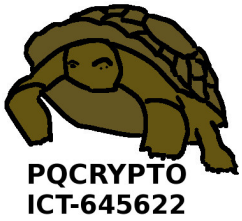

PQCRYPTO

Post-Quantum Cryptography for Long-Term Security

Project number: Horizon 2020 ICT-645622

Small Devices: D1.4 Intermediate Report on Optimized Hardware

Due date of deliverable: 30. September 2016
Actual submission date: May 3, 2017

Start date of project: 1. March 2015 Duration: 3 years

Coordinator:
Technische Universiteit Eindhoven
Email: coordinator@pqcrypto.eu.org
www.pqcrypto.eu.org

Revision 1

Project co-funded by the European Commission within Horizon 2020		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

Small Devices: D1.4 Intermediate Report on Optimized Hardware

Tim Güneysu, Tobias Oder

Contributors:

May 3, 2017
Revision 1

The work described in this report has in part been supported by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO.

Abstract

This document provides the PQCRYPTO project's intermediate report on optimized hardware. It provides the preliminary hardware implementation results of selected post-quantum schemes and corresponding parameters for embedded systems.

Keywords: Post-quantum cryptography, small devices, hardware implementations, public-key encryption, public-key signatures, secret-key encryption, secret-key authentication

Contents

1 Introduction

Modern asymmetric cryptosystems that are designed to face multiple threats and maintain long-term security require conservative parameter choices that typically pose a major implementation challenge for hardware devices. In this report we survey modern post-quantum schemes in regard to their implementation on reconfigurable and static hardware, also known as Field-Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) devices respectively. This report elaborates the advantages and disadvantages of both platforms and describes which platform should be used in which use cases. It reviews the most popular schemes in post-quantum cryptography that support confidentiality (by encryption) and authentication (by digital signatures). Note that the report is designed to focus on the implementation challenges and does not include any detailed discussion of the mathematical background required to understand the design rationales of the respective post-quantum constructions.

2 Target Platforms

In this report, we cover two classes of target architectures for efficient cryptographic implementations, field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs). An FPGA is a reconfigurable integrated circuit. In contrast to an ASIC, an FPGA is not bounded to one specific purpose since the configuration can be changed at any time after manufacturing and even after distribution in the field. This is especially useful when the security measures of a production system in the field turn out to be insecure. To achieve a high level of flexibility, the FPGA largely consists of a regular grid of configurable logic blocks (CLBs). The CLBs themselves consist of a number of slices and a switching matrix. While the slices contain the reconfigurable logic, the switch matrices connect adjacent CLBs and realize the routing of the signals. Besides CLBs, there are other resources available such as block RAM memory and digital signal processors (DSPs). FPGAs are used increasingly often in commercial products since their reconfigurability allows fast prototyping and therefore reduces the time to market. Also the regular structure leads to a simpler design cycle since routing, placement, and timing can mostly be automated.

On the other side, ASICs are designed to serve one particular purpose for which it comprises a specifically tailored hardware circuit. Once manufactured, the design is static and adjustments can only be made by physically exchanging the hardware. But this allows for more compact designs and speed optimizations since only components that are actually necessary are included in the design while FPGA designs most likely will not reach 100% utilization. More compact designs mean cheaper unit costs for high volume designs. Additionally, ASIC designs are usually less power-consuming than FPGA designs. But on the other hand, designing an ASIC comes with upfront non-recurring costs for development tools and expensive respins.

The choice of the target architecture mainly depends on the use case. If a highly optimized implementation is expected to meet the design requirements and high quantities of the product should be produced, it is probably best to choose an ASIC as the target platform as they are cheaper in production. For lower quantities and greater flexibility after distribution, FPGA implementations should be considered as they offer faster prototyping and are cheaper in development. In particular, the aspect of flexibility by FPGAs is often regarded as highly

beneficial for security applications since their cryptographic cores can be easily replaced in the field in case they turn out to be insecure. This particularly holds true for post-quantum cryptography: since many of the proposed constructions are rather new and thus have not undergone a thorough and long-lasting cryptanalytic review yet. FPGA implementations are preferable when quick upgrades are necessary and expected over the lifetime of the service. In this sense, we will put particular emphasis on FPGA implementations in this report.

