})$, we have $\|\mathbf{p}_i\| \leq \|\mathbf{e}_i\| + \sqrt{\bar{m}}\,\|\mathbf{e}_j\|\,\|\mathbf{r}_{ij}\| \leq q/4$. For instance, if $\mathbf{r}_{ij}$ is chosen from a discrete Gaussian distribution with parameter r_{sec} (or alternatively components uniformly at random from $[-r_{sec}, r_{sec}] \cap \mathbb{Z}$). Then, it follows $\|\mathbf{r}_{ij}\| \leq r_{sec}\sqrt{n}$ and subsequently $\|\mathbf{e}_j\| \leq \frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}$ by rearrangement of the terms. This, however, implies that the parameter αq of the error term is bounded by $\frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}\frac{1}{\sqrt{n}}$, since $\|\mathbf{e}_j\| \leq \alpha q\sqrt{n}$. □

LWE Inversion Algorithm For Arbitrary Modulus

– Step1: Set $\tilde{\mathbf{c}}_{\bar{m}+i} = \mathbf{c}_{\bar{m}+i} - \sum_{j=1}^{\bar{m}} \mathbf{c}_j \mathbf{r}_{i,j} = \mathbf{g}_i \mathbf{s} + (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{r}_{i,j}) = \mathbf{g}_i \mathbf{s} + \mathbf{p}_i$.

– Step2: Let $\|\hat{\mathbf{p}}\|_\infty \leq b$ with overwhelming probability. When implementing the scheme, one can decrease the bound b due to direct access to the secret key. The algorithm is independently applied on each entry of $\mathbf{s} = (s_1, \ldots, s_n)$. Therefore, we start by recovering the bits of $s_1 = \sum_{i=0}^{k-1} a_i 2^i$. One proceeds successively and recovers the most significant bits at the beginning.

1. For $i = 0,\ i \leq k$: Let $\mathbf{t} = \mathbf{g}_i \mathbf{s} + \mathbf{p}_i$ and $c = t_1 - 2^i(a_{k-1}2^{k-1} + \ldots + a_l 2^l) \bmod q$, where $a_{k-1}, \ldots, a_l \in \{0,1\}$ represent all bits of s_1 recovered up to the i-th iteration.
2. Check the first bits of $c - b$ and $c + b$ in terms of equality, since $2^i s_1 \in [t_1 - b, t_1 + b]$. For instance, if the bit representation of $c - b$ resp. $c + b$ is $10100\ldots$ resp. $10110\ldots$, then s_1 (for $i = 0$) must have most significant bits 101 with $a_{k-1}a_{k-2}a_{k-3} = 101$.

Case a: If the number of recovered bits in 2. is non-zero, then jump to 1. with $i = i + 1$ and proceed with $c = t_1 - 2^i(a_{k-1}2^{k-1} + \ldots + a_l 2^l) \bmod q$, where $a_{k-1}, \ldots, a_l \in \{0,1\}$ represent all bits recovered up to the i-th iteration.

Case b: If in Step 2 no bits could be recovered due to differing bit representations, then $c \pm b$ have bit representations $10000\ldots$ and 01111. Theoretically, both representations are possible due to the perturbation vector, which is upper bounded by b (e.g. $\log q - \log b \geq 6$). Therefore, one creates two instances each for a different representation and continues the algorithm with $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 10000$ and $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 01111$, respectively.

If Case b occurs at least once, the correct secret $\mathbf{s}$ is attained, if all associated polynomials $\mathbf{e}'_i = \mathbf{c}_i - \mathbf{g}_i \mathbf{s} \bmod q$ lie in $[-4.7 \cdot \alpha q, 4.7 \cdot \alpha q]^n$ for $1 \leq i \leq k$ after normalization of the entries of $\mathbf{e}'_i$ to the range $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor + 1]^n$. For an incorrect secret there must exist an integer i such that the normalized error term $\mathbf{e}'_i$ violates this condition due to the injectivitiy of $g_{\mathbf{A}}(\cdot,\cdot)$ for the chosen parameters.

In order to estimate the bound b, which affects the running time, we compute the exact value of $\|\mathbf{r}_{ij}\|$ due to direct access to the key material. In any case, b is upper bounded by $b \leq \|\mathbf{p}_i\|_\infty = 4.7 \cdot \alpha q \|\hat{\mathbf{r}}_i\|_2 \leq 4.7 \cdot \alpha q \cdot r_{sec}\sqrt{\bar{m}n + 1}$. This mainly follows from Lemma [10, Lemma 2.4] where $\hat{\mathbf{r}}_i = (\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\bar{m}})$ is viewed as a vector of $\mathbb{Z}_q^{n\bar{m}}$.

5.4 New Trapdoor Algorithms for Ideal-Lattices

In the previous sections we gave a generic approach [20] based on [41] of how to instantiate the trapdoor construction in combination with an improved LWE inversion algorithm that allows to retrieve the error term and the secret vector from A-LWE instances. However, the number of public key polynomials in [20] is with $\bar{m} + k$ polynomials and $k = \lceil \log q \rceil$ rather large and hence not suitable for practice. In fact, the trapdoor constructions [20, 41] 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 recover the correct secret from noisy ones. 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. It can further be extended by an arbitrary number of uniform random polynomials (via the HDL mode). 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 Section 5.2. We start with a description of the new trapdoor algorithms $\mathcal{K} = (\mathsf{TrapGen}, \mathsf{LWEGen}, \mathsf{LWEInv})$. Lemma 10 shows that 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 how tags are applied, which inherently relies on uniform random secrets.

Efficient Trapdoors for A-LWE

– TrapGen(1^n) Sample two polynomials $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q$ uniformly at random. Sample 2 random polynomials $\mathbf{r}_i$ according to $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ with parameter αq (e.g., $\alpha q \geq 2\sqrt{n}$) (via the coefficient embedding) for $i \in \{1, 2\}$. The secret key is given by $\mathsf{sk} = [\mathbf{r}_1, \mathbf{r}_2]$ with the corresponding public key $\mathsf{pk} := \mathbf{A}$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{g} - (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2)] \ .$$

- For $\mathbf{q = 2^k}$, we set $\mathbf{g} = 2^{k-1}$.
- For $\mathbf{q = 3^k}$, we set $\mathbf{g} = 3^{k-1}$.
  This approach can be generalized to $q = p^k$ such that $\mathbf{g} = p^{k-1}$.
- For all other moduli, we set $\mathbf{g} = 1$.

If a tag $\mathbf{t}_u$ is applied, we obtain $\mathbf{A}_u$ via $\mathbf{t}_u \cdot \mathbf{g}$. Tagging works differently for LWE instances (see below).

– 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. The public key is given by

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{g} - (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_3)] \ .$$

- For $\mathbf{q = p^k}$ (p is usually a small integer), each coefficient 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 $i \in \{1, \ldots, n\}$. Sample for each coefficient s_i the variable $c_{i,0} \leftarrow_R \{0, \ldots, p-1\}$ uniformly at random. Then invoke $\mathbf{d} = H(c_{1,0}, \ldots, c_{n,0}) \rightarrow \mathbb{Z}^n_{p^{k-1}}$ and set $s_i = c_{i,0} + p \cdot d_i$.

  – For the special case $\mathbf{q = 2^k}$, the last bit of each coefficient is sampled uniformly at random. That is $\mathsf{LSB}(s_i) \leftarrow_R \{0, 1\}$, where LSB denotes the least significant bit. Then, invoke $\mathbf{c} = H_1(\mathsf{LSB}(s_1), \ldots, \mathsf{LSB}(s_n)) \in \mathbb{Z}^n_{2^{k-1}}$ in order to set the remaining bits of s_i. As a result, $s_i = c_i || \mathsf{LSB}(s_i) \in \mathbb{Z}_q$.

  - For the special case $\mathbf{q = 3^k}$, we sample $c_{i,0} \leftarrow_R \{0, 1, 2\}$ uniformly at random, generate $\mathbf{d} = H(c_{1,0}, \ldots, c_{n,0}) \rightarrow \mathbb{Z}^n_{3^{k-1}}$ and finally set $s_i = c_{i,0} + 3 \cdot d_i$.
- For $\mathbf{q \neq p^k}$ we sample the most significant bits of the first ℓ coefficients (e.g. $\ell = 160$ coefficients) based on the relative occurrence in $\mathbb{Z}_q$. This is, for instance, accomplished by first sampling the resp. coefficients uniformly at random from $\mathbb{Z}_q$ and subsequently taking $\mathsf{MSB}(s_i)$ as input to the random oracle

$$H : \{0, 1\}^\ell \rightarrow \mathbb{Z}_{\mathsf{MSB}(s_1)q-2^{k-1}} \times \cdots \times \mathbb{Z}_{\mathsf{MSB}(s_\ell)q-2^{k-1}} \times \mathbb{Z}^{n-\ell}_q$$

  such that $\mathbf{s} = F(\mathsf{MSB}(s_1), \ldots, \mathsf{MSB}(s_\ell))$. There are also other choices possible such as less input bits or 2 bits per coefficient etc. The output is then used to sample the remaining coefficients as well as the remaining bits of the input coefficients.

The error polynomials $\hat{\mathbf{e}}$ can now be sampled from the discrete Gaussian distribution (or in general from the error distribution χ_e). The (A-)LWE instance is then given by $\hat{\mathbf{b}} = \mathbf{A} \cdot \mathbf{s} + \hat{\mathbf{e}} \in \mathcal{R}^3_q$.

– $\underline{\mathsf{LWEInv}(\hat{\mathbf{b}}, \mathsf{sk})}$ We first compute

$$\mathbf{v} = \mathbf{g} \cdot \mathbf{s} + \mathbf{t} = \mathbf{b}_3 + \mathbf{b}_1 \cdot \mathbf{r}_1 + \mathbf{b}_2 \cdot \mathbf{r}_2$$

for some small error $\mathbf{t}$.

