Ring-LWE: Applications to cryptography and their efficient realization

Sujoy Sinha Roy, Angshuman Karmakar, and Ingrid Verbauwhede

ESAT/COSIC and iMinds, KU Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: {firstname.lastname}@esat.kuleuven.be

Abstract. The persistent progress of quantum computing with algorithms of Shor and Proos and Zalka has put our present RSA and ECC based public key cryptosystems at peril. There is a flurry of activity in cryptographic research community to replace classical cryptography schemes with their post-quantum counterparts. The learning with errors problem introduced by Oded Regev offers a way to design secure cryptography schemes in the post-quantum world. Later for efficiency LWE was adapted for ring polynomials known as Ring-LWE. In this paper we discuss some of these ring-LWE based schemes that have been designed. We have also drawn comparisons of different implementations of those schemes to illustrate their evolution from theoretical proposals to practically feasible schemes.

Keywords. post-quantum cryptography, learning with errors, ring learning with errors, implementations

1 Introduction

Post-quantum cryptography has become a popular research topic in cryptography in this decade. Our existing public-key infrastructures greatly rely on cryptographic primitives such as elliptic curve cryptography and RSA. The security if these primitives are based on the hardness of elliptic curve discrete logarithm problem and integer factorization. With our present day computers, these two problems remain computationally infeasible for sufficiently large key size. However a powerful quantum computer together with Shor's(RSA) and Proos and Zalka's(ECDLP) algorithm can solve these problems in polynomial time. Though there is no known powerful quantum computer till date, different organizations are trying to build quantum computers. In 2014 a BBC News article [2] reports an effort by the NSA. Due to these threats the need for quantum computer resistant public key cryptography has emerged. Recently NIST has recommended a gradual shift towards post-quantum cryptography [6] and have called for a standardization process for post-quantum cryptography schemes in the PQCrypto 2016 conference. Different organizations in the field of information storage and processing have responded to this call. For example, Google

has recently introduced the scheme *Frodo* [4] in 1% of all Chrome browsers. The world wide cryptography research community has proposed several candidates for post-quantum public key cryptography. Among them, the schemes based on lattices have received the highest attention thanks to their simpler arithmetic operations and wide range of applicability. In this paper we provide an overview of different lattice based constructions and discuss their implementations.

The LWE problem

The foundations of the cryptosystems that we discuss in this paper are based on the *learning with errors* (LWE) problem that was introduced in 2005 by Regev [28]. The problem is conjectured to be a hard problem and it is as hard as solving several worst-case lattice problems. For a lattice with dimension n, integer modulus q, and an error distribution $\mathcal{X}$ over the integers $\mathbb{Z}$, the LWE problem is defined as follows.

We denote vectors of dimension n by bold fonts. Generate a secret vector $\mathbf{s}$ of dimension n by choosing its coefficients uniformly in $\mathbb{Z}_q$. Generate $\mathbf{a}_i$ uniformly and the error terms e_i from $\mathcal{X}$. Next compute $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \in \mathbb{Z}_q$. The LWE distribution is denoted as $A_{s,\mathcal{X}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ and is the set of tuples $(\mathbf{a}_i, b_i)$. Solving the *decision LWE problem* is to distinguish with non-negligible advantage between the samples from $A_{s,\mathcal{X}}$ and the same number of samples drawn uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Solving the *search LWE problem* is to find $\mathbf{s}$ from a polynomial number of samples drawn from $A_{s,\mathcal{X}}$. The error distribution $\mathcal{X}$ is normally a discrete Gaussian distribution with a standard deviation σ.

The original LWE problem is defined over lattices and is not very efficient due to the use of large matrices. A more computationally efficient variant of the problem, known as the *ring-LWE problem* was introduced by Lyubashevsky, Peikert and Regev in [21]. The ring-LWE problem is defined over a polynomial ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle f \rangle$ where the irreducible polynomial $\langle f \rangle$ has degree n and the coefficients have modulus q. The problem is defined as follows. Sample a secret polynomial $s(x)$, and error polynomials $e_i(x) \in R_q$ with coefficients from $\mathcal{X}$. Next generate polynomials $a_i(x)$ with coefficients chosen uniformly from $\mathbb{Z}_q$. Compute $b_i(x) = a_i(x) \cdot s(x) + e_i(x) \in R_q$. The ring-LWE distribution is the set of polynomial tuples $(a_i(x), b_i(x))$. The *decision ring-LWE problem* is to distinguish between the samples $(a_i(x), b_i(x))$ and the same number of samples generated by choosing the coefficients uniformly. The *search ring-LWE problem* is to find the secret polynomial $s(x)$ from a polynomial number of samples drawn from the ring-LWE distribution. In the next section we will discuss different cryptographic primitives that have been designed using the ring-LWE problem.