3 Hardware Optimization

The efficiency of a hardware implementation strongly depends on the implemented algorithm and its opportunities for optimization. Optimization can happen on different levels. The implemented algorithms can be adjusted such that they better fit the constraints of a target platform (e.g. set the word size of the architecture as bit size of the modulus). During design stage, optimization can also happen on the architectural level. Finally, technical optimization includes the use of specific features of the target platform, like DSPs.

Commonly, algorithms that have a high degree of parallelism benefit more from hardware optimization than algorithms with a mostly serial structure. Parallelization can be used as an optimization technique on different levels. Basic operations, like XOR, can be executed in parallel; also, a design can contain multiple instances of an algorithm that run in parallel. Furthermore, algorithms with emphasis on the data flow instead of the control flow are better suited for hardware optimization since a simple control flow leads to simple state machines and a higher utilization of the used components. Finally, an efficient hardware implementation requires algorithms with operations that can easily be mapped to efficient hardware structures.

A hardware design can be improved with respect to different optimization goals. One goal is to reduce the area that is occupied by the implementation, which means a decrease in the number of components used. This is of particular interest since a smaller area consumption directly translates to a less expensive design. The second goal is to reduce the execution time and increase the throughput of the implementation, which matter, for instance, if the implementation has to be compliant with standard protocols that require a message to be delivered in a certain time frame. Real-time applications are another example where a low latency implementation is crucial. Finally, a wide range of use cases require a hardware implementation to be energy-efficient. In the Internet of Things, many devices only have access to a limited power supply (e.g., sensor networks) and therefore energy-efficient implementations are of notable importance for these applications. In many cases, the developer has to choose between one optimization goal and the other. That is, a faster design can often be achieved by making the design larger (more parallelization). On the other hand, reducing the number of used components usually leads to a lower power consumption. A careful assessment of the demands of the use case is necessary.

In a security-critical context, side-channel attacks have to be taken into account when developing hardware implementations. It is crucial to make sure an attacker with physical access to the target platform is not able to extract secret information by using side-channel information, like timing, power consumption, or electromagnetic emanation (EM). Countermeasures against side-channel attacks usually imply a massive overhead in at least one of the aforementioned metrics.

There are several approaches that can be used to optimize hardware implementations for a given optimization goal. The execution time of an implementation is determined by the

number of clock cycles and the frequency of the device. In order to increase the performance, it is possible to insert buffer storage elements for intermediate results (pipelining) that reduce the critical path and therefore allow a higher maximum frequency but also increase the number of clock cycles it takes to finish the operation. Loops in which one iteration step does not depend on the previous one can be unrolled to be executed in parallel and therefore increase the performance. On the downside, multiple processing elements are necessary. To allow faster memory access, multiple block RAM instances can be used to allow multiple read accesses in one clock cycle. Digital signal processors (DSP) can be used to speed up arithmetic operations.

Increasing the performance often means increasing the number of used components. Techniques to minimize the area consumption of an implementation are the reverse of the performance-oriented approach we described above. A serial execution of operations that share the same components can significantly reduce the resource consumption of the scheme. If block RAM units are used, they can store the maximum amount of data to keep the number of used memory components small even though that means that only one memory access per clock cycle is possible (two for dual-port memories). Large pre-computed tables can be calculated on-the-fly to avoid storing them in memory blocks. A smaller design usually refers to a more energy-efficient design. A low-power implementation can also be achieved by running the device at a lower clock frequency.

To make a design secure against side-channel attacks, one has to also design and implement dedicated countermeasures. Ensuring that an implementation has a constant execution time prevents an attacker from deducing information about the secret key by observing the running time of the device. The typical countermeasures against power and EM side-channels are masking and hiding of secret data.

4 Hardware Implementations

In this section we give an overview of existing implementations of post-quantum schemes. We are not aware of any hardware implementations of multivariate quadratic schemes. For hash-based scheme, there are implementations of one-time signature schemes [4], but as those are only able to generate a single signature for a given key pair, we did not consider those. Thus we mainly focus on implementations of lattice-based and code-based schemes. We also briefly review some hardware implementations of hash-functions as those are the building blocks for the hash-based signature schemes XMSS [7] and SPHINCS [5].