- For $\mathbf{q = p^k}$, the closest integer $c_{0,i} \cdot p^{k-1}$ to each coefficient v_i is recovered. This is possible if $|t_i| < p^{k-1}/2$.
  - For $\mathbf{q = 2^k}$ as a special case, the least significant bit of s_i is recovered as follows

$$\mathsf{LSB}(s_i) = 1 \text{ for } v_i \text{ closer to } q/2 \text{ than to } 0,$$
$$\mathsf{LSB}(s_i) = 0 \text{ else}.$$

    Once having recovered all $\mathsf{LSB}(s_i)$, the hash function is invoked $\mathbf{c} = H_1(\mathsf{LSB}(s_1), \ldots, \mathsf{LSB}(s_n)) \in \mathbb{Z}^n_{2^{k-1}}$ such that $s_i = c_i||\mathsf{LSB}(s_i)$.
  - For $\mathbf{q = 3^k}$ as a special case, $c_{i,0}$ of s_i is recovered as follows:

$$d_{i,0} = 0 \text{ for } v_i \text{ closer to } 0 \text{ than to } 3^{k-1} \text{ or } 2 \cdot 3^{k-1},$$
$$d_{i,0} = 1 \text{ for } v_i \text{ closer to } 3^{k-1} \text{ than to } 0 \text{ or } 2 \cdot 3^{k-1}$$
$$d_{i,0} = 2 \text{ else}.$$

    The secret is then computed as $\mathbf{s} = H(d_{1,0}, \ldots, d_{n,0})$.
- For $\mathbf{q \neq p^k}$ one obtains $\mathbf{v} = \mathbf{s} + \mathbf{t}$ such that the most significant bit of the coefficients can be recovered very easily due to an error of small size and uniform random $\mathbf{s}$. Let $d = ||\mathbf{t}||_\infty$ (see Lemma 9), then it is required to check the most significant bit of $v_i + d$ and $v_i - d$ in terms of equality, since $s_i \in [v_i + d, v_i - d]$. If not equal, the correct representation is searched using brute force and the fact that the coefficients of the error term are always small. If the correct bits have been recovered, one invokes the random oracle on these bits.

The error vector is subsequently retrieved via $\hat{\mathbf{e}} = \hat{\mathbf{b}} - \mathbf{A} \cdot \mathbf{s}$.

Lemma 10. *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 public key is indistinguishable from uniform. □

Tagging the public key in this setting works slightly different to the generic approach as described in Section 5.2. 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 the most significant bits (for $q \neq p^k$) or $d_{i,0}$ (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 vector is again uniform. We consider the case, where $q \neq p^k$, $k = \lceil \log q \rceil$ and $\mathbf{s} = F(\mathsf{MSB}(s_1), \ldots, \mathsf{MSB}(s_n))$ is uniform random for $P[\ \mathsf{MSB}(s_i) = 0\ ] = 2^{k-1}/q$ and $P[\ \mathsf{MSB}(s_i) = 1\ ] = 1 - 2^{k-1}/q$ following LWEGen as above. 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 5.2.

Theorem 5. *Assuming the hardness of (A-)LWE, the distribution of $\mathbf{A}_u \cdot \mathbf{s} + \hat{\mathbf{e}}$, where $\mathbf{s}, u$ are chosen uniformly at random and $\hat{\mathbf{e}}$ is sampled according to the discrete Gaussian distribution, is computationally indistinguishable from $\mathbf{A}_u \cdot (\mathbf{t}_u^{-1}\bar{\mathbf{s}}) + \hat{\mathbf{e}}$ for $\bar{\mathbf{s}} = F(\mathsf{MSB}(\bar{s}_1), \ldots, \mathsf{MSB}(\bar{s}_n))$ and function $F(\cdot)$ as described in LWEGen.*

Proof. In Figure 5.4 we proceed via a sequence of games, where Game 1 (original scheme) differs from Game 2 only in the way the secret polynomial is generated. In Game 1 we generate $\bar{\mathbf{s}} = \mathbf{t}_u \cdot \mathbf{s} = F(\mathsf{MSB}(\bar{s}_1), \ldots, \mathsf{MSB}(\bar{s}_n))$ uniformly at random, since multiplication of a random element with a unit does not change its distribution. Therefore, we can directly generate $\mathbf{t}_u \cdot \mathbf{s}$ uniformly at random and subsequently divide $\mathbf{t}_u$ out of $\bar{\mathbf{s}}$ prior to the generation of an (A-)LWE instance. More precisely, we select $\mathsf{MSB}(\bar{s}_i)$ with probability distribution Q defined as

$$P[\ \mathsf{MSB}(\bar{s}_i) = 0\ ] = 2^{k-1}/q$$
$$P[\ \mathsf{MSB}(\bar{s}_i) = 1\ ] = 1 - 2^{k-1}/q\,.$$

following LWEGen. The coefficients $\bar{s}_i$ are uniform random modulo q, if the remaining bits are sampled uniformly at random from $\mathbb{Z}_{\mathsf{MSB}(\bar{s}_i)q-2^k}$. This can easily be verified, since $1/q = P[\ \mathsf{MSB}(\bar{s}_i) = x\ ] \cdot |1/(x \cdot q - 2^{k-1})|$. In both cases, we obtain samples from $\mathbb{Z}_q$ with probability $1/q$ as required. Therefore, we let $\mathbf{c} = H(\mathsf{MSB}(\bar{s}_1), \ldots, \mathsf{MSB}(\bar{s}_n))$ and $\bar{s}_i = \mathsf{MSB}(\bar{s}_i)||c_i$. Therefore, in Game 2 we replace $F(\mathsf{MSB}(\bar{s}_1), \ldots, \mathsf{MSB}(\bar{s}_n))$ by a uniform element $\bar{\mathbf{s}} = \mathbf{v} \leftarrow_R \mathbb{Z}_q^n$. All other steps remain exactly the same. Therefore, Game 1 is computationally indistinguishable from Game 2. An adversary will notice a difference only with negligible probability. Finally, in Game 3 (ideal game) we simply

Game 3	Game 2	Game 1
$u \leftarrow_R \{0,1\}^n$	$u \leftarrow_R \{0,1\}^n$	$u \leftarrow_R \{0,1\}^n$
Generate $\mathbf{t}_u$	Generate $\mathbf{t}_u$	Generate $\mathbf{t}_u$
Sample $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$	$\mathsf{MSB}(s_i) \leftarrow_Q \{0,1\}$	$\mathsf{MSB}(s_i) \leftarrow_Q \{0,1\}$
	$\bar{\mathbf{s}} = \mathbf{v} \leftarrow_R \mathbb{Z}_q^n$	$\bar{\mathbf{s}} = \mathbf{t}_u \cdot \mathbf{s} = F(\mathsf{MSB}(\bar{s}_1), \ldots, \mathsf{MSB}(\bar{s}_n))$
$\hat{\mathbf{b}} = \mathbf{A}_u \mathbf{s} + \hat{\mathbf{e}}$	$\hat{\mathbf{b}} = \mathbf{A}_u(\mathbf{t}_u^{-1}\mathbf{v}) + \hat{\mathbf{e}}$	$\hat{\mathbf{b}} = \mathbf{A}_u \cdot (\mathbf{t}_u^{-1}\bar{\mathbf{s}}) + \hat{\mathbf{e}}$

Fig. 1. Game hops

make use of the fact that multiplication of a uniform random element with a unit does not change its distribution such that $\mathbf{t}_u^{-1}\mathbf{v}$ is replaced by a uniform random vector $\mathbf{s}$ resulting in an tagged (A-)LWE instance. Thus, Game 3 is indistinguishable from Game 2. As a result, an adversary is not able to distinguish between the original scheme and the ideal game unless he breaks an (A-)LWE instance. This proves our claim. $\square$

Remark 1. For $q = p^k$, the proof works similar. For a coefficient s_i, the terms $c_{i,0} \leftarrow_R \{0, \ldots, p-1\}$ are picked uniformly at random serving as input to the random oracle outputting uniform random $c_{i,k-1} \in \{0, \ldots, p-1\}$ resulting in uniform random coefficients $s_i = c_{i,0} + c_{i,1} \cdot p + ... + c_{i,k-1} \cdot p^{k-1}$.

5.5 Fast Tag Generation and Inversion

Efficient constructions of CCA and RCCA secure encryption schemes [21, 41] are often realized based on the so-called tag approach, where a random tag[41, 1, 46] is applied to the public key prior to encryption. This has been realized in several works such as [41, 21]. 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 and satisfies the unit difference property. That is, for two units $\mathbf{u}, \mathbf{v}$ 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. We describe two practical ways of doing this.

Tag Generation via NTT. Our first approach allows to efficiently generate tags and the corresponding inverses by use of the NTT, which is a key determinant for efficiency in many works. Furthermore, it offers a simple way to efficiently check, if an element belongs to the ring of units $\mathcal{R}_q^\times$. Conceptually, it relies on the NTT transform that acts as an isomorphism mapping polynomials $\mathbf{f}$ from $\mathcal{R}_q$ to $\tilde{\mathbf{f}}$ from $\mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$ via $\tilde{\mathbf{f}} = \mathbf{A} \cdot T(\mathbf{f}) \bmod q$, i.e. $\tilde{f}_i = \sum_{k=0}^{n-1} f_k \psi^k \xi^{ik}$, for $\mathbf{A} = (\xi^{ij})_{1 \le i,j \le n}$ filled with powers of an element $\xi \in \mathbb{Z}_q$ of order n [54]. Multiplying two polynomials is accomplished componentwise by use of the NTT transform applied to each polynomial.

