High Performance Lattice-based CCA-secure Encryption

Rachid El Bansarkhani and Johannes Buchmann

Technische Universität Darmstadt
Fachbereich Informatik
Kryptographie und Computeralgebra,
Hochschulstraße 10, 64289 Darmstadt, Germany
`elbansarkhani@cdc.informatik.tu-darmstadt.de,buchmann@cdc.informatik.tu-darmstadt.de`

Abstract. Lattice-based encryption schemes still suffer from a low message throughput per ciphertext. This is mainly due to the fact that the underlying schemes do not tap the full potentials of LWE. Many constructions still follow the one-time-pad approach considering LWE instances as random vectors added to a message, most often encoded bit vectors. Recently, at Financial Crypto 2015 El Bansarkhani et al. proposed a novel encryption scheme based on the A-LWE assumption (Augmented LWE), 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 traditional one-time-pad approach and it is even possible to combine both concepts. In this paper we revisit this approach and propose amongst others several algebraic techniques in order to significantly improve the message throughput per ciphertext. Furthermore, we give a thorough security analysis as well as an efficient implementation of the CCA1-secure encryption scheme instantiated with the most efficient trapdoor construction. In particular, we attest that it even outperforms the CPA-secure encryption scheme from Lindner and Peikert presented at CT-RSA 2011.

Keywords: Lattice-Based Cryptography, Encryption Scheme, 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 [Ajt96] on worst-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 [GGH97] and NTRU-Sign [HHGP$^+$03]. One of the main contributions of his work is a proof that average-case instances of lattice problems enjoy worst-case hardness. In particular for cryptography, the average-case problems SIS and LWE form the foundation for almost all of the well-known lattice-based cryptosystems. LWE is often used for primitives from Cryptomania such as provably secure encryption schemes [LP11, SS11] including chosen-ciphertext secure encryption [PW08, GPV08, Pei09, ABB10, MP12], identity based encryption [GPV08, CHKP10, ABB10, MP12] and fully homomorhic encryption [BGG$^+$14, BV11, Gen09, GSW13]. Furthermore, it is applied for building secure key exchange protocols [KV09, JD12] and oblivious transfer [PVW08]. 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 [Reg05] 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 distribution with parameter $\alpha q \geq 2\sqrt{n}$. The SIS problem, however, usually serves to build provably secure signature schemes [BG14, DDLL13, GLP12, Lyu12, GPV08, MP12], preimage sampleable trapdoor functions [GPV08, MP12, AP09, Pei10, SS11] and collision-resistant hash functions [LMPR08, ADL$^+$08]. 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. [EDB15, LPSS14, BF11b]) 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 [EDB15] has been proposed that encrypts data in a way that differs from previous constructions following the standard 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 high security standards for many cryptographic applications, for instance needed to secure financial transactions. More specifically, EL Bansarkhani et al. [EDB15] introduce the Augmented Learning with Errors problem (A-LWE), a slightly modified LWE variant, 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 [MP12, EB14], the owner of the trapdoor is empowered to recover the secret resp. error-term and hence reveal the injected data. By this, the authours demonstrate that it is possible to use the error term as a container for the message or further information such as lattice-based signatures following the distributions of the error-term. Following this approach, one can realize RCCA-secure resp. CCA2 secure encryption algorithms that are not build upon the classical one-time-pad approach as before. Beside of its presumed efficiency due to the resemblance of ciphertexts to ordinary LWE samples, the scheme further allows to be combined with the one-time-pad concept, 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 [EDB15]. In particular, we provide several theoretical improvements and a software implementation of the scheme testifying its conjectured efficiency.

1. We introduce new techniques in order to increase the message throughput per ciphertext. In fact, we are able to use almost the full min-entropy of the error term to embed arbitrary messages. Previously in [EDB15], 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 [EDB15] according to Table 1. Here, for simplicity, the ciphertext size is fixed and the other parameters are subsequently derived from the ciphertext size. When comparing our approach with the CPA-secure encryption scheme from Lindner and Peikert [LP11], we attest an improvement factor of $O(\log(\alpha q))$.

$m = c \cdot nk$ $k = \log q$	CCA1 **[MP12]**	CCA1 **[EDB15]**	CCA1 **This work**	CCA1 **This work** + **[MP12]**	CPA **[LP11]**
Ciphertext size	$m \cdot k$	$m \cdot k$	$m \cdot k$	$m \cdot k$	$m \cdot k$
Enc. signature size[1]	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

2. Furthermore, we introduce a new LWE inversion algorithm for arbitrary moduli and hence generalize the algorithm from Micciancio and Peikert [MP12] using a different approach. There exists many applications, where it is desirable to have a modulus of a specific shape such as prime modulus satisfying $q \equiv 1 \bmod 2n$ in the ring setting such that the fast NTT transformation can be used. The inversion algorithm given in [MP12] is only suitable for moduli of the form $q = 2^k$. For moduli different to this shape, the inversion algorithm fails to recover the secret in LWE instances. Our algorithm, however, can efficiently be applied to LWE instances involving an arbitrary modulus.

3. 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 $q \equiv 1 \bmod 2n$. The inversion algorithm can check arbitrary ring elements with regard to invertibility by one NTT transformation. Inverting elements in $\mathcal{R}_q^\times$ is reduced to inversion in $\mathbb{Z}_q^\times$, which is cyclic due to prime modulus. When working in the NTT representation, we are even not required to apply the NTT forward and backward transformations. This will be particularly important for the tagging approach, where a tag is chosen uniformly at random from the ring of units and is subsequently applied to the public key when encrypting messages.

4. We instantiated the CCA1-secure encryption scheme in the ring setting for public keys being both statistically and computationally indistinguishable from uniform. This mainly follows from the trapdoor construction due to [MP12, EB14]. Moreover, we considered different parameter sets (e.g. error and secret key parameter) for $n = 512$ varying the message throughput per ciphertext. The authours of [EDB15] also proposed a high data load encryption mode that additionally allows to encrypt large blocks of data using the same secret and at a minimal increase of the running time. 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 [AFG13] and [BG14] suitable for a bounded number of LWE samples. We differentiate key recovery attacks from attacks against the ciphertext.