4.1 Encryption

There are several hardware implementations of post-quantum encryption schemes. The most relevant ones are reviewed here and summarized in Table 4.1.

4.1.1 Lattice-based Cryptography

Lattice-based cryptography can be divided into two groups, one for the schemes that base their security on standard lattices and one for schemes that base their security on ideal lattices. While the latter offers a higher performance and smaller key sizes they also introduce an additional structure in the underlying lattice and that is why standard lattice schemes are usually considered to be a more conservative choice.

Scheme	Security	Platform	FFs	LUTs	Slices	BRAMs	Time
QC-MDPC McE enc[17]	80 bits	XC6VLX240T	14,429	9,201	2,924	0	13.7 μs
QC-MDPC McE dec[17]	80 bits	XC6VLX240T	32,974	36,554	10,271	0	125.4 μs
QC-MDPC McE enc[39]	80 bits	XC6SLX4	119	226	64	1	3.4 ms
QC-MDPC McE dec[39]	80 bits	XC6SLX4	413	605	159	3	23.0 ms
Goppa McE enc[11]	80 bits	XC3S1400AN	804	1,044	668	3	2.2 ms
Goppa McE dec[11]	80 bits	XC3S1400AN	8,977	22,034	11,218	20	21.6 ms
Ring-LWE enc[36]	105 bits	XC6SLX9	238	317	95	2	0.9 ms
Ring-LWE dec[36]	105 bits	XC6SLX9	87	112	32	1	0.4 ms
Ring-LWE enc/dec[35]	105 bits	V6LX75T	3624	4549	1506	12	26 μs
Standard-LWE enc[20]	128 bits	S6LX45	4,676	6,078	1,866	73	0.8 ms
Standard-LWE dec[20]	128 bits	S6LX45	58	63	32	13	0.2 ms
Lattice-based IBE [38]	80 bits	S6LX25	6,067	7,023	-	16	80 μs
Lattice-based IBE [38]	192 bits	S6LX25	8,686	8,882	-	27	164 μs

Table 4.1: FPGA implementation results of post-quantum encryption schemes. Note that the given security levels are considering the pre-quantum setting.

The (standard) learning with errors (LWE) encryption scheme [31] has been implemented by Howe et al [20]. The two main operations in this scheme are matrix-vector multiplication and Gaussian sampling. The matrix-vector multiplications are performed serially with the help of a single DSP. Gaussian samples are generated using a Bernoulli sampler [9] as it does not require large precomputed tables. The uniformly distributed random numbers that the Gaussian sampler requires as input are generated using the stream cipher Trivium [8]. Their implementation of the LWE encryption requires 6,078 LUTs, 4,676 FFs, and 1,811 slices on a Spartan-6 FPGA. Due to the size of the keys the implementation also requires 73 BRAM modules. This number can be reduces by generating parts of the public key on-the-fly instead of storing it precomputed in BRAMs.

The counterpart to LWE in ideal lattices is called ring learning with errors (R-LWE) [28]. This scheme has been implemented several times. The first implementation of R-LWE has been published by Göttert et al. in 2012 [14]. he authors presented a hardware implementation of the RLWE encryption scheme on a Virtex 7 FPGA. To achieve an acceptable level of performance, the authors tweaked the parameters of the scheme to be able to use the Number-theoretic transform (NTT) for lowering the complexity of polynomial multiplication from $O(n^2)$ to $O(n \log(n)$. In contrast to matrix-vector multiplication, polynomial multiplication in the frequency domain, computed using NTT, can be optimized in several ways. During the transformation, it is necessary to compute the twiddle factors, which are powers of a root of unity. Those twiddle factors can be precomputed or calculated on-the-fly. Designers can choose the preferred implementation depending on the design goals, namely whether the implementation should be optimized for memory consumption or performance. The core operation of the NTT is the butterfly operation that takes two coefficients of the polynomial and performs one multiplication, one addition, and one subtraction. Multiple butterfly operations can be executed in parallel.

