Secure and efficient software implementation of QC-MDPC code-based cryptography

Antonio Guimarães[1], **Diego F. Aranha**[2] **(advisor), Edson Borin**[1] **(co-advisor)**

[1]Institute of Computing, University of Campinas, Campinas, Brazil

[2]Department of Engineering, Aarhus University, Aarhus, Denmark

{antonio.guimaraes,edson}@ic.unicamp.br, dfaranha@eng.au.dk

Abstract. *The emergence of quantum computers is pushing an unprecedented transition in the public key cryptography field. Algorithms in the current standard are vulnerable to attacks using quantum computers and need, therefore, to be replaced. Cryptosystems based on error-correcting codes are considered some of the most promising candidates to replace them for encryption schemes. Among the code families, QC-MDPC codes achieve the smallest key sizes while maintaining the desired security properties. Their performance, however, still needs to be greatly improved to reach a competitive level. In this work, we optimize the performance of QC-MDPC code-based cryptosystems through improvements concerning both their implementations and algorithms. We first present a new enhanced version of QcBits' key encapsulation mechanism, which is a constant time implementation of the Niederreiter cryptosystem using QC-MDPC codes. Comparing with the current state-of-the-art, the BIKE implementation, our code performs 1.9 times faster when decrypting messages. We then optimize the performance of QC-MDPC code-based cryptosystems through the insertion of a configurable failure rate in their arithmetic procedures. Using a failure rate negligible compared to the security level (2^{-128}), we achieve speedups of 1.6 to 2 times in some arithmetic algorithms. By inserting these algorithms in our enhanced version of QcBits, we were able to achieve a speedup of 1.9 on the key generation and up to 1.4 on the decryption time. Comparing with BIKE, our final version of QcBits performs the uniform decryption 2.7 times faster. Moreover, the techniques presented in this work can also be applied to BIKE, opening new possibilities for further improvements.*

1. Introduction

Public-key cryptography has a major role in providing computing systems with capabilities such as data confidentiality, authentication, and integrity. These capabilities, in turn, are often essential when transmitting information and, more generally, in most human interactions with modern technology. The current standards of public-key cryptography for encryption are the RSA [Rivest et al. 1978] and elliptic curve cryptography (ECC) [Koblitz 1987]. They are considered secure and computationally efficient techniques and meet present needs. However, the computing field might be on the verge of a new technological breakthrough: the creation of large scalable quantum computers. Developing applications that could benefit from them is mostly an open field of study, but their impact on the current public-key cryptography standard has long been known. In

1994, Peter Shor formulated an algorithm that can solve integer factorization in polynomial time using a quantum computer [Shor 1994]. This is the problem in which the RSA is based, and a polynomial-time solution for it entirely undermines the security. The same occurs with the discrete logarithm problem, the base of the ECC.

While the community diverges over predictions, some specialists foresee quantum computers capable of breaking the 2048-bit RSA in the next few decades [Moody 2016]. Therefore, a secure and efficient replacement for the current standard of public-key cryptography is necessary. Considering this, in 2016, the USA's National Institute of Standards and Technology (NIST) started the standardization process for *post-quantum* public-key cryptography [NIST 2016]. Among the many alternatives, the code-based cryptography presents some of the most promising candidates. Discovered in 1978 with the McEliece Cryptosystem [McEliece 1978], it is based on a known NP-complete problem (the decoding of general linear codes) and has so far shown to be resistant against attacks using quantum computers.

In its original form, the McEliece cryptosystem relies on very large public keys, which represents an obstacle to the cryptosystem adoption. Derivatives using smaller keys have been proposed, but they usually result in the introduction of vulnerabilities. An exception to that is the implementation of McEliece using QC-MDPC codes, which is believed to be secure. Presented in 2013 [Misoczki et al. 2013], the cryptosystem provides keys about 100 times smaller than the original McEliece, but it comes at the cost of deteriorating the performance and introducing perceptible failure rates to the decryption process. In this context, performance improvement to QC-MDPC code-based cryptosystems is necessary and, in this work[1], we present some contributions toward this goal.