5. In order to attest the conjectured efficiency of the scheme, we implemented the scheme in software for $n = 512$. This implementation is optimized with respect to the underlying architecture using AVX resp. AVX2. For this, we essentially applied the techniques from [GOPS13], which provides several optimizations for the polynomial representation and polynomial multiplication applying the NTT. Using the same platform and optimizations we compared the CCA1-secure scheme with the current state-of-the-art CPA-secure scheme by Lindner and Peikert [LP11, GFS+12]. In fact, our implementations testify that our encryption engine is faster by a factor between 10 and 24 as compared to [LP11] for the considered parameters, whereas our decryption routine is slower in the normal mode $l = 0$ and faster for certain parameters in the high data load encryption mode $l > 0$ (i.e. improvement factors range between 0.2 and 3.3).

1.2 Organization

This paper is structured as follows. In Section 2 we provide the relevant backround of our work. In Section 3 we introduce the A-LWE problem from [EDB15] and present our improvements and techniques. Moreover, we give a description of the resulting CCA1-secure encryption scheme involving our modifications. Subsequently in Section 4, we provide a security analysis of the scheme followed by an efficient implementation in software. Finally, in Section 5 we present the experimental results while comparing with a software implementation of the current state-of-the-art CPA-secure encryption scheme from Lindner and Peikert.

2 Preliminaries

Notation 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).

We will use the 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 prime integers $q > 0$ and n being a power of two. We denote ring elements by boldface lower-case letters e.g. $\mathbf{p}$, whereas for vectors of ring elements we use $\hat{\mathbf{p}}$ and upper-case bold letters for matrices (e.g., $\mathbf{A}$). By $\oplus$ we denote the XOR operator.

Discrete Gaussian Distribution We define by $\rho : \mathbb{R}^n \to (0,1]$ the n-dimensional Gaussian function $\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi \cdot \frac{\|\mathbf{x}-\mathbf{c}\|_2^2}{s^2}}$, $\forall \mathbf{x}, \mathbf{c} \in \mathbb{R}^n$. The discrete Gaussian distribution $\mathcal{D}_{\Lambda+\mathbf{c},s}$ is defined to have support $\Lambda + \mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^n$ and $\Lambda \subset \mathbb{R}^n$ is a lattice. For $\mathbf{x} \in \Lambda + c$, it basically assigns the probability $\mathcal{D}_{\Lambda+\mathbf{c},s}(\mathbf{x}) = \rho_s(\mathbf{x})/\rho_s(\Lambda + c)$.

Lattices. 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 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. ***([GPV08, Theorem 3.1]).*** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. We have*

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

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, [?]). *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $k = \lceil \log q \rceil \geq 1$ with $m = l \cdot k$ be an arbitrary full-rank matrix. The statistical distance $\Delta(\mathcal{D}_{\mathbb{Z}^m,r}, \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{A}),r})$ for uniform $\mathbf{v} \leftarrow_R \mathbb{Z}_q^l$ and $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$ is negligible.*

Lemma 3 (computational,[?]). *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $k = \lceil \log q \rceil \geq 1$ with $m = l \cdot k$ be an arbitrary full-rank matrix. If the distribution of $\mathbf{v} \in \mathbb{Z}_q^l$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^l$, then $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{A}),r}$ is computationally indistinguishable from $\mathcal{D}_{\mathbb{Z}^m,r}$ for $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$.*

Lemma 4 (Lemma 2.4, [Ban95]). *For any real $s > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have*

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

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

Definition 2 (LWE Distribution). *Let n, m, q be integers and χ_e be distribution over $\mathbb{Z}$. By $L_{n,m,\alpha q}^{\mathsf{LWE}}$ we denote the LWE distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$, which drwas $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ 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 $(\mathbf{A}, \mathbf{b}^\top) \leftarrow L_{n,m,\alpha q}^{\mathsf{LWE}}$ 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 $(\mathbf{A}, \mathbf{b}) \leftarrow L_{n,m,\alpha q}^{\mathsf{LWE}}$ 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 proposed in [EDB15] and how it is used in order to inject auxiliary data into the error term of LWE samples. This feature represents the main building block of the generic encryption scheme from [EDB15], which allows to encrypt huge amounts of data without increasing the ciphertext size as compared to previous proposals. 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 costs.

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 $\mathbf{c} \in \mathbb{Z}_p^n$ 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 vector from $\Lambda_{\mathbf{c}}^{\perp}(\mathbf{B}) = \mathbf{c} + \Lambda_p^{\perp}(\mathbf{B})$. However, this requires the knowledge of a suitable basis for $\Lambda_p^{\perp}(\mathbf{B})$. 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 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 authours introduce the Augmented LWE problem (A-LWE), where A-LWE samples resemble ordinary LWE samples 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. But other choices are also possible as long as the parameter of the error vectors exceed the smoothing parameter of the used matrix. We now give a generalized description of the A-LWE distribution using any 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 function. Let $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable trapdoor function (such as $\mathbf{B} = \mathbf{G}$ from [EDB15]). 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 $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),\alpha q}$.*
- *Return $(\mathbf{A}, \mathbf{b}^\top)$ where $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.*

In principal, for A-LWE one differentiates the decision problem $\mathsf{decision\ A\text{-}LWE}^H_{n,m_1,m_2,\alpha q}$ from the corresponding search problem $\mathsf{search\text{-}s\ A\text{-}LWE}^H_{n,m_1,m_2,\alpha q}$, as known from LWE. Furthermore, there exists a second search problem $\mathsf{search\text{-}m\ A\text{-}LWE}^H_{n,m_1,m_2,\alpha q}$, where a challenger is asked upon polynomially many A-LWE samples to find the message $\mathbf{m}$ injected into the error vector in polynomial time. Note that the error distribution could also differ from the discrete Gaussian distribution. For instance, one could use the uniform distribution, for which one obtains similar results.

