Precomputation Methods for Faster and Greener Post-Quantum Cryptography on Emerging Embedded Platforms

Aydin Aysu, Patrick Schaumont

Secure Embedded Systems
Center for Embedded Systems for Critical Applications
Bradley Department of ECE
Virginia Tech, Blacksburg, VA 24061, USA
{aydinay,schaum}@vt.edu

Abstract. Precomputation techniques are useful to improve real-time performance of complex algorithms at the expense of extra memory, and extra preparatory computations. This practice is neglected especially in the embedded context where energy and memory space is limited. Instead, the embedded space favors the immediate reduction of energy and memory footprint. However, the embedded platforms of the future may be different from the traditional ones. Energy-harvesting sensor nodes may extract virtually limitless energy from their surrounding, while at the same time they are able to store more data at cheaper cost, thanks to Moore's law. Yet, minimizing the run-time energy and latency will still be primary targets for today's as well as future real-time embedded systems. Another important challenge for the future systems is to provide efficient public-key based solutions that can thwart quantum-cryptanalysis. In this article, we address these two concepts. We apply precomputation techniques on two post-quantum digital signature schemes: hash-based and lattice-based digital signatures. We first demonstrate that precomputation methods are extensible to post-quantum cryptography and are applicable on current energy-harvesting platforms. Then, we quantify its impact on energy, execution time, and the overall system yield. The results show that precomputation can improve the run-time latency and energy consumption up to a factor of 82.7× and 11.8×, respectively. Moreover, for a typical energy-harvesting profile, it can triple the total number of generated signatures. We reveal that precomputation enables very complex and even probabilistic algorithms to achieve acceptable real-time performance on resource-constrained platforms. Thus, it will expand the scope of post-quantum algorithms to a broader range of platforms and applications.

Keywords: Precomputation, Post-Quantum Signatures, Hash-based Signatures, Lattice-based Signatures, Energy Harvesting Platforms

1 Introduction

Digital signatures are arguably the most important public-key cryptographic primitive. We rely heavily on these signatures to authenticate critical electronic data such as identity information on e-passports, quantity, source, and destination of financial transactions, consumption amount and time of smart-meters, and enterprise names on software distribution. Even tough these applications use well-established standard tools like ECDSA and RSA on a daily basis, recent advances in cryptanalysis and quantum-computers motivate new pillars for the future of our digital security.

Traditional cryptography, and classic public-key cryptography in particular, faces an increasing risk at a catastrophic event because of improvements in quantum computing architectures, and because of continuous progress in the cryptanalysis of traditional public-key algorithms. Table 1 highlights the impact of quantum algorithms on the security of fundamental cryptographic constructions. Grover's algorithm enables a fast database search and hence reduces the security of an n-bit key to $n/2$-bits [23]. Likewise, quantum birthday attacks reduce a collision search complexity from $\mathcal{O}(n/2)$ to $\mathcal{O}(n/3)$ [11]. While these security reductions indicate that quantum computers will affect symmetric key and hash-based constructions, we can still assure pre-quantum security levels by simply doubling and tripling the key size and the hash output respectively. The case for public-key encryption is much worse. Shor's algorithm [45] can solve the factorization and the (elliptic curve) discrete logarithm problem in polylogarithmic time [41]. Hence, we have to increase the key size exponentially which is infeasible in practice. This is the main motivation of post-quantum public-key constructions. We need practical public-key building blocks for the post-quantum era.

Table 1. Security reductions of pre- and post-quantum era

Operation	Hash Function	Symmetric Key Encryption	Public Key Encryption
Standard	SHA-2	AES	ECC
Target Security	n	n	n
Pre-Quantum Security	$n/2$	n	$n/2$
Post-Quantum Security	$n/3$	$n/2$	$(\mathbf{logN})^{\mathbf{3}}$

1.1 Precomputation as an Emerging Topic in Cryptographic Computing

In the post-quantum era, not only the security primitives will change but also the paradigm of computing for the embedded systems. In this paper, we argue that precomputation, an old practice that is typically overlooked in the embedded domains, will be a re-trending phenomena for the emerging computing systems. Precomputation in cryptography was previously proposed to accelerate exponentiation or elliptic curve multiplication [29], [12], [44], [10]. A major disadvantage of precomputation is that it usually requires more computation (energy) and storage, two resources that are constrained in traditional embedded systems. However, these assumptions are changing. The technology of flash memory, the predominant storage unit of embedded domains, introduced 15 new generations of products over the last 20 years, accumulating to a cost improvement of 25,000× [26]. This trend will make the integration of more capable storage units increasingly cheaper. On the other hand, although the Moore's law does not apply to the battery technologies, energy-harvesting platforms make energy no longer a limited and monotonically decreasing concept.

Evidently, this is not the first research on energy optimization for harvesting nodes. Previous work like *Dewdrop* [14] and *DEOS* [49] propose to relabel iterative operations as atomic tasks and then to (dynamically) schedule them for maximum computations with the available energy. In contrast, we leverage our application specific expertise to transform algorithms into a set of divisible tasks. This transformation enables optimizations like partial evaluation with energy-aware partitions; the system precomputes and stores static values at energy-friendly intervals, and minimizes the run-time energy and latency. *Mementos* also divides the atomic operations but it aims to allocate checkpoints within a task to quickly restart in case of power failure or energy depletion [42]. The techniques we apply in this paper are orthogonal, they can be implemented on top of the previous work.

Ateniese *et al.* makes similar claims on precomputation for wireless sensor nodes implementing pre-quantum primitives like ECDSA [3]. They show that a mote can precompute intermediate values when there is an (excess) energy available and then use it to minimize the latency. We extend these methodologies for the post-quantum era. We show that two post-quantum signature schemes, lattice-based and hash-based digital signatures can be accelerated by precomputation techniques. Then, we quantify the savings of this methodology in terms of latency, energy, and system yield.