We build our contributions over QcBits [Chou 2016], a constant-time implementation of a QC-MDPC code-based encryption scheme. We compare our results with the variant 2 of the BIKE suite [Aragon et al. 2017], which is the current state-of-the-art implementation for QC-MDPC codes. BIKE is a candidate at NIST's competition [NIST 2016] and its variant 2 implements the same encryption scheme as QcBits. The constant-time execution, presented by QcBits, is an important feature to avoid timing side-channel attacks. To briefly define, side-channel attacks are those that exploit the possible correlation between the cryptosystem secret data (keys and plain-text) and the physical behavior of the hosting machine during its execution. In the case of timing side-channel attacks, an attacker could, for example, measure the execution time of multiple executions of the cryptosystem and try to infer some information about the data being processed. In a constant-time implementation, such as QcBits, the execution time does not depend on any data being processed, except for data that is public by construction (e.g. the public key, the security parameters, and message length). A *uniform implementation* is a more relaxed version of this concept (Section 2.5.1 of the dissertation). Unless indicated otherwise (e.g. Uniform Decryption), all of our implementations in this work feature constant-time execution.

[1]The complete dissertation is available at `http://repositorio.unicamp.br/bitstream/REPOSIP/334351/1/GuimaraesJunior_AntonioCarlos_M.pdf`

1.1. Contributions and Publications

The contributions presented in this work are divided into three sets. The first two concern the development of new implementation techniques aiming at optimizing the performance of QcBits. They are presented in Sections 2 and 3, and were published at the *Brazilian Symposium on High-Performance Computational Systems* (WSCAD-2017) [Guimarães et al. 2017] and at Wiley's *Concurrency and Computation: Practice and Experience* (CCPE) journal [Guimarães et al. 2018]. The last one concerns the presentation of improvements in the basic arithmetic algorithms necessary to implement a QC-MDPC code-based cryptosystem. This set presents contributions much more generic and that can be explored in other fields of cryptography, even though they were planned in a specific context. It is presented in Section 4 and was published at the *International Workshop on Code-Based Cryptography* [Guimarães et al. 2019].

In 2019, this dissertation received the *Best Dissertation Award* at the CTD of the *Brazilian Symposium on High-Performance Computational Systems* (WSCAD 2019).

2. Accelerating the decoding of QcBits

The original QcBits presents a very good performance level due to the employed techniques and some of the algorithm choices. However, it does not exploit modern instruction set extensions which could improve the performance even further. The decoding of QC-MDPC codes is probabilistic and, hence, it is usually hard to efficiently implement it in constant-time. Considering this, we start our optimizations of QcBits with the decoding process and presented the following contributions.

- The vectorization of the entire code using SSE, AVX2 and AVX512 instructions.
- The implementation of a pre-calculation technique in a word permutation procedure that used to take almost 40% of the execution time.
- A performance gain estimation that could be as high as 5.06 times considering the introduction of simple and generic extensions to the Intel x86 architecture.
- The mitigation of all known power vulnerabilities found in the original implementation with an almost negligible ($< 1\%$) impact on the overall performance.

In terms of performance, our contributions led to speedups of up to 2.6, 4.3 and 10.5 times using SSE, AVX2 and AVX512 instructions, respectively. It should be noted that these gains surpass the number of SIMD lanes found on these standards. Figure 1 shows the final speedups achieved using three industry-standard compilers on two different architectures. `REF` and `CLMUL` indicate two different versions of the original QcBits.

3. An enhanced version of QcBits

The decoding process is arguably the most critical implementation in QC-MDPC cryptosystems considering both performance and security. Nonetheless, key generation and encryption are also important procedures that we could improve in QcBits. Since QcBits was published using outdated parameters that achieve only a 40-bit quantum security level, we decided to continue our optimization by entirely rewriting it, updating the security level, optimizing all its processes and reapplying all our previously developed techniques. Our contributions are summarized below.