Pöppelmann and Güneysu [34] presented an optimized NTT multiplier. Their work was further extended to implement a complete RLWE encryption scheme in 2013 [35]. While the design by Göttert et al. was large and could only be placed on large Virtex-7 FPGAs, Pöppel-

mann and Güneysu proposed an architecture suitable for smaller reconfigurable devices, such as a Spartan 6 device. Furthermore, since their implementation relies on a generic microcode engine, it can also be used for other lattice-based implementations. Since then, several further optimizations have been proposed. Aysu et al. reduced the area consumption of the NTT [3]. Roy et al. enhanced the performance of NTT [37] by optimizing the memory access and simplifying the structure of the algorithm. That design was further optimized by using a more efficient Knuth-Yao sampler [25] which requires less FPGA resources. The smallest FPGA implementation, to the best of our knowledge, has been presented by Pöppelmann and Güneysu [36], the overall resource occupation of which is 32 slices, 1 BRAM, and 1 DSP. To achieve such a low area design, the authors chose a parameter set for which the modulus is a power of two. As a result, there was no need for a modular reduction step. The drawback of the proposed set of parameters is that the NTT is no longer applicable. As a result, the computation time is increased by one order of magnitude.

There are also a number of FPGA implementations of the NTRU encryption scheme [19], like [22] or [27]. However as NTRU is still protected by patents [18] we do not further consider NTRU implementations.

Lattice-based cryptography also allows practical identity-based encryption. The identity-based encryption proposed by Ducas et al. [10] has been implemented for an FPGA as well [38]. However, the implementation only covers the encryption and the decryption that are very similar the the R-LWE encryption scheme. The master key generation and the user key generation has not been implemented as the constrained resources of an FPGA are not sufficient to perform these operations.

4.1.2 Code-based Cryptography

The code-based encryption scheme McEliece [29] and its variant by Niederreiter [30] have also been implemented on hardware devices. McEliece can be instantiated with different codes. While the original scheme uses Goppa codes, quasi-cyclic modest density parity-check (QC-MDPC) codes provide a better efficiency. However, QC-MDPC codes provide an additional structure that might be exploitable. Choosing QC-MDPC codes over Goppa codes is therefore similar to choosing ideal lattices over standard lattice.

McEliece with Goppa codes has been implemented on a Xilinx Spartan-3AN FPGA by Eisenbarth et al [11]. As the implementation relies on Goppa codes, the public and the secret key are huge matrices. The McEliece decryption is much more complex than the encryption due to the decoding algorithm that is used to recover the message. Thus, the implementation of [11] needs 668 slices, 1,044 LUTs, 804 FFs, and 3 BRAMs for the encryption, but 11,218 slices, 22,034 LUTs, 8,977 FFs, 20 BRAMs for the decryption. Both modules (encryption and decryption) also need 4,644 Kbits of Flash memory. The design was improved by Ghosh et al. [13] by reducing the number of required slices during the decryption to 2,979 and the number BRAMs to 5. Furthermore the decryption latency was reduced to 1 ms instead of 10.8 ms.

The performance and resource consumption of QC-MDPC McEliece has been evaluated on reconfigurable hardware in [17] and [39]. While the work of [17] aims for a high-speed implementation for Virtex-6 FPGAs, [39] focuses more on developing a lightweight implementation that even fits on a low-cost Spartan 6-FPGA. Thus the results are very different. While the high-speed implementation of [17] takes 13.7 microseconds for encryption and 125.4 microseconds for decryption, the lightweight implementation of [39] is two orders of magnitude slower as it takes 3.4 milliseconds for encryption and 23 milliseconds for decryption. On the other

hand, the lightweight implementation takes much less resources. The encryption takes only 119 FFs, 226 LUTs, and 64 slices while the high-speed encryption needs 14,429 FFs, 9,201 LUTs, and 2,924 Slices. However, the lightweight implementation requires 1 resp. 3 BRAMs for encryption resp. decryption while the high-speed implementation does not require any.

4.2 Digital Signatures

For post-quantum digital signatures, less hardware implementations exist. The implementations we are aware of are summarized in Table 4.2.

Scheme	Security	Platform	FFs	LUTs	Slices	BRAMs	Time
GLP-Sign[15]	80 bits	XC6SLX16	8,993	7,465	2,273	29.5	1 ms
GLP-Verify[15]	80 bits	XC6SLX16	6,663	6,225	2,263	15	1 ms
BLISS-Sign[33]	128 bits	XC6SLX16	6,420	7,193	2,291	5.5	114 μs
BLISS-Verify[33]	128 bits	XC6SLX16	4,312	5,065	1,687	4	58 μs

Table 4.2: FPGA implementation results of post-quantum signature schemes. Note that the given security levels are considering the pre-quantum setting.

