Evaluation of Lattice-Based Signature Schemes in Embedded Systems

Tim Güneysu
Ruhr-Universität Bochum & DFKI
Bochum, Germany
tim.gueneysu@rub.de

Markus Krausz
Ruhr-Universität Bochum
Bochum, Germany
markus.krausz@rub.de

Tobias Oder
Ruhr-Universität Bochum
Bochum, Germany
tobias.oder@rub.de

Julian Speith
Ruhr-Universität Bochum
Bochum, Germany
julian.speith@rub.de

***Abstract*—Lattice-based cryptography is a promising candidate and remedy in public-key cryptography in case quantum computers become feasible or a major breakthrough in solving the factorization problem or the discrete logarithm problem is achieved. Due to ongoing research in this field, many schemes still lack implementations that examine their practicability, especially for embedded systems. In this work we discuss the potential of lattice-based signature schemes for practical applications on constrained devices in a post-quantum era. We focus on the schemes GLP, BLISS, and Dilithium and discuss their unique properties as well as challenges regarding their implementation on embedded devices. In this regard we present and review optimized implementations of these schemes for ARM Cortex-M4 microcontrollers to evaluate the practical performance of the schemes.**

***Index Terms*—lattices, signatures, microcontroller, dilithium**

I. Introduction

Digital signatures are a fundamental asymmetric cryptographic primitive with numerous applications in our daily life. Every digital transaction that requires authenticity and integrity requires a digital signature over the message or document. Currently deployed signature schemes, like RSA, DSA, or elliptic curve-based solutions, base their security on either the factorization problem or the discrete logarithm problem. Ever since Peter Shor developed a quantum algorithm that is able to solve both problems in polynomial time in 1997 [20], quantum-secure alternatives have gained a lot of attention.

Classical signature schemes are present on millions of devices in the Internet of Things. Regarding the fact that these devices are often deployed in the field for years, a transition to post-quantum cryptography must take place in the foreseeable future. For this transition it is essential to evaluate and assess the suitability of post-quantum signature scheme for contemporary embedded applications.

The National Institute of Standards and Technology even started a standardization process for post-quantum cryptography that also takes into account how the submitted schemes perform on a large number of target platforms. Among the NIST submissions, four major classes of quantum-secure mathematical problems emerged: based on hash functions, on linear codes, on lattices, and on multivariate quadratics. In this work we will focus on signature schemes based on lattices. Lattice-based schemes offer comparatively high performance, reasonable parameter sizes, and the versatility to also instantiate advanced constructions, like identity-based encryption. In general, there are two approaches to lattice-based signatures. The first one is the hash-and-sign approach [11], [12] that requires a trapdoor sampler, and the Fiat-Shamir approach [8], [9], [13] that turns a zero-knowledge proof into a signature scheme. In this work we discuss the typical implementation challenges of lattice-based signature schemes and evaluate selected schemes on a ARM Cortex-M4F microcontroller platform or review existing microcontroller implementations of these schemes. As Fiat-Shamir signature schemes usually are more efficient than hash-and-sign signature schemes and therefore better suited for an implementation on a microcontroller device, we will focus on the Fiat-Shamir schemes GLP [13], BLISS [8], and Dilithium [9].

A. Contribution

In this work, we provide a comprehensive survey of the three signature schemes GLP, BLISS, and Dilithium. We analyze the evolution of lattice-based signature schemes and highlight unique properties of each scheme. We present the up to our knowledge first ARM Cortex-M implementation of the GLP signature scheme and present an optimized implementation of Dilithium. To complete the picture we review implementations of BLISS and its variant BLISS-B to show the different requirements that these schemes have regarding the target platform.

B. Related Work

The GLP scheme has been presented in [13] and an optimized AVX implementation of the scheme can be found in [14]. BLISS [8] and its improvement BLISS-B [7] have microcontroller implementations on AVR [15] and ARM Cortex-M4 [19], [21]. Dilithium has been published at TCHES'18 [10] and has been submitted to the NIST standardization process [9].

II. Evoluation of Lattice-based Signature Schemes

In this section, we review the discussed schemes, GLP [13], BLISS [7], [8], and Dilithium [9]. For the sake of simplicity, we present high-level descriptions of these schemes and refer

[0]This work was partially funded by the European Union H2020 SAFEcrypto project (grant no. 644729).

to the respective original publications for full details and security proofs.

A. Notation

Polynomials in $\mathcal{R}_q = \mathbb{Z}_q[\mathbf{x}]/\langle x^n + 1\rangle$ are labeled by bold lower case letters. Bold upper case letters denote matrices. The Gaussian distribution with standard deviation σ is denoted by D_σ.