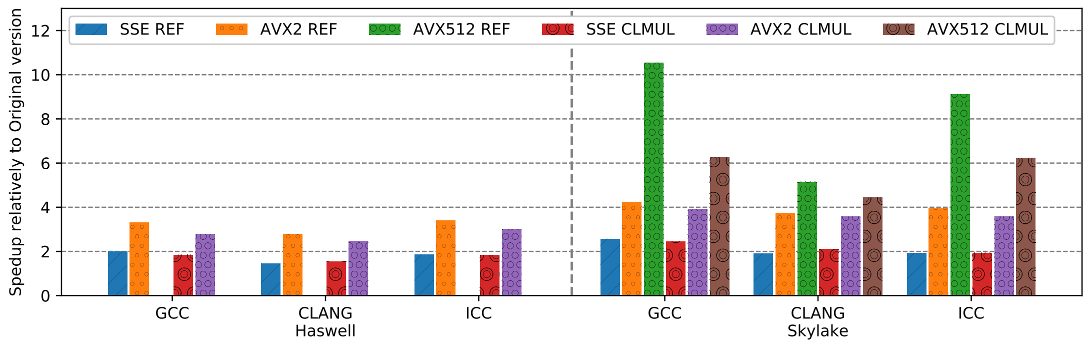

Figure 1. Final speedups achieved with the optimization (relatively to the corresponding Original version execution)

- The update of the security level from the 40-bit quantum security level to the 128-bit quantum security level, meeting NIST's highest security level required for the standardization process.
- The vectorization of the entire implementation using AVX512 instructions.
- The replacement of some of the core algorithms with others that have a better performance in the face of the new AVX512 instructions.
- The implementation of BIKE's batch key generation using QcBits' algorithms.

Our rewritten version of QcBits confirms the impact of our previously developed optimization techniques as well as the new improvements. We achieved speedups of up to 1.9 times in comparison with BIKE and most of our optimizations can be applied to BIKE itself, which could bring even faster implementations for QC-MDPC code-based cryptography. Table 1 details our results. BIKE does not present a constant-time implementation of the key generation procedure for us to compare with. Comparing with the non-constant-time key generation of BIKE, ours is only 3.1 times slower, which is generally a small slowdown for a constant-time implementation.

Table 1. Performance comparison between this work and BIKE (in cycles)

Operation	This work	BIKE 2	Speedup over BIKE
Key Generation	40,265,904	12,944,920	0.32 [a]
Batch Key Gen. (100 keys) [c]	1,231,700	967,331 [b]	0.79 [a]
Batch Key Gen. (400 keys) [c]	928,994	422,133 [b]	0.45 [a]
Encryption	259,306	348,227	1.34
Constant Time Decryption	9,803,835	-	-
Uniform Decryption	5,008,429	9,572,412	1.91

[a] BIKE's polynomial inversion is not constant time.
[b] Result from a fully non-constant time version.
[c] Cost per key

4. Introducing failures to accelerate QC-MDPC code-based cryptography

One of the main features of QC-MDPC cryptosystems is the use of the Quasi-Cyclic (QC) structure, which enables cryptosystems to be entirely implemented using arithmetic over binary polynomials. Three arithmetic operations are executed over them: addition, multiplication, and inversion. The first is efficient, but the others usually represent over 90%

of the execution time. Algorithms for arithmetic over binary polynomials are very well-known and studied. Hence, new generic optimizations for them are usually restrained to the implementation aspect only. In this work, we optimize binary field arithmetic focusing specifically on the case of QC-MDPC code-based cryptography. We modified the algorithms to accept configurable parameters that greatly accelerate them at the cost of introducing a negligible probability of failure depending on the input. Then, we defined methods to correlate this probability of failure (or failure rate) of each algorithm with the impact on performance. In this way, we present the following contributions.

- We introduce the concept of using arithmetic subroutines with a controlled failure rate to accelerate QC-MDPC code-based cryptosystems.
- We present constant-time algorithms for multiplication and inversion over binary polynomials that operate with configurable failure rates.
- We define methods to obtain a correlation between failure rate and performance improvement for each algorithm.
- We show that these algorithms provide a significant performance improvement while introducing an arithmetic failure rate that is negligible ($< 2^{-128}$).