4.2.1 Lattice-based Cryptography

Implementing lattice-based signature schemes is a more challenging task, since the standard deviation of the Gaussian sampler is usually much higher compared to encryption schemes. Furthermore, additional components, such as hash functions and a rejection step, are required. So far, only ideal lattice-based signature schemes have been implemented on FPGAs. The main reason is that standard lattice signature schemes like TESLA [1] need to be instantiated with very large parameters.

The GLP signature scheme [15] was presented by G$\breve{}$neysu et al. in 2012 and implemented on a Spartan 6 FPGA. The BLISS digital signature scheme [9] was also implemented in hardware [33]. The major difference between both schemes is that the error polynomials in BLISS have Gaussian distributed coefficients and ternary coefficients in GLP. The BLISS FPGA design uses a table-based Gaussian sampler which, thanks to the Kullback-Leibler divergence, can be implemented using little memory without affecting the performance. Implementing Gaussian samplers in hardware is a challenging task: Many samplers have an non-constant running time; e.g., when using rejection sampling, the sampler could theoretically require an infinite number of iterations. Other samplers need large precomputed tables that contribute to a significant share of the implementations overall memory consumption. In general, the Knuth-Yao sampler is considered a good trade-off between runtime and memory consumption. For small standard deviations, the binomial sampler [2] is a good choice since it does not require any precomputed tables and has a constant running time. But as the standard deviation grows, other techniques, like the already mentioned Knuth-Yao, become more interesting since they feature a lower average running time and entropy consumption. The entropy consumption of a sampler is important because it is also necessary to implement a pseudo-random number generator that produces uniformly random bits.

4.2.2 Hash-based Cryptography

We are not aware of any hardware implementations of hash-based signature schemes apart from one-time signature schemes. However, the main building block in hash-based cryptography are hash functions and hardware implementations of those are plentiful. The Third SHA-3 Candidate Conference brought up lots of implementations results, like [16, 12, 26, 24, 23, 21]. It outperforms SHA-2 by an order of magnitude and is therefore an interesting candidate to instantiate hash-based signature schemes with.

4.3 Key Exchange

While there are promising lattice-based key exchange schemes, like New Hope [2] and Frodo [6], we are not aware of any hardware implementations of those schemes. Furthermore, the McEliece encryption scheme can be used as key encapsulation mechanism[32]. There are microcontroller implementations of a McEliece KEM [40] but unfortunately no hardware implementations.

5 Conclusions

A lot of post-quantum schemes have been implemented already, especially lattice-based and code-based schemes. So far, ideal lattice-based schemes appear to be the most efficient. However, the additional structure in the underlying lattice is still considered to be a potential thread to its security even though no attacks have been found yet that could exploit this structure. It would be interesting to also compare these implementations to implementations of hash-based schemes, like XMSS or SPHINCS.

References

[1] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, and Özgür Dagdelen. TESLA: tightly-secure efficient signatures from standard lattices. *IACR Cryptology ePrint Archive*, 2015:755, 2015.

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange – a new hope. In *Proceedings of the 25th USENIX Security Symposium*. USENIX, (to appear). Document ID: 0462d84a3d34b12b75e8f5e4ca032869, http://cryptojedi.org/papers/#newhope.

[3] Aydin Aysu, Cameron Patterson, and Patrick Schaumont. Low-cost and area-efficient FPGA implementations of lattice-based cryptography. In *HOST*, pages 81–86. IEEE Computer Society, 2013.

[4] Aydin Aysu and Patrick Schaumont. Precomputation methods for hash-based signatures on energy-harvesting platforms. *IEEE Trans. Computers*, 65(9):2925–2931, 2016.

[5] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical Stateless Hash-Based Signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual*

International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 2015.

[6] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. *IACR Cryptology ePrint Archive*, 2016:659, 2016.

[7] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2011.

[8] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2006.

[9] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology–CRYPTO 2013*, pages 40–56. Springer, 2013.

[10] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2014.