All the proofs from [EDB15] go through without any modification, since the security proofs do not depend on the choice of $\mathbf{B}$ at all. In fact, the proofs require only that $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),\alpha q}$ is indistinguishable from $\mathcal{D}_{\mathbb{Z}^m,\alpha q}$ for a random vector $\mathbf{v}$. But this is true as per Lemma 3 resp. Lemma 2 for $\alpha q > \eta_\epsilon(\Lambda^{\perp}(\mathbf{B}))$ and $\epsilon = \mathsf{negl}(\lambda)$. Hence, we can directly use all the schemes provided in [EDB15] such that we get the following adapted variant of the LWE to A-LWE reduction.

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

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

We note, that these hardness results can directly be translated to the corresponding ring variants.

3.2 Improved Message Embedding

In the following section we propose several techniques in order to enhance the message throughput per ciphertext. These techniques could also be applied to the error vector in the A-LWE distribution. In other words, we aim at choosing an appropriate preimage sampleable matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ such that n' is large enough. For now, we will focus on how to apply this technique to the different encryption schemes and omit the $\mathbf{e}_1$ term when invoking the random oracle, since the secret $\mathbf{s} \in \mathbb{Z}_q^n$ is always resampled and hence provides enough entropy for each fresh encryption query. Our first approach is based on a method used to construct homomorphic signatures in [BF11a]. The second approach does not require such complex procedures in order to allow for the same message throughput.

Intersection Method The intersection method as proposed in [BF11a] 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 elementwise. Therefore, let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two messages, where $\mathbf{m}_1$ resp. $\mathbf{m}_2$ define a coset of Λ_1 resp. Λ_2 in $\mathbb{Z}^m$. 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 $\mathbf{x} \to (\mathbf{x} \bmod p)$ and $\mathbf{x} \to (\mathbf{A} \cdot \mathbf{x} \bmod p)$. 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 $\mathbf{G} \in \mathbb{Z}_q^{m/km}$ from [EDB15] for $k = \log q$,we have $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n})$. In our scheme, however, we have to sample a short vector $\mathbf{e}$ from $(H(\mathbf{s}) \oplus \mathbf{t}) + \Lambda_1 \cap \Lambda_2$ with parameter $\alpha q \geq \eta_\epsilon(\Lambda_1 \cap \Lambda_2)$, where $\mathbf{t}$ is computed as above and the (simplified) description $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^m$ defines a random oracle taking a random secret $\mathbf{s} \in \mathbb{Z}_q^n$ as input. The error vector is then given by $\mathbf{e} \leftarrow D_{\mathbf{b} + \Lambda_1 \cap \Lambda_2, \alpha q}$ with $\mathbf{b} = H(\mathbf{s}) \oplus \mathbf{t}$. Due to $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n}) \leq \alpha q$ (e.g. $\alpha q = 2\sqrt{n}$), 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 as compared to m in [EDB15].

Lattices of the Form $\mathbf{p} \cdot \mathbb{Z}^{\mathbf{m}}$ One realizes that for a given parameter αq of the error vector one can be much more efficient, if one considers only the lattice $\Lambda = p\mathbb{Z}^m$. In this case, the message space is simply defined by the set $\mathbf{m} \in \mathbb{Z}^m / \Lambda \cong \mathbb{Z}_p^m$. 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 $m \log 2p = m(\log p + 1)$. Furthermore the decoding and encoding phase is much faster, since encoding requires only to sample $\mathbf{e} \leftarrow D_{\mathbf{b} + p\mathbb{Z}^m, \alpha q}$ for $\mathbf{b} = H(\mathbf{s}) \oplus \mathbf{m}$ using fast discrete Gaussian samplers such as the Knuth-Yao algorithm and decoding is performed via $H(\mathbf{s}) \oplus (\mathbf{e} \bmod p)$. Optimizing the message throughput requires to increase p such that $\eta_\epsilon(\Lambda) \leq p \cdot \omega(\sqrt{\log n}) \leq \alpha q$ still holds. Doing this, one can embed approximately

$m\log(\alpha q/\omega(\sqrt{\log n}))$ bits of data, which almost coincides with the min-entropy of a discrete Gaussian with parameter αq. Therefore, it is most effective to choose a parameter such that $\alpha q = p\cdot\omega(\sqrt{\log n})$ with $p = 2^i$ for some $i > 0$ in order to embed i bits of data into the error term.

Uniform Error For uniformly distributed errors one can directly use the output of $H(\cdot)$ 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 cryptographic hash function modeled as random oracle such that $\mathbf{e} \leftarrow H(\mathbf{s}) \oplus \mathbf{m}$ 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.

3.3 Setting

Prior to starting with a description of our implementation we define our setting, in which we operate and explain how this affects the performance and usability of the scheme. In particular, 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 of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ 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 [Pol71] and furthermore provide similar worst-case to average-case hardness results [LPR10]. Though it seems to be preferable to operate with a modulus of the form $q = 2^l$, when considering the trapdoor construction and the corresponding LWE inversion algorithm from [MP12, EB14], but it might be more advantageous 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. 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 [GOPS13] in order to efficiently perform polynomial multiplication as already observed in [GOPS13]. 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 [Win96] and there is no need to use the less efficient FFT on the complex numbers. Moreover, one realizes with view to the above representation of $\mathcal{R}_q$ that the number of invertible elements 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, when choosing $q = 8383489$ and $n = 512$ this ratio is approximately 1. This fact helps us to choose better parameters for generating trapdoor. Beyond that, we propose a fast technique to generate ring elements and the corresponding inverses required for tagging the public key in order to ensure CCA security. This method is mainly possible due to the existence of the NTT. In addition, we present in Section 3.7 an LWE inversion algorithm for arbitrarily composed moduli q including prime numbers, since the algorithm given in [MP12] can only be successfully applied if q is a power of two.