Table 2 shows the execution time of the arithmetic algorithms. By introducing them in our enhanced version of QcBits, we achieve a speedup of 1.9 times on the key generation and 1.4 times on the decryption process. Comparing with BIKE, our final version of QcBits performs the uniform decryption 2.7 times faster. Table 3 details the results. They show a significant performance impact of our approach, while the negligible failure rate has almost no downsides to the cryptosystem. The correlation between failure rate and performance improvement was also shown to be very promising, once it is possible to achieve much lower failure rates with little performance penalties. The algorithms we presented were evaluated in the context of QC-MDPC cryptosystems, but, ultimately, they are generic algorithms for arithmetic in GF(2^n) and, thus, could be used in other contexts.

Table 2. Comparison among implementations of multiplication and inversion. Bolded texts represent results from this section.

Operation	Implementation	Failure Rate	Constant Time	Cost (cycles)
Inversion	**Wu *et al.* modified**	2^{-128}	Yes	20,773,925
	Wu *et al.* modified	10^{-8}	Yes	14,979,764
	Wu *et al.*	0	Yes	39,747,301
	NTL [Shoup 2003]	0	No	12,088,846
	Itoh-Tsujii	0	Yes	243,226,595
Multiplication	**Sparse Mult.**	2^{-128}	Yes	79,843
	Sparse Mult.	0	Yes	130,023
	NTL [Shoup 2003]	0	?	161,715

References

Aragon, N., Barreto, P. S. L. M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Gueron, S., Guneysu, T., Aguilar Melchor, C., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.-P., and Zémor, G. (2017). BIKE: Bit Flipping Key Encapsulation. Submission to the NIST post quantum standardization process. Website: `http://bikesuite.org/`.

Table 3. Execution time (cycles) of QcBits, with (this Section) and without (Section 3) the Arithmetic Failure Rate; and of the official BIKE implementation, for comparison.

	128-bit QcBits			BIKE-2	
	This section	Section 3	Speedup	Additional	Speedup
Key Generation	21,332,058	40,265,904	1.89	12,944,920	0.61 *
Encryption	256,655	259,306	1.01	348,227	1.36
Constant-Time Decrypt.	8,016,312	9,803,835	1.22	**	**
Uniform Decryption	3,505,079	5,008,429	1.43	9,572,412	2.73

* BIKE's polynomial inversion is not constant-time.
** BIKE does not present constant-time decryption.
The results for BIKE are equivalent to the column "Constant time implementation" and line "AVX-512" from Table 19 in their paper [Aragon et al. 2017].

Chou, T. (2016). QcBits: Constant-Time Small-Key Code-Based Cryptography. In Gierlichs, B. and Poschmann, A. Y., editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 280–300, Berlin, Heidelberg. Springer Berlin Heidelberg.

Guimarães, A., Aranha, D. F., and Borin, E. (2017). Optimizing the decoding process of a post-quantum cryptographic algorithm. *XVIII Simpósio em Sistemas Computacionais de Alto Desempenho-WSCAD*, 18(1/2017):160–171.

Guimarães, A., Borin, E., and Aranha, D. F. (2019). Introducing Arithmetic Failures to Accelerate QC-MDPC Code-Based Cryptography. In Baldi, M., Persichetti, E., and Santini, P., editors, *Code-Based Cryptography*, pages 44–68, Cham. Springer International Publishing.

Guimarães, A., Aranha, D. F., and Borin, E. (2018). Optimized implementation of QC-MDPC code-based cryptography. *Concurrency and Computation: Practice and Experience*.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209.

McEliece, R. J. (1978). A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116.

Misoczki, R., Tillich, J. P., Sendrier, N., and Barreto, P. S. L. M. (2013). MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes. 2013 IEEE International Symposium on Information Theory, pages 2069–2073.

Moody, D. (2016). Post-quantum cryptography: NIST's plan for the future. Talk given at PQCrypto. `https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf`.

NIST (2016). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. NIST web page. `http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf`.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134.

Shoup, V. (2003). Number Theory C++ Library (NTL).