Public-key encryption schemes

An encryption scheme based on the ring-LWE problem has been proposed by Lyubashevsky, Peikert and Regev in [21]. The steps are described below.

1. $KeyGen()$: Generate a polynomial $a \in R_q$ with coefficients chosen uniformly in $\mathbb{Z}_q$. Next sample two polynomials $r_1, r_2 \in R_q$ from $\mathcal{X}$ and compute $p = r_1 - a \cdot r_2 \in R_q$. The public key is (a, p) and the private key is r_2.
2. $Enc(a, p, m)$: First encode the message m to a polynomial $\bar{m} \in R_q$. Sample three polynomials $e_1, e_2, e_3 \in R_q$ from $\mathcal{X}$. The ciphertext is the pair of polynomials $c_1 = a \cdot e_1 + e_2$ and $c_2 = p \cdot e_1 + e_3 + \bar{m} \in R_q$.
3. $Dec(c_1, c_2, r_2)$: Compute $m' = c_1 \cdot r_2 + c_2 \in R_q$ and decode the coefficients of m' to either 0 or 1.

After the proposal of the encryption scheme, several implementations of the encryption scheme followed [12, 25, 29, 7, 3, 18, 27]. The basic arithmetic operations are polynomial multiplication, addition, subtraction, and generation of error polynomials from a discrete Gaussian distribution. For around 100 bit security, the implementations use a parameter set with $n = 256$, a 13-bit modulus q, and a *narrow* discrete Gaussian distribution with standard deviation σ around 4.5. Among all the arithmetic operations, polynomial multiplication is the costliest one. To perform fast polynomial multiplication, the implementations use the number theoretic transform (NTT) which is a variant of the fast Fourier transform (FFT) over integer rings. For the generation of error polynomials from the discrete Gaussian distribution $\mathcal{X}$, the implementations use one of the following sampling algorithms [8]: rejection sampling, inversion sampling and the Knuth-Yao sampling. In Fig. 1 a simplified hardware architecture for ring-LWE encryption [29] is shown. The architecture uses its polynomial arithmetic unit to perform polynomial addition and multiplication, and the discrete Gaussian sampler (based on Knuth-Yao algorithm) to generate the error polynomials. To achieve fast computation time, the architecture uses an efficient memory access scheme. For more details, authors may follow [29]. In Table 1 and Table 2 we show some of the implementation results on hardware and software platforms respectively for different parameter sets (n, q, σ).

Digital signature schemes

Using hard lattice problems to create efficient digital signature scheme was first demonstrated by Hoffstein et al. [15]. Their 'Hash and Sign' signature scheme NTRUSign was an extremely efficient scheme in practice but the original scheme's 'Hash and sign' approach leaks information about the private key,

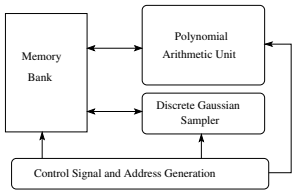

Fig. 1. Architecture (simplified) for ring-LWE encryption [29]

Implementation Algorithm	Parameters (n, q, σ)	Device	LUTs/FFs/ DSPs/BRAM18	Freq (MHz)	Cycles/Time(μs) Encryption	Decryption
Roy et. al.[29]	(256,7681,4.516)	Xilinx	1349/860/1/2	313	6.3k/20.1	2.8k/9.1
	(512,12289,4.859)	V6LX75T	1536/953/1/3	278	13.3k/47.9	5.8k/21
Pöppelmann et. al.[25]	(256,7681,4.516)	Xilinx	4549/3624/1/12	262	6.8k/26.2	4.4k/16.8
	(512,12289,4.859)	V6LX75T	5595/4760/1/14	251	13.7k/54.8	8.8k/35.4
RLWE-Enc[26]	(256,4096,3.33)	Xilinx	317/238/95/1	144	136k/946	-
RLWE-Dec		S6LX9	112/87/32/1	189	-	66k/351