Theorem 6 (Fast Inversion of Elements via the NTT). *Let q be a prime and n be a power of two s.t. $q \equiv 1 \bmod 2n$. Moreover, let $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ and $\tilde{\mathbf{f}} = \mathbf{A} \cdot T(\mathbf{f}) \bmod q$ for $\mathbf{f} \in \mathcal{R}_q^\times$ and $T(\mathbf{f}) = (f_0, \psi f_1, \ldots, \psi^{n-1} f_{n-1})$ with NTT transformation matrix $\mathbf{A} = (\xi^{ij})_{1 \le i,j \le n}$, where $f_0, \ldots, f_{n-1}$ denote the coefficients of the polynomial $\mathbf{f}$ (coefficient embedding). The inverse of $\mathbf{f}$ is then given by $\mathbf{f}^{-1} = \mathbf{g}$, where*

$$T(\mathbf{g}) = n^{-1}\mathbf{A}^{-1}\tilde{\mathbf{g}}$$

and $\tilde{g}_i \cdot \tilde{f}_i = 1 \bmod q$ *with* $1 \leq i \leq n$*. Moreover, an element* $\mathbf{f}$ *possesses an inverse if only if* $\tilde{f}_i \neq 0 \bmod q \ \forall i$*.*

Proof. By a simple calculation one verifies that the polynomial $\mathbf{c}$ with constant 1 and zero coefficients elsewhere has NTT transform $\tilde{\mathbf{c}} = (1, \ldots, 1)$. Hence, two elements $\mathbf{f}$ and $\mathbf{g}$ are inverses of each other in case $\tilde{g}_i \cdot \tilde{f}_i = 1 \bmod q$ for $1 \leq i \leq n$ due to componentwise multiplication $T(\mathbf{c}) = \mathsf{NTT}_\xi^{-1}[\, \mathsf{NTT}_\xi(T(\mathbf{f})) \circ \mathsf{NTT}_\xi(T(\mathbf{g}))\,] = \mathsf{NTT}_\xi^{-1}(\tilde{\mathbf{c}})$. This can be attributed to the fact that the NTT maps polynomials to the ring $\mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$, where inversion is performed componentwise over $\mathbb{Z}_q$. As a result, one can efficiently check that an element is invertible, if $\tilde{f}_i \neq 0 \bmod q \ \forall i$ is satisfied.

Lemma 11 (Generators of the Unit Group). *Let* q *be a prime satisfying* $q \equiv 1 \bmod 2n$ *and* p *a primitive root of unity in* $\mathbb{Z}_q^\times$ *such that* $p^{q-1} \equiv 1 \bmod q$*. Then, the elements* $\mathbf{c}_i = \mathsf{NTT}_\xi^{-1}(\tilde{\mathbf{c}}_i)$ *for*

$$\tilde{\mathbf{c}}_i = (1, \ldots, 1, p, 1, \ldots, 1)^\top$$

with p *at the* i*-th position are generators of* $\mathcal{R}_q^\times$*.*

Proof. Consider the elements $\mathbf{c}_{x_1,\ldots,x_n} = \mathbf{c}_1^{x_1} \cdots \mathbf{c}_n^{x_n}$. Using the NTT transformation and componentwise multiplication, we obtain

$$\mathbf{c}_{x_1,\ldots,x_n} = \mathsf{NTT}_\xi^{-1}[(p^{x_1}, \ldots, p^{x_n})],$$

which is invertible for all $x_i \in \mathbb{N}$. Since p generates the whole group $\mathbb{Z}_q^\times$, the elements $\mathbf{c}_i$ generate $\mathcal{R}_q^\times = \mathsf{NTT}_\xi^{-1}((\mathbb{Z}_q \backslash \{0\})^n)$. □

From Theorem 6 and Lemma 11 it follows that the unit group $\mathcal{R}_q^\times = \mathsf{NTT}_\xi^{-1}((\mathbb{Z}_q \backslash \{0\})^n)$ contains $(q-1)^n$ elements such that the difference $\mathbf{g} - \mathbf{f}$ of two random polynomials $\mathbf{g}, \mathbf{f} \in \mathcal{R}_q^\times$ lies again in $\mathcal{R}_q^\times$ with probability $(1 - \frac{1}{q-1})^n$. More precisely, the difference $\mathbf{g} - \mathbf{f}$ is invertible, if all components of $\mathbf{A} \cdot T(\mathbf{f} - \mathbf{g}) \bmod q$ are non-zero. For large enough q the unit difference property, that is only employed in the security proof, holds with overwhelming probability. For practice, however, other choices for q are also possible. We therefore define the tag space $\mathcal{T} = \mathcal{R}_q^\times$. Since multiplication always involves the NTT transform and the tag approach further requires to multiply the tag or its inverse, we generate from a seed the components of $\tilde{\mathbf{f}}$ rather than coefficients of $\mathbf{f}$. This approach has two major advantages over the standard approach, since one NTT transformation is saved and it is moreover very efficient to generate elements from $\mathcal{R}_q^\times$ as it only involves the generation of n random elements from $\mathbb{Z}_q \backslash \{0\}$. Generating elements directly from $\mathcal{R}_q$ and testing them for invertability is apparently not as practical. Following the proposed approach inverting $\tilde{\mathbf{f}}$ is also accomplished in the NTT. That is, the corresponding $\tilde{\mathbf{g}}$ is computed following Theorem 6. Such a strategy reduces the efforts from computing inverses in $\mathcal{R}_q^\times$ to inversion in $\mathbb{Z}_q$, which provides an efficiency advantage over standard techniques.

Tag Generation in the Ring $\mathcal{R}_{3^e}^n$. Another appropriate choice of a ring, which allows to ensure all properties of tags as described above, is $\mathcal{R}_q^n$ for $n = 2^l, l \in \mathbb{N}$ and $q = 3^e$. However, the associated algorithms are less efficient than in the former approach, since the NTT is no more applicable. On the other hand, in [17] a tag space has been defined, that satisfies the unit difference property (not only statistically).

Corollary 2 (Corollary 4, [17]). *Let* $n \geq 4$ *be a power of* 2*,* $q \geq 3$ *a power of* 3 *and* $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$*. Then any non-zero polynomial* $\mathbf{t} \in \mathcal{R}_q^n$ *of degree* $d < n/2$ *and coefficients in* $\{0, \pm 1\}$ *is invertible in* $\mathcal{R}_q$*.*

Following Corollary 2 we can set the tag space $\mathcal{T}$ to contain all non-zero polynomials $\mathbf{t} \in \mathcal{R}_q^n$ of degree $d < n/2$ with coefficients in $\{0, 1\}$. Then, the difference of two distinct polynomials is also a non-zero polynomial of degree $d < n/2$ with coefficients in $\{0, \pm 1\}$, which is due to Corollary 2 invertible. Sampling elements from the tag space $\mathcal{T}$ containing $2^{n/2-1} - 1$ tags is straightforward as a uniform random string of size $n/2 - 1$ bits suffices to represent a tag. Inversion of a tag is efficiently accomplished by use of the FFT similar to Theorem 6. Since $\mathcal{T}$ contains units $\mathbf{t}$, where both elements $\mathbf{t}, \mathbf{t}^{-1} \in \mathcal{R}_q^n$, it is possible to invert $\mathbf{t}$ just in the FFT similar to Theorem 6 but for complex numbers. The FFT backward transformation will output $\mathbf{t}^{-1}$.

In general, Corollary 2 gives rise for other rings with similar features.

5.6 CCA-secure Encryption Scheme from A-LWE

By use of our improved message embedding technique in combination with the new trapdoor construction from Section 5.2, we provide a description of the modified CCA1 secure encryption scheme from [21] adapted to the ring setting. Therefore, we call our encryption scheme LARA (**LA**ttice-based encryption of data embedded in **RA**ndomness), which reflects the fact that the data is embedded in the randomness used to ensure the security of the scheme.
We highlight that in case the public key is extended by random polynomials (HDL mode), the related error term can be sampled with an arbitrary large error size or can even be chosen to be uniform random. In fact, the trapdoor is only applied to the part of an A-LWE instance, which is associated to $\mathbf{A}'$.

To this end, the scheme is operated with two different parameters αq and βq (if $l > 0$) with an improved message expansion factor. Furthermore, define $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ for n a power of two

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi}$ with $p_1 = 2^{k_1}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi}$ with $k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$ and $p_2 = 2^{k_2}$.

The CCA1-secure encryption scheme allows to encrypt data $\mathbf{m}$ of size $c = 3n \log p_1 + l \cdot n \log p_2$ bits in the error term. For a ring $\mathcal{R}_q$ not containing an appropriate tag space, the tag is omitted resulting in a scheme ensuring only CPA-security following [21]. For the generic approach with the trapdoor construction described in Section 5.2 one proceeds analogously, however by use of the LWE inversion algorithm from Section 5.3 for decryption.

We note that the unrestricted and flexible error term $\hat{\mathbf{e}}_2$ can serve to transport lattice-based signatures due to coinciding distributions [21]. That is, signatures can directly be exploited as the error term without any initial transformations or encodings. This enhances the security of the scheme as it additionally provides an authentication mechanism for encrypted messages.

In Theorem 7 we prove CCA2-security of our encryption scheme in the random oracle model.

Theorem 7. *In the random oracle model the encryption scheme* $\mathsf{LARA}_{\mathsf{CCA2}}$ *from below is CCA2-secure assuming the hardness of decision ring ALWE.*

Proof (sketch). In the description below, we also show how to obtain CCA2-secure constructions by use of an additional hash function H_3, where $h = H_3(\mathbf{s}, \hat{\mathbf{e}}, u)$ is appended to the ciphertext. Since it already guarantees CCA1 security (see [21] for further details), it is only needed to prove security of the second stage in the CCA2 game similar to [21] as h is uniform random and does not change the view of the adversary when appended to the ciphertext. A PPT-adversary is no more able to alter the ciphertext without the knowledge of the correct input (of high min-entropy) to

H_3. This can be seen as $\mathbf{s}, \hat{\mathbf{e}}, u$ uniquely determine a ring-(A)LWE instance and modifying the ciphertext changes any of these elements. But in order to succeed the adversary needs also to update h, which is only possible with the correct input to H_3 (see [21]). □