B. Signature Scheme: GLP

The GLP scheme is an extension of the work by Lyubashevsky [16], [17]. The private key is made of two polynomials $\mathbf{s}_1, \mathbf{s}_2$ with random ternary coefficients, i.e. coefficients in $\{-1, 0, 1\}$. The public key consists of a globally constant polynomial $\mathbf{a}$ and another polynomial $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$. The signing and verification procedures are sketched in Algorithms 1 and 2. Two masking polynomials $\mathbf{y}_1, \mathbf{y}_2$ are sampled that are used to hide the secret key in the signature. The message together with the masking polynomials and the global constant $\mathbf{a}$ is given as input to a random oracle (typically instantiated by a hash function) as required by the Fiat-Shamir transform. If the coefficients of the signature polynomials $\mathbf{z}_1, \mathbf{z}_2$ exceed a certain threshold (defined by the parameter k), the signature is rejected. The verification checks the validity of the signature with the help of the public key. The proposed parameters provide a security of 80 bits pre-quantum security [14].

Algorithm 1: GLP SIGNING ALGORITHM

Input: $\mathbf{s}_1, \mathbf{s}_2 \in [-1, 1]^n$, message $m \in \{0,1\}^*$
Output: $\mathbf{z}_1, \mathbf{z}_2 \in [-(k-32), k-32]^n$, $c \in \{0,1\}^{160}$

1 $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} [-k, k]^n$
2 $\mathbf{c} \leftarrow \mathrm{H}(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2,\ m)$
3 $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1,\ \mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
4 **if** $\mathbf{z}_1$ *or* $\mathbf{z}_2 \notin [-(k-32), k-32]^n$ **then**
5 go to step 1

Algorithm 2: GLP VERIFICATION ALGORITHM

Input: $z_1, z_2^{'} \in [-(k-32), k-32]^n$, $t \in \mathcal{R}_q$, $c \in \{0,1\}^{160}$, message $m \in \{0,1\}^*$
Output: Accept (valid) or Reject (invalid)

1 Accept if
2 $\mathbf{z}_1, \mathbf{z}_2 \in [-(k-32), k-32]^n$ and
3 $\mathbf{c} = \mathrm{H}(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c},\ m)$

C. Signature Scheme: BLISS

With the objective to reduce the signature size further, Ducas et al. developed the signature scheme BLISS [8]. One major difference to GLP is that the masking polynomial $\mathbf{y}$ is distributed according to a Gaussian distribution instead of being uniformly distributed in a small interval. Furthermore the rejection step is performed according to a bimodal Gaussian. This can be seen in the choice of the random bit b that determines whether $\mathbf{Sc}$ is added to or subtracted from $\mathbf{y}$. The changes in BLISS lead to a signature size of only 5.6 kbit instead of 9.0 kbit for GLP. Another improvement is that in GLP, there are on average seven repetitions for each signing attempt while BLISS has a rejection rate of only 1.6. Signing and verification of BLISS is sketched in Algorithms 3 and 4.

Algorithm 3: BLISS SIGNING ALGORITHM

Input: Key pair $(\mathbf{A}, \mathbf{S})$ such that $\mathbf{AS} = q \bmod 2q$, message $m \in \{0,1\}^*$
Output: $\mathbf{z}, \mathbf{c}$

1 $\mathbf{y} \xleftarrow{\$} D_\sigma^m$
2 $\mathbf{c} \leftarrow \mathrm{H}(\mathbf{A}\mathbf{y} \bmod 2q,\ m)$
3 Choose random bit $b \in \{0,1\}$
4 $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b\mathbf{Sc}$
5 Continue with a probability $1/\left(M \exp\left(-\frac{||\mathbf{Sc}||^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)\right)$ otherwise **restart**

Algorithm 4: BLISS VERIFICATION ALGORITHM

Input: Public key $\mathbf{A}$, signature $\mathbf{z}$, $c \in \{0,1\}^{160}$, message $m \in \{0,1\}^*$
Output: Accept (valid) or Reject (invalid)

1 Accept if
2 $\mathbf{z}$ does not exceed bounds and
3 $\mathbf{c} = \mathrm{H}(\mathbf{A}\mathbf{z} + q\mathbf{c} \bmod 2q,\ m)$

There is also an improvement to BLISS called BLISS-B [7] that has an improved key generation and achieves a rejection rate of 1.4 for signing for the same security level of 128 bits of pre-quantum security.