Table 1. Performance of Ring-LWE encryption in hardware

Implementation Algorithm	Parameters (n, q, σ)	Device	Cycles Encryption	Decryption
Boorghany et. al.[3]	(256,7681,4.516)	ARM7TDMI	878,454	226,235
Boorghany et. al.[3]	(256,7681,4.516)	ATMega64	3,042,675	1,368,969
de Clercq et. al.[7]	(256,7681,4.516)	Cortex-M4F	121,166	43,324
Göttert et. al.[12]	(256,7681,4.516)	Core 2 Duo	4,560,000	1,710,000
Pöppelmann et. al.[27]	(256,7681,4.516)	AX128	874,347	215,863
Liu et. al. [19]	(256,7681,4.516)	AX128	666,671	299,538

Table 2. Performance of Ring-LWE encryption in software

namely the shape of the parallelepiped. It was first exploited by Gentry and Szydlo [11] and later Regev and Nguyen [23] developed this weakness further to show that an attacker can recover the private key with as few as 400 signatures.

Later Melchor et al. [22] used Gausssian sampling to hide this leakage efficiently using rejection sampling introduced by Lyubashevsky [20]. Though Lyubashevsky's scheme helped to create secure and efficient digital signature schemes like PASSSign [16] and BLISS [9], his scheme itself was very inefficient due to the requirement of sampling from Gaussian distributions with large standard deviation and very high rejection rates. From the computational point of view the most significant part of such signature schemes are polynomial multiplication and discrete Gaussian sampling. Unlike the encryption scheme, the standard deviation of the discrete Gaussian distribution is orders of magnitude larger to make these schemes secure and keep the signature sizes small. For example, the signature scheme by Lyubashevsky in [20] requires a standard deviation σ in between 3×10^4 and 1.7×10^5. Implementation of a fast sampler for such a large standard deviation is a difficult problem. Hence the focus has been in the direction of designing signature schemes with smaller standard deviation.

The Bimodal Lattice Signature Scheme known as BLISS [9] is a very popular lattice based signature scheme. It has been implemented on a wide variety of devices. The standard deviation of BLISS-I has $\sigma = 215$ for 128 bit security [9], which is of magnitude smaller than the previous signature schemes, but still larger than the σ used in encryption schemes.

For efficiency and security we need to store $O(\tau\sigma$ entries to sample from a discrete Gaussian distribution with standard deviation $\sigma(\tau = 12)$. The large memory requirement of BLISS makes it a challenging job to implement it on devices with limited memory and computing power. In addition, the authors

Implementation	Security	Signature Size	SK Size	PK Size	Sign(ms)	Verify(ms)
BLISS-0	$\leq$ 60 bits	3.3 kb	1.5 k b	3.3kb	0.241	0.017
BLISS-I	128 bits	5.6 kb	2 kb	7kb	0.124	0.030
BLISS-II	128 bits	5 kb	2 kb	7kb	0.480	0.030
BLISS-III	160 bits	6 kb	3 kb	7kb	0.203	0.032
BLISS-IV	192 bits	6.5 kb	3 kb	7kb	0.375	0.032
RSA-2048	103-112 bits	2 kb	2 kb	2kb	1.180	0.038
RSA-4096	$\geq$ 128 bits	4 kb	4 kb	4kb	8.660	0.138
ECDSA 256	128 bits	0.5 kb	0.25 kb	0.25kb	0.106	0.384
ECDSA 384	192 bits	0.75 kb	0.37 kb	0.37kb	0.195	0.853

Table 3. Benchmark on a (Intel Core i7 at 3.4 Ghz,32 GB RAM) with openssl 1.0.1c [9]ECDSA on a prime field F_p: ecdsap160,ecdsap256 and ecdsap384 in openssl

of BLISS also proposed a new Gaussian sampling technique that requires only $O(\log(\tau\sigma^2))$ storage thus making the scheme suitable scheme for small devices.

An efficient implementation of BLISS is by Pöppelmann and Ducas and Güneysu [24]. The implementation uses the Peikert's convolution lemma and the Kullback-Leibler divergence to design a practical and efficient discrete Gaussian sampler. Using the Peikert's convolution lemma, a sample from the distribution with $\sigma = 215.73$ is constructed by mixing two samples from a narrower distribution with $\sigma = 19.53$. This optimization is very useful since designing a sampler for such a small standard deviation is a lot easier. The Kullback-Leibler divergence is used to get a precision for a desired bit-security. A simplified architecture diagram of BLISS-I from [24] is shown in Fig. 2. The architecture is composed of a polynomial arithmetic unit, a discrete Gaussian sampler, a sparse polynomial multiplier, a compression block (which includes a rejection sampler), and a Huffman encoder. On a Xilinx Spartan-6 FPGA the implementation [24] takes 114.1 μs for signature generation and 61.2 μs for signature verification. It is worth noting here that there exists lattice based signature scheme that don't require Gaussian sampling at all [14] [13]. And thus trading off speed with signature size. Also scheme proposed by Bai and Galbraith [1] requires Gaussian sampling only in the keygen part, making the time critical signing process efficient.