$\mathsf{LARA_{CCA}}$:CCA-secure Encryption Scheme - Ring Variant

$\mathsf{KGen}(1^n)$: Generate $\mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathcal{R}_q^{3+l}$, where $(\mathbf{A}', \hat{\mathbf{r}}) \leftarrow \mathsf{TrapGen}(1^n)$ (see Section 5.4) and $\mathbf{A}' = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3] \in \mathcal{R}_q^3$ with $\mathsf{pk} := \mathbf{a}_3$ and $\mathsf{sk} := \hat{\mathbf{r}} = [\mathbf{r}_1, \mathbf{r}_2]$.
- HDL mode: High data load encryption $l > 0$ with $\mathbf{A}'' \leftarrow H_1(\mathsf{seed})$, else $l = 0$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^c$ with $c = 3n \log p_1 + l \cdot n \log p_2)$: Sample a uniform random tag $\mathbf{t}_u \in \mathcal{T}$ (see Section 6) and generate $\mathbf{A}'_u$ following Section 5.4. The secret

$$\mathbf{t}_u \cdot \mathbf{s} = F(\mathbf{s})$$

is generated according to $\mathsf{LWEGen}(1^n)$ (see Section 5.4). Compute

$$\hat{\mathbf{v}} = (\mathbf{v}_1, \ldots, \mathbf{v}_{3+l}) = \mathsf{encode}(H_2(\mathbf{s}) \oplus \mathbf{m}) \in \mathbb{Z}_{p_1}^3 \times \mathbb{Z}_{p_2}^l,$$

where $H_2 : \{0,1\}^* \to \{0,1\}^c$ is a cryptographic hash function modeled as random oracle. Finally, sample
- $\mathbf{e}_i \leftarrow_R \mathcal{D}_{p_1\mathbb{Z}^n + \mathbf{v}_i, \alpha q}$ for $1 \leq i \leq 3$
- $\mathbf{e}_i \leftarrow_R \mathcal{D}_{p_2\mathbb{Z}^n + \mathbf{v}_i, \beta q}$ for $4 \leq i \leq l+3$

viewing the error polynomials $\hat{\mathbf{e}} = [\mathbf{e}_1, \ldots, \mathbf{e}_{l+3}]$ as vectors via the coefficient embedding. The ciphertext (tagged A-LWE instance) is then given by

$$\hat{\mathbf{c}} = (\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2) = \mathbf{A}_u \cdot \mathbf{s} + \hat{\mathbf{e}}, u\,.$$

- **(CCA2 security)** Output $\hat{\mathbf{c}}, u, h = H_3(\mathbf{s}, \hat{\mathbf{e}}, u) \in \{0,1\}^\lambda$.

$\mathsf{Dec}(\mathsf{sk}, (\hat{\mathbf{c}}, u, h))$: Compute $\mathbf{A}'_u = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{t}_u \cdot \mathbf{g} - (\mathbf{a}_1 \cdot \mathbf{r}_1 + \mathbf{a}_2 \cdot \mathbf{r}_2)]$, then
1. If parsing $\hat{\mathbf{c}}$ causes an error or $u = 0$ output $\perp$, otherwise invoke

$$\mathbf{t}_u \cdot \mathbf{s} \leftarrow \mathsf{LWEInv}(\hat{\mathbf{c}}_1, \mathsf{sk}),$$

where $\hat{\mathbf{c}}_1 = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ and compute

$$\hat{\mathbf{e}} = \hat{\mathbf{c}} - [\mathbf{A}'_u \mid \mathbf{A}''] \cdot \mathbf{s}.$$

2. If $\| \hat{\mathbf{e}}_1 \| > \alpha q \sqrt{3n}$ or $\| \hat{\mathbf{e}}_2 \| > \beta q \sqrt{ln}$, output $\perp$.
3. **(CCA2 security)** Check that $h = H_3(\mathbf{s}, \hat{\mathbf{e}}, u)$, else output $\perp$.
4. The message is retrieved via

$$\mathbf{m} = \mathsf{decode}(\hat{\mathbf{e}}_1 \bmod p_1, \hat{\mathbf{e}}_2 \bmod p_2) \oplus H(\mathbf{s}).$$

5.7 Remark: Lindner-Peikerts Encryption Scheme is not CCA-secure

In the following section, we briefly discuss why the encryption scheme due to Lindner and Peikert does only provide CPA security and not CCA. By allowing the adversary oracle access to the decryption algorithm in the first phase of the CCA game, he can prepare ciphertexts in such a way that the oracle will send the secret key as its decryption to the adversary. Therefore, suppose first the secret polynomial $\mathbf{r}_1$ is binary. Then, the adversary proceeds as follows. Here, the public key is given by $\mathbf{p} = \mathbf{r}_2 - \mathbf{a}\mathbf{r}_1$, where $\mathbf{r}_1 \in \mathcal{R}_2$.

- $(\mathbf{c}_1, \mathbf{c}_2)$ will be sent to the decryption oracle, where $\mathbf{c}_1 = \mathbf{a} + q/2$ and $\mathbf{c}_2 = \mathbf{p}$
- The decryption oracle will respond by computing $\lceil (\mathbf{c}_1\mathbf{r}_1 + \mathbf{c}_2)/\frac{q}{2} \rfloor = \lceil (\frac{q}{2}\mathbf{r}_1 + \mathbf{r}_2)/\frac{q}{2} \rfloor = \mathbf{r}_1$ and sending the message $\mathbf{r}_1$ to the adversary.

If the error and the secrets are coming from a wider distribution such that $\|\mathbf{r}_i\|_\infty \leq b$ the adversary can obtain the secret key by at most $k = \log q$ queries to the decryption oracle. For simplicity, let $q = 2^k$. Then, the adversary proceeds as follows.

For $i = 1$ to $k - 1$:

- $(\mathbf{c}_1^{(i)}, \mathbf{c}_2^{(i)})$ will be sent to the decryption oracle, where $\mathbf{c}_1^{(i)} = \mathbf{a} + \frac{q}{2^i}$ and $\mathbf{c}_2^{(i)} = \mathbf{p} - \frac{q}{2^i} \cdot \bar{\mathbf{r}}_1^{(i)}$, and $\mathbf{r}_1^*$ accumulates all recovered bits of $\mathbf{r}_1$ up to the i-th iteration $(\bar{\mathbf{r}}_1^{(1)} = 0)$.
- The oracle will decrypt to $\mathbf{t}^{(i)} = \lceil (\mathbf{c}_1^{(i)} \cdot \mathbf{r}_1 + \mathbf{c}_2^{(i)})/\frac{q}{2} \rfloor = \lceil (\frac{q}{2^i} \cdot (\mathbf{r}_1 - \bar{\mathbf{r}}_1^{(i)}) + \mathbf{r}_2)/\frac{q}{2} \rfloor$. The adversary will then compute $\bar{\mathbf{r}}_1^{(i+1)} = \bar{\mathbf{r}}_1^{(i)} + 2^{i-1} \cdot \mathbf{t}^{(i)} \bmod q$

After the $(k-1)$-th iteration the advesary obtains $\bar{\mathbf{r}}_1^{(k-1)} = \mathbf{r}_1$.

6 Security Analysis

In order to estimate the security of the scheme, we mainly adopt the embedding approach, which is more appropriate for a bounded number of samples as observed in [8]. The distinguishing attack, however, provides better results if the number of available LWE samples is large. In principal, the embedding approach proceeds by reducing the LWE problem to the unique shortest vector problem (u-SVP). One mainly differentiates the standard embedding approach [4] with the variant that has recently been shown to be more efficient especially for a small number of samples [8]. In the following, we give a description of the main ingredients of the embedding approach for the matrix variant. Our analysis is based on the fact that (A-)LWE instances with a uniform random secret are not harder than with secrets sampled from the error distribution.

6.1 Embedding Approach

Let $(\mathbf{A}^\top, \mathbf{b})$ be an LWE sample with $\mathbf{b} = \mathbf{A}^\top\mathbf{s} + \mathbf{e} \bmod q$, where $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$ follows the discrete Gaussian distribution. The idea of this attack is to turn the problem of finding a closest lattice point (CVP) to the target vector $\mathbf{b}$ into a unique-SVP problem. Therefore, the authors start with a carefully crafted matrix $\mathbf{A}_e = \begin{bmatrix} \mathbf{A}^\top & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$ and the corresponding q-ary embedding lattice

$$\Lambda_q(\mathbf{A}_e) = \{\mathbf{u} \in \mathbb{Z}^{m+1} \mid \exists \mathbf{x} \in \mathbb{Z}^{n+1} \text{ s.th. } \mathbf{A}_e\mathbf{x} \equiv \mathbf{u} \bmod q\}.$$

A short lattice point is given by $\mathbf{u} = \begin{pmatrix} \mathbf{e} \\ 1 \end{pmatrix} = \mathbf{A}_e \begin{pmatrix} -\mathbf{s} \\ 1 \end{pmatrix}$. Its length is upper bounded by $s\sqrt{m}$. In [22] it was conjectured that a lattice basis reduction algorithm will find a shortest vector with high probability if

$$\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \delta^{n(\Lambda)} \cdot \tau \quad (1)$$

is satisfied for an algorithm characteristical Hermite-factor δ and a lattice or algorithm specific constant $\tau \approx 0.4$. One observes by this relationship that the gap between the first and second successive minimum of the lattice Λ plays an important role for the success probability of the underlying algorithm. In order to estimate the successive minima we need the determinant of the lattice, which is given by $\det(\Lambda_q(\mathbf{A}_e)) = q^{m-n}$ with overwhelming probability for a random lattice. Subsequently, by use of the Gaussian heuristic one can deduce estimations for the lengths of the successive minima

$$\lambda_i(\Lambda) \approx \frac{\Gamma(1 + n(\Lambda)/2)^{1/n(\Lambda)}}{\sqrt{\pi}} \det(\Lambda)^{1/n(\Lambda)}. \tag{2}$$

Substitution of λ_1 and λ_2 in equation (1) by equation (2) and rearrangement of the terms provides

$$\delta \approx \left(\frac{\Gamma(1 + \frac{m+1}{2})^{1/(m+1)}}{\sqrt{\pi \cdot m} \cdot \tau \cdot s} q^{\frac{m+1-n}{m+1}} \right)^{1/(m+1)}$$

for the required Hermite factor in order to break LWE samples via the embedding attack, where $dim(\Lambda_q(\mathbf{A}_e)) = m + 1$. Now, it is possible to estimate the time required to successfully mount an attack on LWE and subsequently derive the bit security of the underlying LWE instances. In particular, it is needed to preprocess the lattice basis by a strong basis reduction algorithm such as BKZ or the more advanced BKZ 2.0 in order to achieve the required Hermite factor. In [34] a model has been proposed that is deduced by a limited set of experiments and subsequent extrapolations

$$\log_2(T(\delta)) = 1.8/\log_2(\delta) - 110\,. \tag{3}$$

These experiments were performed on a computer allowing for $2.3 \cdot 10^9$ operations per second.