3.4 Trapdoors

In this paragraph we shortly introduce the trapdoor generation algorithm [MP12, EB14], which is used in order to construct a public key that is indistinguishable from uniform and allows to invert LWE instances. We will restrict to the ring variant following [EB14] with polynomials in $\mathcal{R}_q$. Thus, let $k = \lceil \log q \rceil$ and $\bar{m} > 0$ be an integer.
$\mathsf{TrapGen}(1^n)$:

- **Statistical instantiation.** Sample $\bar{m}$ polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in R_q^{\bar{m}}$ 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

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

– **Computational instantiation.** 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 $r_{i,j}$ 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 public key is then computed as follows

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

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 $1 \leq i \leq k$.

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 the entries of the trapdoor polynomials $\mathbf{r}_{i,j}$ to be of small size. Conversely, a small number of random polynomials leads to larger values. In fact, one can apply the regularity bound from [SSTX09, 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 $[-b, \ldots, b]$ 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 for $\bar{m}$. In this case, however, a better regularity bound is given by [SS11, Lemma 3.1] resp. [LPR13, Corollary 7.5]. For inverting LWE instances it is also important that the parameters are chosen to be small enough in order to efficiently recover the secret. The computational instantiation requires only one single polynomial $\mathbf{a}_1$ sampled uniformly at random. The other polynomials are sampled from the LWE distribution using $\mathbf{a}_1$. This approach, however, requires the secret when encrypting a message to be distributed from the discrete Gaussian distribution in order to correctly recover the secret.

3.5 CCA secure Encryption Scheme from A-LWE

By use of our improved message embedding technique, we provide a description of the modified CCA1 secure encryption scheme from [EDB15] adapted to the ring setting. The above observation that techniques for larger message throughput from Section 3.2 have no impact on the distributions, CCA security directly follows from [EDB15]. Thus, let $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$ for n a power of two and $\alpha q \geq p \cdot \omega(\sqrt{\log n})$ according to Section 3.2. Set $k = \lceil \log q \rceil$ and denote by $\mathsf{TDF} = (\mathsf{KeyGen}, g, g^{-1})$ a trapdoor function with $g_{\mathbf{A}}(\mathbf{x}, \hat{\mathbf{y}}) := \mathbf{A}\mathbf{x} + \hat{\mathbf{y}} \in \mathcal{R}_q^t$ for $t = l + m$.

$\mathsf{KGen}(1^n)$: Generate public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathcal{R}_q^{l+m}$, where $\mathbf{A}'' \in \mathcal{R}_q^m$ is instantiated statistically or computationally according to Section 3.4 with trapdoor $\mathsf{sk} := \hat{\mathbf{r}}$.

- Without high data load encryption $l = 0$.
- With high data load encryption following [EDB15], then $l > 0$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{msg} \in \{0,1\}^c,$ for $c = (m+l)n \log p)$: Sample a tag $\mathbf{t}_u \in \mathcal{R}_q^\times$ uniformly at random and generate $\mathbf{A}''_\mathbf{u}$ following Section 3.4. Then, sample $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$ (resp. $\mathbf{s} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\alpha q}$) and compute $\hat{\mathbf{v}} = (\mathbf{v}_1, \ldots, \mathbf{v}_{m+l}) = \mathsf{encode}(H(\mathbf{s}) \oplus \mathsf{msg}) \in \mathcal{R}_q^{m+l}$, where $H : \{0,1\}^* \to \{0,1\}^c$ is a cryptographic hash function modeled as random oracle. Finally, sample $\mathbf{e}_i \leftarrow_R \mathcal{D}_{\Lambda^\perp_{\mathbf{v}_i}(p\cdot\mathbf{I}),\alpha q} = \mathcal{D}_{p\mathbb{Z}^n+\mathbf{v}_i,\alpha q}$ for $1 \leq i \leq m+l$, where $\mathbf{v}_i$ is viewed as a vector from $\mathbb{Z}^n$ rather than a polynomial when added to $p\mathbb{Z}^n$. The ciphertext is then given by $\hat{\mathbf{c}} = g_\mathbf{A}(\mathbf{s}, \hat{\mathbf{e}})$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Compute $g_\mathbf{A}^{-1}(\hat{\mathbf{r}}, \mathbf{c}) = (\mathbf{s}, \mathbf{e})$ for $\mathbf{A}_\mathbf{u} = [\mathbf{A}' \mid \mathbf{A}''_\mathbf{u}] \in \mathcal{R}_q^{l+m}$,as follows:

1. **Statistical instantiation** Let $\mathbf{c}' = (\mathbf{c}_{l+1}, \ldots, \mathbf{c}_{l+\bar{m}})$. Then compute $\tilde{\mathbf{c}}_{l+\bar{m}+i} = \mathbf{c}_{l+\bar{m}+i} - h_{\mathbf{c}'}(\hat{\mathbf{r}}_i)$ for $1 \leq i \leq k$ and $m = \bar{m} + k$. Now, invoke the LWE inversion algorithm from Section 3.7 on $\tilde{\mathbf{c}}_{l+\bar{m}+1}, \ldots \tilde{\mathbf{c}}_{l+\bar{m}+k}$ in order to recover $\tilde{\mathbf{s}} = \mathbf{t}_u \cdot \mathbf{s}$. Subsequently, compute $\hat{\mathbf{e}} = \hat{\mathbf{c}} - \mathbf{A}_u\mathbf{s}$ for $\mathbf{s} = \mathbf{t}_u^{-1}\tilde{\mathbf{s}}$.
2. **Computational instantiation** Compute $\tilde{\mathbf{c}}_{l+i+1} = \mathbf{c}_{l+i+1} - \mathbf{c}_{l+1} \cdot \mathbf{r}_{i,1}$ for $1 \leq i \leq k$. The remaining steps are identical to the statistical instantiation.

Return $\mathsf{msg} = (\hat{\mathbf{e}} \bmod p) \oplus H(\mathbf{s})$.

We note here, that the secret vector $\mathbf{s}$ can also be exploited for additional messages and together with the one-time-pad approach from [MP12] it allows to embed further $2nk - n$ bits (e.g. $\mathbf{s} = (H'(\mu), \mu) \in \mathbb{Z}_q^n$ with μ containing enough bits). The mode for high data load encryption (i.e. $l > 0$) allows to encrypt huge amounts of data at a minimal increase of the running time. The corresponding polynomials (resp. matrix part) $\mathbf{A}'$ can efficiently be generated from a random seed. And also the security of the scheme does not change. This is particularly interesting for secure backups or traffic over the internet. In the following, we recall an adapted security statement of the CCA1-secure encryption scheme from [EDB15].