D. Signature Scheme: Dilithium

One major property of GLP and BLISS is that they are based on *ideal* lattices which means that the underlying lattice is structured. To this day, there is no attack known that exploits this structure and is more efficient than other attacks but it is not clear if there might be such an attack in the future. Therefore the designers of Dilithium made a more conservative choice and select *module* lattices as security foundation of their scheme. In ideal lattices all arithmetic is carried out on polynomials. Module lattices rather consist of (small) matrices in which the matrix elements are polynomials. Obviously, this results in larger parameter sizes and slower computations. Algorithms 5 and 6 show the signing and verification of Dilithium. Table I provides a comparison of signature and key size for the three discussed schemes. We compare the instantiations of the schemes that were implemented. The Roman numeral behind the name of the scheme in Table I refers to the parameter set that was implemented.

III. IMPLEMENTATION

Implementing a lattice-based signature scheme on embedded devices poses a number of challenges to the developer. All

Algorithm 5: DILITHIUM SIGNING ALGORITHM

Input: Key pair $((\mathbf{A}, \mathbf{t}), (\mathbf{s}1, \mathbf{s}_2))$, message $m \in \{0,1\}^*$
Output: Signature $\mathbf{z}, c$

$\mathbf{z} := \perp$
while $z = \perp$ **do**
  $\mathbf{y} \leftarrow\in [-(\gamma-1), \gamma-1]^(n \cdot l)$
  $\mathbf{w}_1 :=$ HighBits($\mathbf{Ay}$)
  $\mathbf{c} :=$H($\mathbf{w}_1$, m)
  $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
  if *z or lower bits of* ***Ay*** *exceed bounds* **then**
    $\mathbf{z} := \perp$

Algorithm 6: DILITHIUM VERIFICATION ALGORITHM

Input: Public key $(\mathbf{A}, \mathbf{t})$, signature $\mathbf{z}, c$, message $m \in \{0,1\}^*$
Output: Accept (valid) or Reject (invalid)

$\mathbf{w}'_1 :=$ HighBits($\mathbf{Az} - c\mathbf{t}$)
Accept if
$\mathbf{z}$ does not exceed bounds and
$\mathbf{c} =$H($\mathbf{w}'_1$, m)

discussed schemes have in common that their most important operation is polynomial multiplication. This can be efficiently realized with the number-theoretic transform (NTT) that lowers the complexity for polynomial multiplication from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. By applying a series of so-called butterfly operations to the polynomial it gets shifted from the time domain to the frequency domain in which polynomial multiplication is simple point-wise multiplication of the coefficients.

Within the NTT, modular arithmetic is performed. The choice of a suitable reduction algorithm has a major impact on the performance of the entire implementation of the scheme. Sometimes it even makes sense to combine different reduction technique as one could be faster for reducing the result of an addition or subtraction and the other is better suited to reduce the result of a multiplication. In [2] Alkim et al. apply such a combination of the Montgomery reduction [18] and the Barrett reduction [3] to the NTT. Depending on the parameters and the target architecture, *lazy* reduction has to be considered. In this context, *lazy* means that the reduction is not performed after each operation, but only when a result is expected to be too big to be stored in a single register.

All schemes require to draw samples from some kind of distribution. For GLP and Dilithium the sampler is uniform and therefore rather easy to implement but BLISS requires sampling from a Gaussian distribution. A large number of samplers exists that have different properties. Picking a suitable sampler is a time-memory trade-off. See [19] for a comparative analysis of different sampling algorithms for BLISS.

TABLE I
COMPARISON OF SELECTED INSTANTIATIONS OF GLP, BLISS, AND DILITHIUM. SIZES ARE GIVEN IN BITS.

Scheme	Security	Signature	Secret key	Public key
GLP I	80 pre-q	8,950	1,620	11,800
BLISS I	128 pre-q	5,600	2,000	7,000
BLISS-B III	160 pre-q	6,000	3,000	7,000
Dilithium III	128 post-q	21,608	28,032	11,776

IV. MICROCONTROLLER EVALUATION

To evaluate the practical performance of the discussed schemes, we compare their implementations on an ARM Cortex-M4F microcontroller. Our target platform is the STM32F4DISCOVERY board that runs at 168 MHz. To have a common evaluation framework for all implementations, we make use of pqm4 [1]. In the pqm4 framework, the running time of an operation is measured in cycle counts using libopencm3[1]. The framework can also measure the stack usage with the help of stack canaries.