The standard embedding approach from above is not so efficient in case one is given only a few LWE samples. As a result, the optimal attack dimension is never reached. To circumvent this situation, one changes the embedding lattice as follows

$$\Lambda_q^{\perp}(\mathbf{A}_o) = \{\mathbf{v} \in \mathbb{Z}^{m+n+1} \mid \mathbf{A}_o \cdot \mathbf{v} = \mathbf{0} \mod q\}$$

and hence allows for a finer analysis, where $\mathbf{A}_o = \left[\mathbf{A}^\top \mid \mathbf{I} \mid \mathbf{b}\right]$. Following this approach, one increases the dimension from $m + 1$ to $m + n + 1$. By a trivial computation one verifies that $\mathbf{u} = \left(\mathbf{s}, \mathbf{e}, -1\right)^T \in \Lambda_q^{\perp}(\mathbf{A}_o)$. The length of this vector is bounded by $s\sqrt{m+n+1}$. Using the framework from above with $\det(\Lambda_q^{\perp}(\mathbf{A}_o)) = q^m$ one obtains the estimated security level. The ring variant requires to multiply the number of polynomials by n, i.e. $m = t \cdot n$ for $\mathbf{A} \in \mathcal{R}_q^t$.

6.2 Analysis of Key Recovery Attacks

The encryption schemes presented in the previous sections give rise to a new discrete Gaussian sampler that Instead of breaking the encryption scheme by attacking the ciphertext, it is possible to recover the key. Therefore, we have to differentiate between the statistical and computational instantiation of the public key. We, therefore, restrict to the ring variant.

- **Statistical Instantiation:** In order to recover the key, we have to solve the ring-ISIS problem for k instances $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ with $1 \leq i \leq k$ and uniform random vector of polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in \mathcal{R}_q^{\bar{m}}$. In this case, we have to find preimages $\hat{\mathbf{x}}_i$ such that $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)$ and the inequality $\|\mathbf{p}_i\| =\| \mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{x}_{ij} \| \leq \|\mathbf{e}_i\| + \|\mathbf{e}_j\| \, \|\mathbf{x}_{ij}\| \sqrt{\bar{m}} \leq q/4$

is satisfied for $1 \leq i \leq k$ (see Section 5.3). Lemma 9 gives a reasonable upper bound on the length $\|\hat{\mathbf{x}}_i\|_2$ for $1 \leq i \leq k$. Then, we have to find a short vector in the lattice $\Lambda^{\perp}(\hat{\mathbf{c}}_i) = \{\hat{\mathbf{x}} \in \mathcal{R}^{\bar{m}+1} \mid \sum_{i=1}^{\bar{m}+1} \mathbf{c}_i \mathbf{x}_i \equiv \mathbf{0} \bmod q\}$, where $\hat{\mathbf{c}}_i = [\hat{\mathbf{a}}, -h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)]$. In fact, the vector $[\hat{\mathbf{r}}_i, 1]$ is contained in the lattice, which is by construction a solution to the ring-SIS problem. By means of of the embedding approach explained in the previous section one derives the bit security.

- **Computational Instantiation:** Here, the public key is composed by a uniform random polynomial $\mathbf{a}_1$ and k ring-LWE instances (Section 5.4) or one ring-LWE instance (Section 5.2) following the improved construction. For each sample in the public key a new secret and error has been generated. Therefore, we are required to consider each sample independently within the security analysis. To this end, we can use the embedding approach from above or alternatively the approach from [43] in order to estimate its security.

7 A Fast Discrete Gaussian Sampler - FastCDT

In this section we introduce a novel discrete Gaussian sampler that we call FastCDT that is tailored to respect the way of embedding data into the error term during encryption. It is highly efficient and outperforms even the currently most efficient CDT based Gaussian samplers in terms of running time and working memory. The respective tables of our new sampler can also be generated on the fly at the cost of a slight increase of running time due to the generation of at most 44 elements (statistical distance $\epsilon \leq 2^{-100}$), almost the whole probability mass is concentrated on the 11 mid elements in most cases. It is therefore efficiently applicable in constrained devices characterized by low resources. This sampler may be combined with other techniques such as guided tables or the convolution method described in [50].

In fact, the basic tool required to realize such a sampler is given by Lemma 3 instantiated with a lattice of the form $\Lambda = p \cdot \mathbb{Z}^m$, for an integer $p > 0$. In order to sample a vector $\mathbf{e} \in \mathbb{Z}^m$ statistically close to $\mathcal{D}_{\mathbb{Z}^m,\alpha q}$ with $\eta_\epsilon(\Lambda) \leq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \approx 4.7 \cdot p = \alpha q$, one samples a vector $\mathbf{b} \leftarrow_R \mathbb{Z}_p^m$ uniformly at random and subsequently a discrete Gaussian from $\mathcal{D}_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$ following Lemma 3. This seems to be more expensive due to two sampling steps, but a closer look reveals that almost the whole entropy of $\mathbf{e}$ is coming from a uniform random source, which is far more efficient than sampling discrete Gaussians. This is particularly interesting for constrained devices preferring uniformly sampled vectors over other distributions. The remaining entropy is obtained via the sampling step $\mathcal{D}_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$. The support $\mathsf{supp} = \{\mathbf{b} + p\mathbb{Z}^m\}$ of this distribution consists of at most 44 elements $\mathsf{supp} \cap [-4.7^2 p, 4.7^2 p]$ with overwhelming probability, which allows to be generated dynamically rather than in the key generation step since the number of required CDT table entries amounts to 44 elements at most. But almost the whole probability mass is concentrated on the 10 mid elements (one case with 11 elements) following Lemma 12, since two neighboring elements of $\mathbf{b} + p\mathbb{Z}^m$ have a large relative distance of p decreasing the corresponding probabilities rapidly. This makes the sampler very flexible allowing to be instantiated following one of the 3 approaches presented below, which provide different trade-offs between flexibility, storage requirements and running time. This is not possible when using Knuth-Yao or the standard CDT technique, which depends on p with a table containing $4.7 \cdot \alpha q \approx \lceil 4.7^2 \cdot p \rceil$ entries. Our algorithm requires a constant table size of 44 elements for arbitrary p. As another performance advantage, one observes based on the shape of the support that almost the whole probability mass lies on the set $S_i = [-5p, 5p] \cap \{b_i + p\mathbb{Z}\}$ (see Lemma 12), making binary search less efficient than testing the CDT table entries of S_i via linear search starting with b_i or $p - b_i$. Only with small probability, one looks into the remaining table entries. In the following, we highlight the various trade-offs between flexibility, storage requirements and running time.

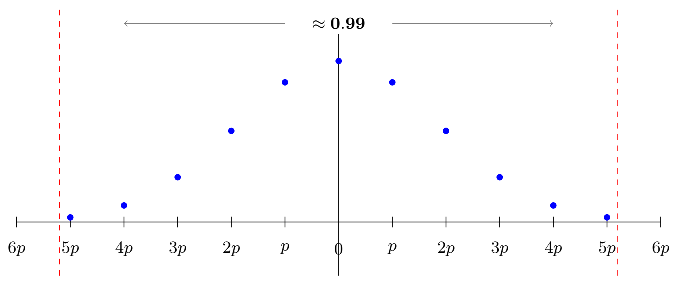

Fig. 2. FastCDT: Probability Distribution $\mathcal{D}_{b+p\mathbb{Z},\alpha q}$ for all $p > 0$, $b = 0$ and $\alpha q = p \cdot 4.7$.

1. Generate the whole table at the beginning. This will occupy as much memory as the standard CDT approach. The algorithm will start to operate on the dedicated $10-11$ elements, out of which only 5 (either left or right) are considered via linear search once having sampled the partition $\mathbf{b} \in \mathbb{Z}_p^m$. This set of elements encompasses almost the whole probability mass. In case CDT does not find the correct value within this range, the standard binary search algorithm or linear search is invoked on the remaining elements out of 22. The expected value for the number of table look ups is approximately 2.
2. Generate everything on the fly. Whenever a partition has been sampled, the algorithm starts generating the associated 44 elements, out of which an element is sampled by first considering the $10-11$ mid elements via linear search. This approach is interesting for high performance processors maintaining a low working memory. But also for constrained devices with low memory capacities it can be useful if the parameter is huge.
3. Generate only the $10-11$ elements per partition $\mathbf{b} \in \mathbb{Z}_p^m$. This is done at the start of the algorithm (or on the fly). The remaining ones out of 22 (only half of the elements are needed) are generated in case the algorithm gets out of the range (see Lemma 12).

In summary, our approach is identical to the standard CDT approach for $p = 1$ or equivalently $\alpha q = \omega(\sqrt{\log n}) \approx 4.7$. However, if $p > 0$ our approach requires only to sample an additional uniform vector $\mathbf{b} \in \mathbb{Z}_p^m$, whereas the other steps remain exactly as efficient as for $p = 1$. In the case of the standard CDT sampler the table size increases and hence the number of table lookups. For instance, $\mathsf{FastCDT}$ requires to sample 2 uniform random elements and about 2 table lookup (5 look-ups in the worst-case) as compared to the standard CDT sampler with one uniform random element and about $\log(4.7p) \approx 2 + \log(p)$ table look ups in the worst-case. Furthermore, our approach can generate the table elements (only 44 elements) dynamically, if required, as opposed to fill the complete table with $4.7 \cdot \alpha q$ elements in the key generation phase. The most flexible approach can be advantageous, if αq is large. It is also noteworthy to mention, that $\mathsf{FastCDT}$ combines many sub-samplers, because it can also be utilized to sample a discrete Gaussian vector from any given partition modulo p. Remarkably, both the standard CDT and $\mathsf{FastCDT}$ samplers claim the same total storage size for the respective tables. In the figures above, we illustrate the algorithms of $\mathsf{FastCDT}$, where Algorithm 1 initializes the respective tables, Algorithm 2 samples a coset $\mathbf{b} \in \mathbb{Z}_p^m$ uniformly at random and Algorithm 3 selects an element from the induced table via linear search starting with the mid element.