Theorem 2. ***(Adapted [EDB15, Theorem 4])*** *Let p be an integer such that $\alpha q = p \cdot \omega(\sqrt{\log n})$. Then, the encryption scheme above is CCA1-secure assuming the hardness of* $\mathsf{decision\ A\text{-}LWE}_{n,0,m,\alpha q}$.

The proof directly follows from [EDB15].

3.6 Fast Tag Generation and Inversion

Ensuring CCA resp. RCCA security requires to apply the tagging approach as realized in [MP12]. It is needed to generate a tag from a large set $\mathcal{U}$. We propose a technique which allows to generate tags and its corresponding inverses much faster than with polynomial division in combination with the extended Euclidean algorithm. Furthermore, it allows to efficiently check whether an element belongs to the ring of invertible elements $\mathcal{R}_q^\times$. Conceptually, it is based 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 $\mathbf{A} = (\xi^{ij})_{1 \leq i,j \leq n}$ and an element $\xi \in \mathbb{Z}_q$ of order n [Win96]. Multiplication of two polynomials is performed componentwise after applying the NTT transform on each polynomial.

Theorem 3. *Let q be a prime and n be a power of two with $q \equiv 1 \bmod 2n$. 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 $T(\mathbf{f}) = (f_0, \psi f_1, \ldots, \psi^{n-1} f_{n-1})$ and NTT transformation matrix $\mathbf{A} = (\xi^{ij})_{1 \leq i,j \leq n}$, where $f_0, \ldots, f_{n-1}$ denote the coefficients of the polynomial $\mathbf{f}$ (coefficient embedding). The inverse of $\mathbf{f}$ is given by $\mathbf{f}^{-1} = \mathbf{g}$ for*

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

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. One can easily check 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

$1 \le i \le n$ 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 $\mathbb{Z}_q\backslash\{0\}$. As a result, one can easily check that an element is invertible whenever all $\tilde{f}_i \neq 0 \bmod q\ \forall i$. $\square$

From Theorem 3 it follows that the set $\mathcal{R}_q^{\times}$ contains $(q-1)^n$ elements such that for two random polynomials $g, f \in \mathcal{R}_q^{\times}$ the difference $g - f$ lies in $\mathcal{R}_q^{\times}$ with probability $(1 - 1/(q-1))^n$. More precisely, the difference $g - f$ 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, used only in the security proof, holds with overwhelming property. For practice, however, it suffices to set $q = 8383489$. Since multiplication always involves the NTT transform and tagging 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 way, since one NTT transformation is saved and it is, furthermore, very easy to generate invertible elements as it is only required to generate n random elements from $\mathbb{Z}_q\backslash\{0\}$. This does apparently not hold in case one desires to generate an invertible element directly from $\mathcal{R}_q$. Following this, inversion of $\tilde{\mathbf{f}}$ is also performed in the NTT state by generating the corresponding $\tilde{\mathbf{g}}$ according to Theorem 3. Such a strategy allows to compute inverses over $\mathbb{Z}_q\backslash\{0\}$ rather than $\mathcal{R}_q^{\times}$, which is much more efficient.

3.7 LWE Inversion For Arbitrary Modulus

In [MP12] an efficient LWE inversion algorithm has been proposed that works only for moduli q being a power of two. The second part of the LWE inversion algorithm fails whenever the modulus is of different type. Roughly speaking, the algorithm recovers the components of the secret $\mathbf{s}$ bitwise starting from the last bit. Shifting $\mathbf{s}$ to the left deletes the most significant bits $\bmod q$. However, if the modulus is not of this form, the secret is wrapped around and this approach does not work any more. In the following we give a description of our approach. The first part remains essentially the same. For the sake of simplicity, let $l = 0$ such that $\mathbf{A} = \mathbf{A}''$ 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 a ciphertext.

- **Step 1**: Set $\tilde{\mathbf{c}}_{\bar{m}+i} = \mathbf{c}_{\bar{m}+i} - \sum_{j=1}^{\bar{m}} \mathbf{c}_j \mathbf{r}_{ij} = \mathbf{g}_i\mathbf{s} + (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{r}_{ij}) = \mathbf{g}_i\mathbf{s} + \mathbf{p}_i$.
- **Step 2**: Let $\|\mathbf{p}\|_\infty \le b$ with overwhelming probability. When implementing the scheme, one can decrease the bound b, since the length of $\mathbf{r}_{ij}$ needs not to be upper bounded 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 as opposed to the least significant bits following [MP12].
  1. For $i = 0,\ i \le k$: 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 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 $c \pm b$ has bit representations $10000\ldots$ resp. 01111. Theoretically, both representations are possible due to the perturbation vector, which is upper bounded by b (e.g. $\log q - \log b \ge 6$). Therefore, one creates two instances each for a different representation and continues the algorithm with $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 10000$ resp. $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 01111$.

If the second case occurs at least once, the correct secret $\mathbf{s}$ is attained by checking whether the 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 all $1 \le i \le k$ after normalization of the entries of $\mathbf{e}'_i$ to the range $[-q/2, q/2]^n$, because otherwise there exists an i such that the normalized $\mathbf{e}'_i$ are not all contained in $[-4.7\cdot\alpha q, 4.7\cdot\alpha q]^n$ due to injectivity of $g_{\mathbf{A}}(\cdot,\cdot)$ for the chosen parameters.

The choice of a parameter for the error term can be derived with the help of the following lemma.