[11] Thomas Eisenbarth, Tim Güneysu, Stefan Heyse, and Christof Paar. MicroEliece: McEliece for embedded devices. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2009.

[12] Kris Gaj, Ekawat Homsirikamol, Marcin Rogawski, Rabia Shahid, and Malik Umar Sharif. Comprehensive evaluation of high-speed and medium-speed implementations of five SHA-3 finalists using xilinx and altera fpgas. *IACR Cryptology ePrint Archive*, 2012:368, 2012.

[13] Santosh Ghosh, Jeroen Delvaux, Leif Uhsadel, and Ingrid Verbauwhede. A speed area optimized embedded co-processor for McEliece cryptosystem. In *23rd IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2012, Delft, The Netherlands, July 9-11, 2012*, pages 102–108. IEEE Computer Society, 2012.

[14] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In *Cryptographic Hardware and Embedded Systems–CHES 2012*, pages 512–529. Springer, 2012.

[15] 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 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.

[16] Frank K Gürkaynak, Kris Gaj, Beat Muheim, Ekawat Homsirikamol, Christoph Keller, Marcin Rogawski, Hubert Kaeslin, and Jens-Peter Kaps. Lessons learned from designing a 65 nm asic for third round sha-3 candidates. 2012.

[17] Stefan Heyse, Ingo von Maurich, and Tim Güneysu. Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 273–292. Springer, 2013.

[18] J. Hoffstein, J. Pipher, and J.H. Silverman. Public key cryptosystem method and apparatus, June 27 2000. US Patent 6,081,597.

[19] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[20] James Howe, Ciara Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and K. Beeden. Standard lattices in hardware. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, pages 162:1–162:6. ACM, 2016.

[21] Bernhard Jungk. Evaluation of compact fpga implementations for all sha-3 finalists. In *The Third SHA-3 Candidate Conference*, 2012.

[22] Abdel Alim Kamal and Amr M Youssef. An fpga implementation of the ntruencrypt cryptosystem. In *Microelectronics (ICM), 2009 International Conference on*, pages 209–212. IEEE, 2009.

[23] Jens-Peter Kaps, Panasayya Yalla, Kishore Kumar Surapathi, Bilal Habib, Susheel Vadlamudi, and Smriti Gurung. Lightweight implementations of sha-3 finalists on fpgas. In *The Third SHA-3 Candidate Conference*, 2012.

[24] Elif Bilge Kavun and Tolga Yalcin. On the suitability of sha-3 finalists for lightweight applications. 2012.

[25] Donald E Knuth and Andrew C Yao. The complexity of nonuniform random number generation. *Algorithms and complexity: new directions and recent results*, pages 357–428, 1976.

[26] Kashif Latif, M Muzaffar Rao, Arshad Aziz, and Athar Mahboob. Efficient hardware implementations and hardware performance evaluation of sha-3 finalists. 2012.

[27] Bingxin Liu and Huapeng Wu. Efficient architecture and implementation for ntruencrypt system. In *Circuits and Systems (MWSCAS), 2015 IEEE 58th International Midwest Symposium on*, pages 1–4. IEEE, 2015.

[28] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[29] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, 1978.

[30] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform.*, 15(2):159–166, 1986.

[31] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[32] Edoardo Persichetti. Secure and anonymous hybrid encryption from coding theory. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, volume 7932 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2013.

[33] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2014.

[34] Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In *Progress in Cryptology–LATINCRYPT 2012*, pages 139–158. Springer, 2012.

[35] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *Selected Areas in Cryptography–SAC 2013*, pages 68–85. Springer, 2013.

[36] Thomas Pöppelmann and Tim Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2796–2799. IEEE, 2014.

[37] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact ring-LWE cryptoprocessor. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 371–391. Springer, 2014.

[38] Tobias Oder Tim Güneysu. Towards lightweight identity-based encryption for the post-quantum-secure internet of things. In *18th International Symposium on Quality Electronic Design (ISQED 2017)*. IEEE, 2017.

[39] Ingo von Maurich and Tim Güneysu. Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices. In Gerhard Fettweis and Wolfgang Nebel,

editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6. European Design and Automation Association, 2014.

[40] Ingo von Maurich, Lukas Heberle, and Tim Güneysu. IND-CCA secure hybrid encryption from QC-MDPC Niederreiter. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2016.