Homomorphic encryption scheme

The beauty of the ring-LWE problem is that it is not restricted to encryption and signature schemes. It has been used to design efficient homomorphic encryption schemes. With homomorphic encryption, computations can be performed on encrypted data. Due to its homomorphism, equivalent computations are automati-

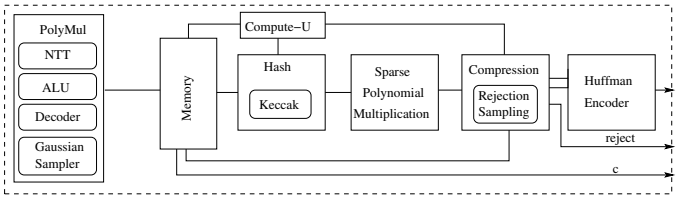

Fig. 2. Architecture for BLISS-I signing [24]

cally performed on the plaintext data. Thus with homomorphic operations, users can upload their encrypted data in a powerful cloud service and still perform computations in the cloud on the encrypted data. With the emergence of cloud service, a need for data privacy is gradually increasing. Beside data processing, homomorphic encryption can have applications in oblivious computations such as encrypted search. In an encrypted search, a user sends her encrypted keyword to a search engine and the search engine returns encrypted search result. The search engine and the associated data vendors are oblivious of the user's search.

A ring-LWE based homomorphic encryption scheme uses a basic ring-LWE encryption scheme and two additional functions `Add` and `Mult` to perform arithmetic operation on encrypted data. However in comparison to a simple ring-LWE encryption scheme, a homomorphic encryption scheme requires a much larger parameter set to support a desired multiplicative depth. An analysis on the choice of parameter set for a required multiplicative depth for two homomorphic encryption schemes FV [10] and YASHE [5] is provided by Tancrde and Naehrig in [17]. In the next part we provide some results for our implementation of the encryption/decrytion for a parameter set that supports a depth of four.

We first designed the YASHE homomorphic encryption scheme for the parameter set with irreducible polynomial degree $n = 2048$, 105-bit modulus q, and standard deviation $\sigma = 11.32$. The architecture uses full precision arithmetic: operations are performed modulo q. To perform coefficient-wise multiplication, a 106-bit Karatsuba multiplier is used. The architecture is implemented in a Xilinx ML605 board. with a Gigabit Ethernet interface. The homomorphic encryption-decryption processor consumes 6K LUTs, 5K FFs, 24 BRAMs and 27 DSP multipliers. At 125 MHz frequency, encryption takes 6.8 ms and decryption takes 6.5 ms. However later a sub-field attack became applicable for the YASHE scheme and the implementation became insecure.

Next we designed an architecture for the FV homomorphic scheme with a similar parameter. This scheme is secure against the recent sub-field attack. To achieve efficiency, we used the Chinese Remainder Theorem (CRT) in the polynomial arithmetic. With this, operations modulo q reduces into several smaller arithmetic operations modulo smaller primes. This is particularly suitable for FPGA implementation where the DSP multipliers have small data width. When implemented on the same FPGA board, the area consumption is: 6K LUTs, 4K FFs, 36 BRAMs and 12 DSP multipliers. At 125 MHz frequency, the architecture takes a total of 2ms to perform one encryption and one decryption. This is a lot faster than the previous implementation of the YASHE scheme, though the YASHE scheme is around 1.5 times faster than the FV scheme. The efficiency is achieved thanks to the use of CRT.

2 Current trends

In this paper we have presented an overview of the implementations of ring-LWE based cryptosystems. For public key encryption, ring-LWE encryption schemes are faster than ECC based schemes. However, memory requirement is a bottle-

neck for implementations on extremely resource constrained platforms such as passive RFID tags. The recent focus of the research community is to reduce the parameter size so that memory requirement can be reduced.