Lemma 5. *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*

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

Proof. Since $\mathbf{p}_i = (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j\mathbf{r}_{ij})$, we have $\|\mathbf{p}_i\| \le \|\mathbf{e}_i\| + \sqrt{\bar{m}}\,\|\mathbf{e}_j\|\,\|\mathbf{r}_{ij}\| \le q/4$. 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 $\|\mathbf{r}_{ij}\| \le r_{sec}\sqrt{n}$ and subsequently $\|\mathbf{e}_j\| \le \frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}$ 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 $\|\mathbf{e}_j\| \le \alpha q\sqrt{n}$. □

In order to estimate the bound b, which affects the running time, we can compute $\|\mathbf{r}_{ij}\|$ practically, since we have direct access to the key material. In any case, b is upper bounded by $b \le \|\mathbf{p}_i\|_\infty = 4.7\cdot\alpha q\,\|\hat{\mathbf{r}}_i\|_2 \le 4.7\cdot\alpha q\cdot r_{sec}\sqrt{\bar{m}n}$. This mainly follows from Lemma [Ban95, Lemma 2.4] with $\hat{\mathbf{r}}_i = (\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\bar{m}})$ viewed as a vector of $\mathbb{Z}_q^{n\bar{m}}$ rather than an element of $\mathcal{R}_q^{\bar{m}}$.

4 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 [BG14]. 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 [AFG13] with the variant that has recently been shown to be more efficient especially for a small number of samples [BG14]. In the following, we give a description of the main ingredients of the embedding approach for the matrix variant.

4.1 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 authours 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

$$\Lambda_q(\mathbf{A}_e) = \{\mathbf{u} \in \mathbb{Z}^n \mid \exists \mathbf{x} \text{ s.th. } \mathbf{A}_e\mathbf{x} \equiv \mathbf{0} \bmod q\}\,.$$

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 $s\sqrt{m}$. In [GN08] it was conjectured that a lattice basis reduction algorithm will find a shortest vector with high probability if

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

is satisfied for an algorithm characteristical Hermite-factor δ and a lattice resp. 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 play 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 by means of the embedding approach, 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. For instance, the authours of [LP11] proposed a model 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 allowing 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 $\mathbf{A} \in \mathcal{R}_q^t$

4.2 Analysis of Key Recovery Attacks

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.

- **Statistical Instantiation:** In order to recover the key we have to solve the ring-SIS problem for k instances $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ with $1 \leq i \leq k$. In this case, we have to find preimages $\hat{\mathbf{x}}_i$ such that $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)$ for $1 \leq i \leq k$ and $\|\mathbf{p}_i\| = \left\| \mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{x}_{ij} \right\| \leq \|\mathbf{e}_i\| + \|\mathbf{e}_j\| \|\mathbf{x}_{ij}\| \sqrt{\bar{m}} \leq q/4$ from Section 3.7. Suppose that the entries of $\|\hat{\mathbf{x}}_i\|_2 \leq b$ for $1 \leq j \leq \bar{m}$ and $1 \leq i \leq k$. Then, the Hermite factor required to solve the problem can be estimated by $\delta = (b/q^{n/m})^{1/m}$. This follows from [MR09], where the length of a shortest vector is estimated by $q^{n/m}\delta^m$ for $m = (\bar{m}+1) \cdot n$ and $\Lambda^{\perp}(\hat{\mathbf{c}}_i) = \{\hat{\mathbf{x}} \in \mathcal{R}_q^{\bar{m}+1} \mid \sum_{i=1}^{\bar{m}+1} \mathbf{c}_i \mathbf{x}_i \equiv \mathbf{0} \bmod q\}$, where $\hat{\mathbf{c}}_i = [\hat{\mathbf{a}}, h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)]$. By means of equation (3) one derives the bit security.

- **Computational Instantiation:** Here, the public key is composed by k LWE instances and a uniform random polynomial $\mathbf{a}_1$ used to generate the LWE samples. For each LWE 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 in order to estimate its security.

5 Software Implementation and Performance Analysis

At the inplementation front we considered several optimizations. As for the polynomial representation and optimizations with respect to the NTT we refer to the work [GOPS13], 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. We present a description of the main ingredients of our optimizations in Section 5.1 and continue in the following section with our performance analysis resp. implementation results.

5.1 Software Implementation and Optimization

Polynomial Representation The polynomial representation mainly follows the work [GOPS13], 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 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 For polynomial multiplication we use the NTT transformation, which exists due to the choice of $q \equiv 1 \bmod 2n$ for $n = 512$ as already discussed in Section 3.3. The NTT benefits from the fact, that the root of unity is an element of $\mathbb{Z}_q$ and hence avoids to operate with complex roots needed for the standard FFT. We also adopt the optimizations from [GOPS13] made to the NTT.

Tag Generation As for generating the tag, we use a seed in order to generate the coefficients of the NTT transformation of the tag $\mathsf{NTT}_\xi(T(\mathbf{u}))$ as described in Section 3.6. As a consequence, the corresponding polynomial will never be transformed back to $\mathbf{u}$, hence, saving one transformation. Inverting of $\mathsf{NTT}_\xi(T(\mathbf{u}))$ in the decryption step is performed componentwise over $\mathbb{Z}_q$, allowing to be much faster than over $\mathcal{R}_q$. Furthermore, the polynomials $\mathbf{g}_i$ are polynomials with constant term 2^i and zero coefficients elsewhere. Hence, multiplication of $\mathbf{u}$ with $\mathbf{g}_i$ requires only to multiply the components of $\mathsf{NTT}_\xi(T(\mathbf{u}))$ with 2^i rather than transforming $\mathbf{g}_i$ to its NTT representation.

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 application of the NTT on the public key prior to polynomial multiplication is not needed. This saves one transformation in each step. The way of generating tags as already shown above is perfectly adapted to this case.

Sampling Discrete Gaussians The error term and potentially also the secret of A-LWE resp. LWE instances are sampled according to the discrete Gaussian distribution. In our implementation we use the Knuth-Yao algorithm, which efficiently generates discrete Gaussian samples as already observed in several works. However, we are required to initialize the Knuth-Yao sampler for $\mathcal{D}_{p\mathbb{Z}+c}$ for all $c \in \mathbb{Z}_p$.