Lemma 12. *Let $p \in \mathbb{Z}$ and $s = p \cdot \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \approx p \cdot 4.7$. Then, $|S_i| \leq 11$ and $\rho(S_i)/\rho(\Lambda_p^\perp) \approx 0.99$ for $\Lambda_i^\perp = i + p\mathbb{Z}$ and $S_i = \{i + p\mathbb{Z}\} \cap [-5p, 5p]$.*

Proof. First, we obtain $\rho(\Lambda_i^\perp) = \rho(\Lambda_p^\perp) \pm \mathsf{negl}(n)$ as per Lemma 4, since $\rho_{s,-i}(\Lambda_p^\perp) = \rho_s(\Lambda_p^\perp + i) = \rho_s(\Lambda_i^\perp)$ and $\eta_\epsilon(\Lambda_p^\perp) \leq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ for a negligible parameter $\epsilon \leq 2^{-100}$

according to Lemma 6. Furthermore, it suffices to consider the restricted set $\Lambda_p^\perp \cap [-4.7s, 4.7s]$ with at most 45 elements, which are assigned the overwhelming portion of the probability mass. One easily obtains the following interesting identity

$$\begin{aligned}\rho(\Lambda_p^\perp) = \sum_{j=-\infty}^{\infty} e^{-\pi\cdot\frac{(j\cdot p)^2}{(4.7\cdot p)^2}} &= -1 + 2\sum_{j=0}^{\infty} e^{-\pi\cdot\frac{(j\cdot p)^2}{(4.7\cdot p)^2}} \\ &= -1 + 2\sum_{i=0}^{\infty}(e^{-\pi/4.7^2})^{j^2} \\ &= 4.7\end{aligned}$$

with high probability for all p. This series represents a theta function independent from p and can hence be computed using only a small number of representatives. Furthermore, one easily verifies the inequality $|S_i| \leq 11$ such that we have only to show that $\rho(S_i)/\rho(\Lambda_p^\perp) \approx 0.99$. There are only two cases to be considered, $i = 0$ and $i > 0$. As for $i = 0$, we obtain a finite series independent from p:

$$\rho(S_i)/\rho(\Lambda_p^\perp) = \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{5} e^{-\pi\cdot\frac{(j\cdot p)^2}{(4.7\cdot p)^2}} = \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{5} e^{-\pi\cdot\frac{j^2}{4.7^2}} \approx 0.997\,.$$

For $i > 0$, we have $|S_i| = 10$ and

$$\begin{aligned}\rho(S_i)/\rho(\Lambda_p^\perp) = \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{4} e^{-\pi\cdot\frac{(j\cdot p+i)^2}{(4.7\cdot p)^2}} &= \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{4} e^{-\pi\cdot(\frac{j+i/p}{4.7})^2} \\ \geq 1 - 2\cdot\frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=5}^{22} e^{-\pi\cdot\frac{j^2}{4.7^2}} \approx 0.985\end{aligned}$$

□

Algorithm 1: Building CDT Arrays

```
Data: Integer $p$, $\alpha q = p\cdot\omega(\sqrt{\log n}) \approx p\cdot 4.7$

$\mathsf{supp}_i = \{i + p\mathbb{Z}\} \cap (-4.7\alpha q, 4.7\alpha q)$
$c = \max_{0\leq i\leq p-1} |\mathsf{supp}_i| \leq 45$

for $i = 0 \to p-1$ do
    for $j = -22 \to 22$ do
        $\mathsf{CDT}_{b_i}(j) = \sum_{l=-22}^{j} \rho(i + l\cdot p)/\rho(\mathsf{supp}_i)$
    end
end
```

Algorithm 2: FastCDT

```
Data: Integer $p$, $\alpha q = p\cdot\omega(\sqrt{\log n}) \approx p\cdot 4.7$
Sampling from $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ :
    $\mathbf{b} \leftarrow_R \mathbb{Z}_p^n$
    $\mathbf{z} \leftarrow_R \mathcal{D}_{\mathbf{b}+p\mathbb{Z}^n,\alpha q}$ via Algorithm 3 for $\mathbf{b}, p, n$
Output  z
```

Algorithm 3: CDT Sampling

```
Data: $p \in \mathbb{Z}, n \in \mathbb{N}, \mathbf{b} \in \mathbb{Z}_p^n$
Given $\mathsf{CDT}_{b_i} : [-22, 22] \cap \mathbb{Z} \to [0, 1]$ for $1 \le i \le n$
for $i = 1 \to n$ do
    $r \leftarrow_R [0, 1]$
    # linear search
    for $j = 0 \to \pm 5$
    $z_i \leftarrow \mathsf{linSearch}(r, \mathsf{CDT}_{b_i}(j))$
    if $z_i = \perp$
    # binary search
    for $j = \pm 6 \to \pm 22$
      $z_i \leftarrow \mathsf{binSearch}(r, \mathsf{CDT}_{b_i}(j))$
end
```

Previous works relied on the entropy of normally distributed variables and argued based on experiments that the equation also holds for discrete Gaussians. In Lemma 13 we deduce a closed expression for the entropy of discrete Gaussian distributed vectors.

Lemma 13. *Let $p \in \mathbb{Z}$ and $s = p \cdot \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \approx p \cdot 4.7$. Then, the entropy of a discrete Gaussian $r \leftarrow_R \mathcal{D}_{\mathbb{Z},s}$ is given by $\mathbb{H}_\infty(r) = \log(e^{1/2}s)$ (with overwhelming probability).*

Proof. We can restrict the support of the sampler to $\mathbb{Z} \cap [-4.7s, 4.7s]$ which is assigned the overwhelming portion of the probability mass following Lemma [10, Lemma 2.4]. From Lemma 2 and the algorithms for $\mathsf{FastCDT}$, we have to sample an integer b uniformly at random from $\mathbb{Z}_p$. Subsequently, we sample a discrete Gaussian via the distribution $\mathcal{D}_{b+p\mathbb{Z},s}$. We note that $\rho(\Lambda_i^\perp) = \rho(\Lambda_p^\perp) \pm \mathsf{negl}(n)$ as per Lemma 4 and Lemma 12. The entropy is then given by

$$\mathbb{H}_\infty(r) = -\sum_{i=-\infty}^{\infty} P(i) \cdot \log P(i) \;=\; -\sum_{i=0}^{p-1} \sum_{j=-\infty}^{\infty} \frac{\rho(i + j \cdot p)}{p \cdot \rho(\Lambda_i^\perp)} \cdot \log\left(\frac{\rho(i + j \cdot p)}{p \cdot \rho(\Lambda_i^\perp)}\right)$$

$$= -\frac{1}{p \cdot \rho(\Lambda_p^\perp)} \sum_{i=0}^{p-1} \sum_{j=-\infty}^{\infty} \rho(i + j \cdot p) \cdot \log(\rho(i + j \cdot p)) + \log(p \cdot \rho(\Lambda_p^\perp))$$

For the left term of the last equation, we can deduce a simple series with parameter 4.7 for all p. In fact, we have

$$-\sum_{j=-\infty}^{\infty} \rho(x + j \cdot p) \cdot \log(\rho(x + j \cdot p)) = \sum_{j=-\infty}^{\infty} e^{-\pi \frac{(x+j\cdot p)^2}{s^2}} \cdot \pi \frac{(x + j \cdot p)^2}{s^2} \log e$$

$$= \sum_{j=-\infty}^{\infty} e^{-\pi \frac{(x/p+j)^2}{4.7^2}} \cdot \pi \frac{(x/p + j)^2}{4.7^2} \log e$$

A trivial computation shows that the maximum of $\rho(x) \cdot \log(\rho(x))$ is at $x = \frac{s}{\sqrt{\pi}}$ with $\rho(x) \cdot \log(\rho(x)) = 1/e$. Furthermore, we see from the last equation that we need only to consider

$x \in [0,p] \cap \mathbb{Z}$ or equivalently $y = x/p \in [0,1] \cap \mathbb{Q}$ for arbitrary p. Thus, we have

$$\sum_{j=-\infty}^{\infty} e^{-\pi \frac{(y+j)^2}{4.7^2}} \cdot \pi \frac{(y+j)^2}{4.7^2} \log e = 2.35 \cdot \log e,$$

which is deduced from the expectation value via $\frac{\pi \cdot \rho(\Lambda_p^{\perp}) \cdot \log e}{4.7^2} E[x^2]$, since

$$E[(x+c)^2] = E[x^2] = \sum_{j=-\infty}^{\infty} \frac{e^{-\pi \frac{x^2}{4.7^2}}}{\rho(\Lambda_p^{\perp})} \cdot x^2 = \frac{4.7^2}{2 \cdot \pi},$$

where the first equation follows from [42] for $c \in \mathbb{R}$. In fact, it has been shown that $E[x^2] = \frac{4.7^2}{2\pi}$ and $E[(x-c)^2] \leq 4.7^2 \cdot \left(\frac{1}{2\pi} + \frac{\epsilon'}{1-\epsilon'}\right)$ for $4.7 \geq 2\eta_{\epsilon'}(\mathbb{Z})$. Following this, we need to solve the equation $4.7 = 2 \cdot \sqrt{\frac{\ln(2(1+1/\epsilon'))}{\pi}}$ with respect to ϵ'. Since ϵ' is negligible, we still obtain an expression very close to $E[x^2]$.