The AVX implementation of GLP from [14] is the basis for our ARM implementation of GLP. Note that the implementation from [14] works on the data type double, we therefore changed the data type to 32-bit integers what is the native data type for a 32-bit platform. For Dilithium we optimized the NTT of the reference implementation with assembly instructions to make it more efficient on the microcontroller. In particular we merged two of the eight stages of the NTT to reduce memory accesses. To be able to compare all implementations in the same setting, we also incorporated the BLISS implementation from [19] and the BLISS-B implementation from [21] into the pqm4 framework.

Table II shows the cycle counts of the implementations and in Table III the stack usage is listed. When comparing the performance numbers, one has to keep in mind that the schemes provide different security levels (see Table I). The first thing to note is that the key generation of BLISS is two orders of magnitude slower than the key generations of other schemes. This is due to rejections that happen during the BLISS key generation. In BLISS-B this rejection step has been removed what leads to a massive speed-up. BLISS-B also has a very fast verification with less than 300k cycles. The performance of Dilithium is worse than the performance of BLISS. This however comes to no surprise as the security properties of Dilithium are much more robust. In terms of stack usage, we again observe that BLISS-B needs the least amount of dynamic memory. The much higher memory requirements of Dilithium are again due to the underlying module lattice assumptions.

V. CONCLUSION

In this work, we illustrated the evolution of lattice-based signature schemes and their fitness to microcontroller applications. While the design considerations of earlier schemes were mostly driven by efficiency considerations, Dilithium

[1] http://libopencm3.org/

TABLE II
CYCLE COUNTS FOR OUR MICROCONTROLLER IMPLEMENTATION MEASURED AT A CLOCK FREQUENCY OF 168 MHZ.

Operation	Key Gen	Sign	Verify
GLP I	1,995,981	16,170,642	2,160,841
BLISS I	229,718,709	4,648,240	539,253
BLISS-B III	259,185	5,447,332	292,762
Dilithium III	2,320,362	8,348,349	2,342,191

TABLE III
STACK USAGE IN BYTES FOR OUR MICROCONTROLLER IMPLEMENTATION.

Operation	Key Gen	Sign	Verify
GLP I	12,520	23,432	17,336
BLISS I	19,560	20,340	11,628
BLISS-B III	1,168	7,512	2,816
Dilithium III	50,488	86,568	54,800

follows a more conservative approach. With regard to the NIST post-quantum standardization process, Dilithium has been designed to offer robust security and sacrifices some performance for that. While BLISS is more efficient in terms of memory consumption and cycle counts, Dilithium offers a security analysis that takes into account a potential quantum attacker and is not as much affected by a potential attack on the structure of ideal lattices as BLISS would be.

An important issue for microcontroller implementations is the protection against side-channels. The implementations compared in this work do not claim protection or countermeasures to side-channel attacks as such a discussion would go beyond the scope of this paper. For an extended discussion of side-channel attacks on lattice-based cryptographic constructions we refer to the reader to works such as [4]–[6].

REFERENCES

[1] pqm4 - post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4. Accessed: 2018-06-27.

[2] Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. Newhope on ARM Cortex-M. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 332–349. Springer, 2016.

[3] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In Andrew M. Odlyzko, editor, *CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.

[4] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 354–384. Springer, 2018.

[5] Nina Bindel, Juliane Krämer, and Johannes Schreiber. Hampering fault attacks against lattice-based signature schemes: countermeasures and their efficiency (special session). In *Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion, CODES+ISSS*, pages 8:1–8:3. ACM, 2017.

[6] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 323–345. Springer, 2016.

[7] Léo Ducas. Accelerating bliss: the geometry of ternary polynomials. *IACR Cryptology ePrint Archive*, 2014:874, 2014.

[8] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013.

[9] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRSTALYS - Dilithium. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[10] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[11] 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.

[12] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

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

[14] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In Philippe Gaborit, editor, *Post-Quantum Cryptography, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, volume 7932 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2013.

[15] Zhe Liu, Thomas Pöppelmann, Tobias Oder, Hwajeong Seo, Sujoy Sinha Roy, Tim Güneysu, Johann Großschädl, Howon Kim, and Ingrid Verbauwhede. High-performance ideal lattice-based cryptography on 8-bit AVR microcontrollers. *ACM Trans. Embedded Comput. Syst.*, 16(4):117:1–117:24, 2017.

[16] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[17] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

[18] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.

[19] Tobias Oder, Thomas Pöppelmann, and Tim Güneysu. Beyond ECDSA and RSA: lattice-based digital signatures on constrained devices. In *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 110:1–110:6. ACM, 2014.

[20] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[21] Julian Speith, Tobias Oder, Marcel Kneib, and Tim Güneysu. A lattice-based AKE on ARM Cortex-M4. In *BalkanCryptSec 2018*, 2018.