High Data Load Encryption In order to allow for high data load encryption, the number of polynomials $l > 0$ in $\mathbf{A}' \in \mathcal{R}_q^l$ must be non-zero. These polynomials are completely random and can hence be generated from a random seed. Therefore, it suffices to only store $\mathbf{A}'' \in \mathcal{R}_q^m$ and a seed for $\mathbf{A}'$ in its NTT representation allowing for faster operations while saving storage using a seed. Also with respect to security, one observes that increasing l does not decrease the bit security, since the optimal dimension has already been exceeded by $\mathbf{A}''$.

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 a pseudo-random generator using Salsa20 stream cipher in order to increase the number of output bits. This allows to produce as many random bits as required.

5.2 Performance Anaysis and Implementation Results

Parameter	**Description**	**Used for**
n	Dimension	$n = 512$
q	Modulus	$q \equiv 1 \bmod 2n$
$\mathcal{R}_q$	Cyclotomic ring	$\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$
p	Message range	$p = 2^x$, x bits/coeff.
αq	Parameter of the error distribution (e.g. $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$)	$\alpha q = 4.7 \cdot p$
m	Number of polynomials in $\mathbf{A}''$	$\mathbf{A}'' \in \mathcal{R}_q^m$
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 $\mathbf{A}'' \in \mathcal{R}_q^m$	$m = \bar{m} + k$
Message size	$(m+l)n\log_2(p)$	
Ciphertext size	$(m+l)n\log_2(q)$	
Public key size	$\mu + mn\log_2(q)$	
Secret key size	$\bar{m}nk\log_2(r_{sec})$	

Table 2. Parameters

We implemented both our scheme and the CPA-secure one from Lindner and Peikert LP11 on a machine that is specified by an

- Intel Core i7-4500U processor operating at 1.8GHz and 4GB of RAM. We used a gcc-4.8.2 compiler with compilation flags Ofast, mavx2, msse2avx, march=corei7-avx and march=core-avx-2.

The most time critical operations are polynom multiplications, which take about 14922 cycles including three NTT transformations, which require the major block of the running time.
Our performance results are given in Table 3 and Table 4, each for a different instantiation of the public key $\mathbf{A}$ according to Section 3.4. In particular, Table 3 contains the implementation results for the computational instantiation of $\mathbf{A}$, whereas Table 4 provides the corresponding data for a statistically instantiated public key. We provide timings, bit security, message sizes and message expansion factors (ciphertext size/message size ratio) depending on different parameters defined in Table 2 (see Appendix 5.2).

At the first glance, one observes that the ciphertext size of LP11 is larger than in our setting by a factor of $2 \cdot \log p$ for the same message size. 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 case we can encrypt $\log p = \log(\alpha q/4.7)$ bits per entry. This leads to message expansion factors of 5.75 resp. 2.9 (see Table 3 and Table 4) in case we encrypt 4 resp. 8 bits per entry (of size 23 bits) as compared to a factor of 46 for LP11. As an immediate consequence, our encryption engine must be much faster than LP11, since encryptions represent A-LWE instances resembling ordinary LWE samples in its purest form. In fact, when comparing the timings of both schemes, we observe that our scheme outperforms $\mathsf{LP11}$ by factors between 10 and approximately 24 for the same message size and conservatively chosen parameters in our setting. That is the timings of $\mathsf{LP11}$ would be even worse in case we choose the same discrete Gaussian parameter. Huge improvement 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. The overhead is solely restricted to generating new error polynomials and multiplying the secret with the random polynomials from $\mathbf{A}'$, which can in turn be generated from a random seed.

In our implementations, we let all error polynomials be identically distributed. In fact, one could get much better improvement factors, if l is increased and the error polynomials corresponding to $\mathbf{A}'$ are sampled from a wider discrete Gaussian distribution and hence encrypt more bits per entry while being still secure due to large αq. For decryption no length conditions are imposed on $\mathbf{A}'$. From a security point of view, our scheme has a bounded number of samples exceeding the optimal dimension with respect to the embedding approaches from [AFG13] and [BG14]. As a result, the bit security of the scheme lies independently from the number of samples $l + m$ between 279 and 395 for computationally instantiated public key and about 202 for a statistical instantiation, in case the ciphertext is attacked. This is much higher than compared to $\mathsf{LP11}$ for the selected parameters. Due to the bound from Lemma 5, there exists a relationship among the lengths resp. parameters of the error polynomials corresponding to $\mathbf{A}''$ and the secret keys, which should not exceed a certain bound. Therefore, one is recommended to select a proper tradeoff between these parameters. A lower parameter r_{sec} of the secret key allows to select a higher error parameter αq corresponding $\mathbf{A}''$, since $\mathbf{A}'$ is not affected. However, the decryption routine in $\mathsf{LP11}$ is faster (factors between 1.5 and 4) than ours in case $l = 0$. This is mainly due to the trapdoor resp. LWE inversion algorithm from Section 3.7, whose efficiency depends on the bound b, where a high bound allows only to recover a few bits per step. But for increasing l, the decryption engine of our scheme gets much faster. Once having recoverd $\mathbf{s}$ the ciphertext part $\hat{\mathbf{c}}'$ corresponding to $\mathbf{A}'$ is decrypted by $\hat{\mathbf{e}}' = \hat{\mathbf{c}}' - \mathbf{A}' \bmod q$, which is similar to $\mathbf{c}_1 \cdot \mathbf{r}_2 + \mathbf{c}_2$ from $\mathsf{LP11}$ in terms of operations, except that we can decrypt more bits per entry in our scheme as compared to $\mathsf{LP11}$. As a consequence, for large enough l decryption is performed faster than $\mathsf{LP11}$. For instance, in Table 3 our decryption routine outperforms $\mathsf{LP11}$ by a factor of approximately 3.3 for a message of size 1081344 bits and $l = 10$ and a factor of 1.7 for a message of size 258048 bits and $l = 2$.