Due to its wide popularity in designing public key cryptography primitives and homomorphic encryption schemes, it is expected that in future more efficient schemes will emerge. Beside efficiency, a new focus in this area is in the direction of physical security of the schemes. The secret in a ring-LWE based scheme is a polynomial and arithmetic operations involve masking data and the secret using discrete Gaussian noise. Hence any leakage from the masking computation could reveal information about the secret to an attacker.

3 Acknowledgements

This work was supported in part by the Research Council KU Leuven: C16/15/058. G.0876.14N, and by the European Commission through the Horizon 2020 research and innovation programme under contract No H2020-ICT-2014-644371 WITDOM, H2020-ICT-2014-644209 HEAT and the ERC grant.

References

1. S. Bai and S. D. Galbraith. *An Improved Compression Technique for Signatures Based on Learning with Errors*, pages 28–47. Springer, Cham, 2014.
2. BBC News. NSA 'developing code-cracking quantum computer'. January 2014. http://www.bbc.com/news/technology-25588605.
3. A. Boorghany, S. B. Sarmadi, and R. Jalili. On Constrained Implementation of Lattice-based Cryptographic Primitives and Schemes on Smart Cards. Cryptology ePrint Archive, Report 2014/514, 2014.
4. J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. Cryptology ePrint Archive, Report 2016/659, 2016.
5. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Proc. of International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, 2013.
6. C. Boutin. Nist kicks off effort to defend encrypted data from quantum computer threat. April 2016. http://www.nist.gov/itl/csd/nist-kicks-off-effort-to-defend-encrypted-data-from-quantum-computer-threat.cfm.
7. R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede. Efficient software implementation of ring-lwe encryption. In *Proc. of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 339–344, 2015.
8. L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.
9. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. *Lattice Signatures and Bimodal Gaussians*, pages 40–56. Springe, Berlin, Heidelberg, 2013.
10. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/.
11. C. Gentry and M. Szydlo. *Cryptanalysis of the Revised NTRU Signature Scheme*, pages 299–320. Springer, Berlin, Heidelberg, 2002.

12. N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss. On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. *Cryptographic Hardware and Embedded Systems–CHES 2012*, 7428:512–529, 2012.
13. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. *Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems*. Springer, 2012.
14. T. Güneysu, T. Oder, T. Pöppelmann, and P. Schwabe. *Software Speed Records for Lattice-Based Signatures*, pages 67–82. Springer, 2013.
15. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Proceedings of the 2003 RSA Conference on The Cryptographers' Track*, CT-RSA'03, pages 122–140, Berlin, Heidelberg, 2003. Springer-Verlag.
16. J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, and W. Whyte. *Practical Signatures from the Partial Fourier Recovery Problem*, pages 476–493. Springer International Publishing, Cham, 2014.
17. T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology — AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2014.
18. M. Liu and P. Q. Nguyen. Solving BDD by Enumeration: An Update. In *Proceedings of the 13th International Conference on Topics in Cryptology*, CT-RSA'13, pages 293–309, Berlin, Heidelberg, 2013. Springer-Verlag.
19. Z. Liu, H. Seo, S. Sinha Roy, J. Großschädl, H. Kim, and I. Verbauwhede. *Efficient Ring-LWE Encryption on 8-Bit AVR Processors*, pages 663–682. Cryptographic Hardware and Embedded Systems – CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Springer, 2015.
20. V. Lyubashevsky. Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, 2011. http://eprint.iacr.org/2011/537.
21. V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. In *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.
22. C. A. Melchor, X. Boyen, J.-C. Deneuville, and P. Gaborit. *Sealing the Leak on Classical NTRU Signatures*, pages 1–21. Springer, Cham, 2014.
23. P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures, 2006.
24. T. Pöppelmann, L. Ducas, and T. Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. Cryptology ePrint Archive, Report 2014/254, 2014. http://eprint.iacr.org/.
25. T. Pöppelmann and T. Güneysu. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In *Selected Areas in Cryptography – SAC 2013*, Lecture Notes in Computer Science, pages 68–85. Springer, 2014.
26. T. Pöppelmann and T. Güneysu. Area Optimization of Lightweight Lattice-Based Encryption on Reconfigurable Hardware. In *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS-14)*, 2014, Preprint.
27. T. Pöppelmann, T. Oder, and T. Güneysu. *High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers*, pages 346–365. Springer, 2015.
28. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.
29. S. Sinha Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede. Compact ring-lwe cryptoprocessor. In *Cryptographic Hardware and Embedded Systems CHES 2014*, volume 8731 of *LNCS*, pages 371–391. Springer, 2014.