Using this, we deduce the following expression for the term of equation (3)

$$\frac{1}{p \cdot \rho(\Lambda_p^{\perp})} \sum_{i=0}^{p-1} \sum_{j=-\infty}^{\infty} \rho(i + j \cdot p) \cdot \log(\rho(i + j \cdot p)) = \frac{p \cdot 2.35 \cdot \log e}{p \cdot \rho(\Lambda_p^{\perp})} = 0.5 \log e$$

Hence, we have $\mathbb{H}_{\infty}(r) = \log(p \cdot \rho(\Lambda_p^{\perp})) + 0.5 \log e = \log(e^{1/2} s)$, where $\rho(\Lambda_p^{\perp}) = 4.7$ with high probability following Lemma 12. $\square$

Table 2 compares different state-of-the-art discrete Gaussian samplers such as the standard CDT sampler and Knuth-Yao versus FastCDT. In fact, by use of FastCDT we realize improvement factors of about $1.7 - 2.0$ as compared to Knuth-Yao and even $1.5 - 2.6$ in comparison to the standard CDT sampler. Theoretically, one would expect the FastCDT sampler to be for a large p essentially as fast as with small parameters for a given efficient uniform random source. This observation is due to the constant table size at runtime. However, the small cache memory of today's architectures causes delays in case a requested table is moved into the cache. This occurs when p becomes large and hence the number of tables increases.

Parameter	**Knuth-Yao**	**CDT**	**FastCDT**
p	Timings in sec.	Timings in sec.	Timings in sec.
16	5.6	5.1	3.3
128	6.6	7.3	3.4
1024	7.7	9.6	3.7
4096	8.4	11.3	4.3
Improvement Factor	$\times$**1.7** $-$ $\times$**2.0**	$\times$**1.5** $-$ $\times$**2.6**	-

Table 2. Comparison of different samplers for 10^8 samples

We note that using the Rény divergence [9] in place of the statistical distance one might further decrease the table size and thus improve the sampler. We also observe that our sampler might be resistant against side channel attacks due to the uniform random vector of high entropy, that is drawn during the sampling procedure. Moreover, the discrete Gaussian sampler can also be applied for any real parameter $\alpha q \geq 4.7$ at a slight increase of the table size. In this case, we set $p = \lfloor \alpha q / 4.7 \rfloor$ with $\mathsf{const} = \alpha q / p$. Now, for each coset $b + p\mathbb{Z}$ the table is filled with a few additional entries.

8 Software Implementation and Performance Analysis

At the implementation front we consider several optimizations and present a description of the main ingredients in Section 8.1. As for the polynomial representation and optimizations with respect to the NTT we also refer to the work [26], which provides a description of an optimized implementation exploiting the single-instruction multiple-data (SIMD) instructions of the AVX instruction set extension in modern chipsets. Accordingly, we give an optimized FFT implementation for $q = p^k$. Furthermore, we applied our FastCDT sampler from Section 7 whenever a discrete Gaussian is consumed.

Lemma 14 (Lemma 2, [20]). *Let* $\mathbf{v} \in \mathbb{Z}^n$ *be a vector with* $\| \mathbf{v} \|_2 < b \cdot \sqrt{n}$*. Then the maximal bit size needed to store this vector is bounded by* $n \cdot (1 + \lceil \log_2(b) \rceil)$ *bits.*

Parameter	Description	Used for
n	Dimension	$n = 512$
q	Modulus	$q \equiv 1 \bmod 2n$ or $q = 2^k$
$\mathcal{R}_q$	Cyclotomic ring	$\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$
p_1, p_2	Message range	$p_i = 2^{x_i}$, x_i bits/coeff.
αq	Error distribution $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ associated to $\mathbf{A}'$	$\alpha q = 4.7 \cdot p_1$
βq	Error distribution $\mathcal{D}_{\mathbb{Z}^n,\beta q}$ associated to $\mathbf{A}''$	$\beta q = 4.7 \cdot p_2$
m	Number of polynomials in $\mathbf{A}'$	$\mathbf{A}' \in \mathcal{R}_q^m$, e.g. $m = 3$
l	High data load encryption mode: $l > 0$	$\mathbf{A}'' \in \mathcal{R}_q^l$
μ	Seed to generate $\mathbf{A}'' \in \mathcal{R}_q^l$	
r_{sec}	Parameter of the secret key distribution	$\mathcal{D}_{\mathbb{Z}^n,r_{sec}}$ or $\mathcal{U}([-r_{sec}, r_{sec}]^n)$
$\bar{m}$	Number of random polynomials generating $\mathbf{A}' \in \mathcal{R}_q^m$	e.g. $\bar{m} = 2$
For $\mathbf{m = 3}$ (Section 5.4)		
Message size	$3 \cdot n \log_2 p_1 + l \cdot n \log_2 p_2$	
Ciphertext size	$(3 + l) n \log_2(q)$	
Public key size	$\mu + n \log_2(q)$	
Secret key size	$2n(1 + \lceil \log_2(r_{sec}) \rceil)$	

Table 3. Parameters

8.1 Optimizations

Polynomial Representation The polynomial representation mainly follows the work [26], which is optimized for $n = 512$. In particular, polynomials are stored in an array of 512 double-precision floating point values. Using the single instruction multiple-data (SIMD) instructions of AVX allows to operate on vectors of 4 double precision floating points in the 256-bit ymm registers such that 4 double-precision multiplications and 4 additions of polynomial coefficients are performed within each cycle via the instructions vmultpd resp. vaddpd. In fact, only 64 polynomial coefficients fit into the available 16 ymm registers.

Polynomial Multiplication and NTT/FFT For polynomial multiplication we use the NTT transformation, which exists due to the choice of primes satisfying $q \equiv 1 \bmod 2n$ for $n = 512$ as already discussed in Section 5.1. The NTT benefits from the fact, that the root of unity is an element of $\mathbb{Z}_q$, thereby avoiding to operate with complex numbers required for the standard FFT. We also provide an efficient self-made implementation of the FFT by use of AVX/AVX2, which is almost as efficient as the NTT. In fact, the NTT performance factor is as low as 1.7 in comparison to the FFT. The FFT transform is used for instantiations with non-prime moduli such as $q = p^k$, where the NTT is not applicable. Similar to [26], we precompute tables of the relevant constants prior to protocol execution.

Tag Generation As for generating the tag in the setting $q = 1 \bmod 2n$, we sample a uniform seed and expand it to the desired length by use of a PRNG in order to generate the coefficients of its NTT representation $\mathsf{NTT}_\xi(T(\mathbf{u}))$ as described in Section 5.5. We note that it will never be transformed back to the polynomial representation $\mathbf{u}$, hence saving one transformation. Inverting $\mathsf{NTT}_\xi(T(\mathbf{u}))$ in the decryption step is performed componentwise over $\mathbb{Z}_q$ rather than over $\mathcal{R}_q$. This leads to a remarkable speed up.

Storing in the NTT representation Due to the existence of the NTT with $\xi \in \mathbb{Z}_q$, we can store the whole public key $\mathbf{A}$ in its NTT representation without increasing the storage requirements. This even leads to a faster encryption and decryption engine, since one NTT forward transformation is saved regarding multiplications with the public key. The way tags are generated as indicated in Section 5.5 is perfectly tailored to this setting.

Sampling Discrete Gaussians The error term and potentially also the secret of A-LWE and LWE instances are sampled according to the discrete Gaussian distribution. In our implementation we use the FastCDT sampler, which outperforms current state-of-the-art samplers. This is discussed in Section 7.

High Data Load Encryption In order to allow for high data load encryption, the number of polynomials $l > 0$ in $\mathbf{A}'' \in \mathcal{R}_q^l$ have to be non-zero. These polynomials (and 2 polynomials in $\mathbf{A}'$) are completely uniform and can hence be generated from a random seed (or just the respective NTT representations). Therefore, it suffices to store only $\mathbf{A}' \in \mathcal{R}_q^3$ and a seed for $\mathbf{A}''$. But it is also possible to store $\mathbf{A}''$ in its NTT representation allowing for fast operations. One observes that increasing l does not decrease the bit security, since the optimal dimension has already been exceeded by $\mathbf{A}'$. Furthermore, the error size associated to $\mathbf{A}''$ is larger than the error size related to $\mathbf{A}'$.

Generation of Random Polynomials Seeds are produced by means of the Linux kernel random-generator /dev/urandom. We instantiate the random oracle $H(\cdot)$, when encrypting messages, by an efficient pseudo-random generator such as Salsa20 or ChaCha20. This allows to produce as many random bits as required.

8.2 Performance Analysis and Implementation Results

We implemented both our CPA/CCA secure schemes and the CPA-secure construction LP11 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, msse2avx and march=core-avx-2.

Scheme $m = 3$	**PK Size** (# polynomials)	**Random Poly** (# polynomials from one seed)	**SK Size** (# polynomials)
LARA			
$\mathbf{l = 0}$	1	2	1
$\mathbf{l = 2} \cdot m$	1	8	1
$\mathbf{l = 5} \cdot m$	1	17	1
$\mathbf{l = 10} \cdot m$	1	29	1
LP11 **[33]**	1	1	1

Table 4. Comparison of key sizes in terms of number of polynomials

In Table 4 we provide an overview of the key sizes. We compare the CPA secure construction LP11 with our CPA/CCA secure construction from Section 5.6 using the efficient trapdoor candidate from Section 5.4. We observe that the key sizes are identical in terms of number of polynomials. We need only to store one secret key and retrieve the other one from the public key. In our setting we require only one additional uniform random polynomial for the standard instantiation $l = 0$. In practice, however, all uniform random polynomials can be generated just from a seed such that the effective sizes are equal. We note that other CCA-secure constructions are less efficient and thus not included in the tables below.

In terms of performance, the most time critical operations are polynomial multiplications, which take about 14922 cycles including three NTT transformations consuming the major block of the running time. Our performance results are given in Table 5, Table 6 and Table 7. In particular, Table 5 contains the implementation results for our CPA-, CCA1- and CCA2-secure schemes presented in Section 5.6 at a security level of 128 bits. Table 7 and Table 6 depict the results of the NTT and FFT implementations at a security level of approximately $90 - 100$ bits by use of the LWE estimator[1]. We provide timings, bit security, message sizes and message expansion factors (ciphertext size/message size ratio) depending on different parameters defined in Table 3.