	Parameters		**Sizes (bits)**		**Timings (cycles)**		**Security (bits)**		
	$\mathbf{r_{sec}}$	$\alpha\mathbf{q}$	**Message**	**Message Exp.**	**Encrypt**	**Decrypt**	**[AFG13]**	**[BG14]**	**Key Recov.**
This Work									
$\mathbf{l = 0}$									
	46	300	73728	3.8	2419200	6487200	279	355	272
	20	601	86016	3.3	2538000	6499800	331	436	207
	8	1203	98304	2.9	2460600	4190400	395	554	158
$\mathbf{l = 2}$									
	46	300	221184	3.8	5481000	7227000	279	355	272
	20	601	258048	3.3	5425200	7363800	331	436	207
	8	1203	294912	2.9	5603400	5029200	395	554	158
$\mathbf{l = 10}$									
	46	300	811008	3.8	17762400	11253600	279	355	272
	20	601	946176	3.3	17492400	11712600	331	436	207
	8	1203	1081344	2.9	18199800	9163800	395	554	158
LP11 **[LP11]**									
	-	300	73728	46	34326000	2070000	279	355	272
	-	75	86016	46	38664000	2592000	202	250	272
	-	75	98304	46	40986000	2844000	202	250	272
	-	75	258048	46	102024000	7218000	202	250	272
	-	75	294912	46	129042000	8766000	202	250	272
	-	75	946176	46	406800000	27756000	202	250	272
	-	75	1081344	46	433800000	30546000	202	250	272
Openssl – RSA 1024									
(80 bit security)	-	-	73728	1	159030000	247752000	-	-	-
	-	-	86016	1	184680000	287712000	-	-	-
	-	-	98304	1	210330000	327672000	-	-	-
	-	-	258048	1	552330000	860472000	-	-	-
	-	-	294912	1	630990000	983016000	-	-	-
	-	-	946176	1	2022930000	3151512000	-	-	-
	-	-	1081344	1	2311920000	3601728000	-	-	-

Table 3. Comparison of encryption schemes from LP [LP11] with Section 3.5 for $\mathbf{A} \approx_c \mathcal{U}(\mathbb{R}_q^t)$

	Parameters			**Sizes (bits)**		**Timings (cycles)**		**Security (bits)**		
	$\mathbf{r_{sec}}$	$\bar{\mathbf{m}}$	$\alpha\mathbf{q}$	**Message**	**Message Exp.**	**Encrypt**	**Decrypt**	**[AFG13]**	**[BG14]**	**Key Recov.**
$\mathbf{l = 0}$										
	16	46	75	141312	5.75	4698000	18005400	202	250	>> 250
	16	23	75	94208	5.75	3484800	11293200	202	250	>> 250
	8	46	75	141312	5.75	4743000	16788600	202	250	>> 250
	8	23	75	94208	5.75	3506400	10297800	202	250	>> 250
$\mathbf{l = 2}$										
	16	46	75	423936	5.75	12855600	20799000	202	250	>> 250
	16	23	75	282624	5.75	8931600	13102200	202	250	>> 250
	8	46	75	423936	5.75	12904200	19582200	202	250	>> 250
	8	23	75	282624	5.75	9073800	12193200	202	250	>> 250
$\mathbf{l = 10}$										
	16	46	75	1554432	5.75	45410400	32056200	202	250	>> 250
	16	23	75	1036288	5.75	30578400	20835000	202	250	>> 250
	8	46	75	1554432	5.75	45349200	30643200	202	250	>> 250
	8	23	75	1036288	5.75	30402000	19650600	202	250	>> 250
LP11 **[LP11]**										
	-	-	75	141312	46	56934000	4068000	202	250	272
	-	-	75	94208	46	37656000	2718000	202	250	272
	-	-	75	423936	46	172296000	12024000	202	250	272
	-	-	75	282624	46	113922000	7974000	202	250	272
	-	-	75	1554432	46	660600000	45324000	202	250	272
	-	-	75	1036288	46	451800000	30798000	202	250	272
Openssl – RSA 1024										
(80 bit security)	-	-	-	141312	1	302670000	471528000	-	-	-
	-	-	-	94208	1	201780000	314352000	-	-	-
	-	-	-	423936	1	906300000	1411920000	-	-	-
	-	-	-	282624	1	605340000	943056000	-	-	-
	-	-	-	1554432	1	3324240000	5178816000	-	-	-
	-	-	-	1036288	1	2216160000	3452544000	-	-	-

Table 4. Comparison of encryption schemes from LP [LP11] with Section 3.5 for $\mathbf{A} \approx_s \mathcal{U}(\mathbb{R}_q^t)$

Acknowledgements

The work presented in this paper was performed within the context of the Software Cluster project Sinnodium and Software Campus project *IT-GiKo* by Rachid El Bansarkhani. It was funded by the German Federal Ministry of Education and Research (BMBF). The authors thank Steven Galbraith for useful comments with respect to the preceding paper.

Bibliography

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

[ADL$^+$08] 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.

[AFG13] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In *ICISC 2013*, 2013.

[Ajt96] 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.

[AP09] 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.

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

[BF11a] 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.

[BF11b] 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.

[BG14] 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.

[BGG$^+$14] 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.

[BV11] 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.

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

[DDLL13] 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.

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

[EDB15] Rachid El Bansarkhani, Özgür Dagdelen, and Johannes Buchmann. Augmented learning with errors: The untapped potential of the error term, 2015. `http://eprint.iacr.org/`.

[Gen09] 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.

[GFS$^+$12] Norman Göttert, Thomas Feller, Michael Schneider, Sorin A. Huss, and Johannes Buchmann. On the design of hardware building blocks for modern lattice-based encryption schemes. In *CHES*, volume 7428 of *LNCS*, pages 512–529. Springer, 2012.

[GGH97] 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.

[GLP12] 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.

[GN08] 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.

[GOPS13] 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.

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

[GSW13] 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.

[HHGP$^+$03] 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.

[JD12] 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/`.

[KV09] 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.

[LMPR08] 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.

[LP11] 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.

[LPR10] 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.

[LPR13] 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.

[LPSS14] 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.

[Lyu12] 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.

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

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*. Springer, 2009.

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

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

[Pol71] John M. Pollard. The Fast Fourier Transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.

[PVW08] 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.

[PW08] 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.

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

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

[SSTX09] 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.

[Win96] Franz Winkler. *Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation)*. Springer, 1 edition, 1996.