(FFT)	Parameters			Sizes (bits)		Timings (cycles/bit)		Security (bits)
	r_{sec}	p_1	p_2	Message	Message Exp.	Encrypt	Decrypt	[4]
LARA$_{\text{CPA}}$ $(\mathbf{q = 2^{20}})$								
$\mathbf{l = 0}$	23	2^7	-	10752	2.8	21.7 [0.09 ms]	14.8 [0.06 ms]	128
$\mathbf{l = 2} \cdot m$	23	2^7	2^{11}	44544	2.0	16.0 [0.29 ms]	7.2 [0.13 ms]	128
$\mathbf{l = 10} \cdot m$	23	2^7	2^{11}	179712	1.8	14.0 [1.05 ms]	5.3 [0.40 ms]	128
LARA$_{\text{CCA1}}$ $(\mathbf{q = 3^{12}})$								
$\mathbf{l = 0}$	23	2^6	-	9216	3.1	29.1 [0.11 ms]	22.2 [0.08 ms]	128
$\mathbf{l = 2} \cdot m$	23	2^6	2^{11}	43008	2.0	17.0 [0.30 ms]	8.8 [0.15 ms]	128
$\mathbf{l = 10} \cdot m$	23	2^6	2^{11}	178176	1.8	14.7 [1.09 ms]	6.1 [0.45 ms]	128
LARA$_{\text{CCA2}}$ $(\mathbf{q = 3^{12}})$								
$\mathbf{l = 0}$	23	2^6	-	9216	3.1	50.1 [0.19 ms]	42.0 [0.16 ms]	128
$\mathbf{l = 2} \cdot m$	23	2^6	2^{11}	43008	2.0	22.0 [0.39 ms]	13.0 [0.23 ms]	128
$\mathbf{l = 10} \cdot m$	23	2^6	2^{11}	178176	1.8	16.8 [1.23 ms]	7.4 [0.54 ms]	128
LP11 **[33]** $(\mathbf{q = 2^{20}})$	23	-	-	512	46	444 [0.09 ms]	65 [0.01 ms]	128

Table 5. Comparison of our CPA-, CCA1-, and CCA2-secure scheme with LP [33] using FFT

At the first glance, one observes for $l = 0$ that the message expansion factor of LP11 is larger than in our setting by a factor of $2 \cdot \log p_1$. More specifically, LP11 generates ciphertexts of size $2n \log q$ bits for n message bits, meaning that in average a half bit is encrypted per entry, whereas in our scheme we encrypt data of $\log p_1 = \log(\alpha q/4.7)$ bits per entry at a ciphertext size

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

of $3n \log q$ bits. This results in message expansion factors of approximately 2 in case we encrypt 11 bits per entry (of size 23 bits) as compared to a factor of 46 for LP11. Our CPA-secure scheme (see Table 6) consisting of 3 ciphertext polynomials ($l = 0$) requires 255272 and 155328 cycles to encrypt and decrypt 16896 bits of data or equivalently 11 bits per entry (i.e. 15.1 resp. 9.1 cycles per bit), whereas LP11 consisting of 2 polynomials encrypts only 512 bits of data within 204093 cycles, i.e. 398.6 cycles per message bit. However, the fast decryption procedure of LP11 requires only 36383 cycles. By use of the NTT approximately 151771 and 149735 cycles are required in order to encrypt and decrypt 7680 bits of data, whereas LP11 accomplishes the same operations for 512 data bits within 150269 and 11217 cycles (see Table 7). This corresponds to 19.5 and 19.4 cycles per bit for encryption and decryption as opposed to 293.1 and 21.9 cycles per bit in case of LP11. Increasing the parameter l strongly reduces the number of cycles per bit due to two reasons. First, similar to standard LWE the same secret vector can be used to generate a large number of A-LWE instances, thus increasing the number of error polynomials containing data. Second, since the error size associated to $\hat{\mathbf{e}}_2$ is not crucial for decryption, we can make it arbitrary large and pack a large portion of the data into this part. This approach improves the efficiency and flexibility of the scheme as it is not needed to invoke the encryption engine several times. For instance, $\hat{\mathbf{e}}_2$ can play the role of a signature on the data embedded into $\hat{\mathbf{e}}_1$. We note that it is also possible to use a uniform random error term, which would allow us to use the full bandwidth.

$\mathbf{q = 2^{23}}$	**Parameters**			**Sizes (bits)**		**Timings (cycles/bit)**		**Security (bits)**
(FFT)	$\mathbf{r_{sec}}$	$\mathbf{p_1}$	$\mathbf{p_2}$	**Message**	**Message Exp.**	**Encrypt**	**Decrypt**	**[4]**
LARA$_{\mathsf{CPA}}$								
$\mathbf{l = 0}$	10	2^{11}	-	16896	2.1	15.1 [0.10 ms]	9.1 [0.06 ms]	≈ 90
$\mathbf{l = 2 \cdot} m$	10	2^{11}	2^{12}	53760	2.0	13.5 [0.30 ms]	5.7 [0.12 ms]	≈ 90
$\mathbf{l = 10 \cdot} m$	10	2^{11}	2^{12}	201216	1.9	13.1 [1.10 ms]	4.7 [0.39 ms]	≈ 90
LP11 **[33]**	10	–	–	512	46	398.6 [0.08 ms]	71.0 [0.01 ms]	≈ 95

Table 6. Comparison of our CPA-secure scheme with LP [33] using $q = 2^k$ (FFT)

$\mathbf{q = 8383489}$	**Parameters**			**Sizes (bits)**		**Timings (cycles/bit)**		**Security (bits)**
(NTT)	$\mathbf{r_{sec}}$	$\mathbf{p_1}$	$\mathbf{p_2}$	**Message**	**Message Exp.**	**Encrypt**	**Decrypt**	**[4]**
LARA$_{\mathsf{CPA}}$								
$\mathbf{l = 0}$	20	2^{5}	-	7680	4.6	19.5 [0.06 ms]	19.4 [0.06 ms]	≈ 100
$\mathbf{l = 2 \cdot} m$	20	2^{5}	2^{12}	44544	2.3	10.7 [0.20 ms]	4.3 [0.08 ms]	≈ 100
$\mathbf{l = 10 \cdot} m$	20	2^{5}	2^{12}	192000	2.0	9.5 [0.76 ms]	2.1 [0.17 ms]	≈ 100
LP11 **[33]**	20	–		512	46	293.4 [0.06 ms]	21.9 [0.004 ms]	≈ 100

Table 7. Comparison of our CPA-secure scheme with LP [33] for q prime (NTT)

In fact, when comparing the absolute timings of both schemes, we observe that the running time of LARA$_{\mathsf{CPA}}$, LARA$_{\mathsf{CCA}}$ and LP11 are in the same order or even equal. However, the decryption routine of LP11 is faster than in our setting. In terms of cycles per message bits, LARA realizes improvement factors up to 26.3 and 7.8 for encryption and decryption. Even larger factors are achieved in the high data load encryption mode for $l > 0$, because we can extend the public key by random polynomials and encrypt messages using the same secret in its NTT/FFT transformation. The overhead is solely restricted to generating new error polynomials and multiplying the secret with the random polynomials from $\mathbf{A}''$. Indeed, beside of the theoretical evidence our implementation also confirms that the discrete Gaussian sampler FastCDT allows to sample discrete Gaussians with a large error size almost as efficient as with a small one. Much better improvement factors are obtained, if l is increased and the error polynomials associated to $\mathbf{A}''$ are sampled from a wider discrete Gaussian distribution $\beta q = p_2 \cdot 4.7$ and hence encrypt more bits per entry. For decryption no length conditions are imposed on the error polynomials related to $\mathbf{A}''$.

Bibliography

[1] 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.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996.

[3] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, March 2009.

[4] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In *ICISC 2013*, 2013.

[5] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[6] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: A proposal for the SHA-3 standard, 2008. In The First SHA-3 Candidate Conference.

[7] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[8] Shi Bai and StevenD. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, LNCS, pages 28–47. Springer, 2014.

[9] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. Cryptology ePrint Archive, Report 2015/483, 2015.

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

[11] 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.

[12] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16. Springer, March 2011.

[13] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In PhongQ. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, LNCS. Springer, 2014.

[14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011.

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

[16] Dana Dachman-Soled, Rosario Gennaro, Hugo Krawczyk, and Tal Malkin. Computational extractors and pseudorandomness. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 383–403. Springer, March 2012.

[17] Léo Ducas and Daniele Micciancio. *CRYPTO 2014*, chapter Improved Short Lattice Signatures in the Standard Model, pages 335–352. Springer, 2014.

[18] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and JuanA. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer Berlin Heidelberg, 2013.

[19] NagarjunC. Dwarakanath and StevenD. Galbraith. Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing*, 25(3):159–180, 2014.

[20] 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.

[21] 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`.

[22] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, April 2008.

[23] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[24] 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.

[25] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and JuanA. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCSe*, pages 75–92. Springer, 2013.

[26] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, volume 7932 of *LNCS*. Springer, 2013.

[27] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, August 1997.

[28] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, September 2012.

[29] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, JosephH. Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. SPRINGER, 2003.

[30] Xiaodong Lin Jintai Ding. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. `http://eprint.iacr.org/`.

[31] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 636–652. Springer, December 2009.

[32] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA 2000*, pages 937–941. ACM, 2000.

[33] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011.

[34] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, February 2011.

[35] San Ling, DuongHieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-lwe and applications in traitor tracing. In JuanA. Garay and Rosario Gennaro, editors, *CRYPTO 2014*, volume 8616 of *LNCS*. Springer, 2014.

[36] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, April 2012.

[37] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for FFT hashing. In *FSE*, volume 5086 of *LNCS*, pages 54–72. Springer, 2008.

[38] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, May 2010.

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

[40] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and PhongQ. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, 2013.

[41] 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.

[42] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages

372–381. IEEE Computer Society Press, October 2004.

[43] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*. Springer, 2009.

[44] 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.

[45] 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.

[46] 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.

[47] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, August 2008.

[48] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008.

[49] John M. Pollard. The Fast Fourier Transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.

[50] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. *Enhanced Lattice-Based Signatures on Reconfigurable Hardware*, pages 353–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[51] 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.

[52] 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.

[53] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, December 2009.

[54] Franz Winkler. *Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation)*. Springer, 1 edition, 1996.