1.2 Organization

The rest of the paper is organized as follows. Section 2 motivates an application scenario and the benefits of partitioning the computational modes. Section 3 introduces the post-quantum digital signature schemes and the applied precomputation methods. Section 4 describes the target platform. Section 5 reports the implementation results. Section 6 highlights related implementations and section 7 concludes the paper.

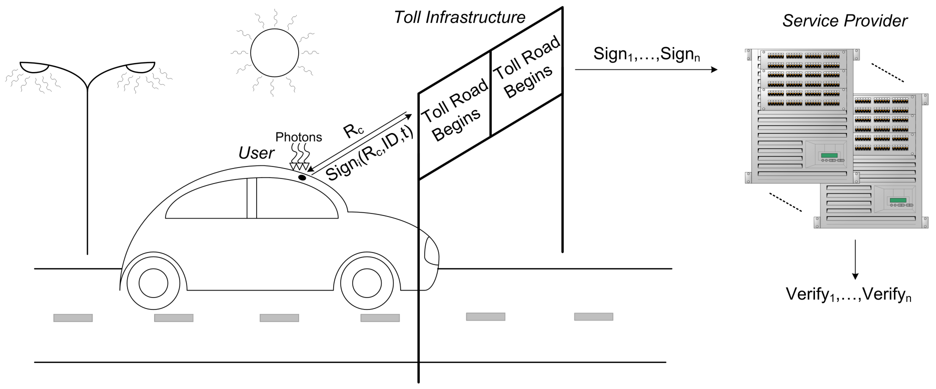

Fig. 1. The operations of a road tolling authentication system with digital signatures

Table 2. The asymmetry of the two computing devices

Computing device	Center of the cloud (Servers)	Edge of the cloud (Portable embedded nodes)
Operation	Signature verification	Signature generation
Platform	High-end CPUs	Simple microcontrollers
Rate	1000 verifications per min.	1 signing per hour
Optimization	**Throughput**	**Latency**

2 Post-Quantum World

In this section, we will conceptualize a scenario for the post-quantum era and later argue why we consider precomputation as a very suitable optimization methodology for these type of applications.

Figure 1 shows a road tolling system, a classic application of digital signatures using embedded platforms. These systems authenticate motor vehicles as they drive past a toll booth. The authentication protocol uses digital signatures. The toll infrastructure sends a random challenge (R_c) and the on-board unit replies with a digital signature on the challenge, identification attributes, and a time stamp. The toll infrastructure collects and stores these signatures at fine grain intervals (as the car passes by). At coarse grain intervals (eg. at the end of the day), it transmits all signatures to the service provider which is responsible for verifying the signatures and charging the associated users.

While most contemporary toll roads use a toll booth infrastructure, future tolling mechanisms will require automatic real-time handling of traffic with minimal interruption of traffic flow. Moreover, the future cars could also incorporate Vehicle-to-Infrastructure backbones which can enable a myriad of applications [48]. Therefore, we claim that even if the security requirements and the em-

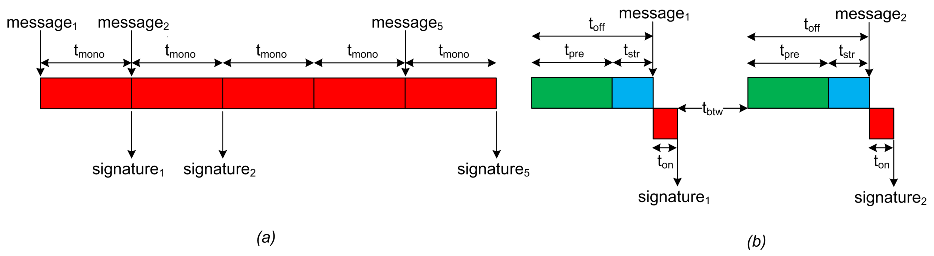

Fig. 2. Principle of operation for the offline phase (a) and the online phase (b)

bedded platforms of the future may change, generating signatures on-demand with minimal latency and energy will still be of primary importance for many applications.

Table 2 summarizes the differences of the computing devices for the target scenario. The cloud server consists of a sea of high-end CPUs that can verify thousands of signatures per minute. Operations within the cloud are streamlined and the executions are optimized for throughput, to maximize the number of operations per unit time. In contrast, the edge of the cloud is a portable embedded node that is typically a low-end microcontroller. These nodes become rarely active (eg. once per hour) but they are real-time, hence they are optimized for latency, to minimize the execution time per signing.

2.1 The need for partitioning and precomputation

The nature of real-time applications allows partitioning computations into offline and online phases. The offline phase refers to the workload that can be handled before an operation request comes. In our context, it corresponds to all computations that can be completed without full knowledge of the message to sign; it includes operations like accumulating an entropy pool, generating keys, selecting random numbers, and computing message independent variables. Online computations include all operations that have an immediate dependency on the message. These operations appear with the real-time signing request. The system then transitions to the online phase and quickly generates the output with precomputed coupons.

Figure 2 shows the principle of operation for the monolithic case (Figure 2(a)) vs. a partitioned one (Figure 2(b)). Typically, the total execution time of the monolithic mode (t_{mono}) is shorter than the partitioned mode ($t_{off} + t_{on}$), making it more suitable for throughput optimization. Therefore, the servers usually follow a monolithic mode as they constantly execute a streamline of operations. On the other hand, the embedded nodes can utilize the partitioning to minimize the execution time if the operations are partitioned such that $t_{on} \leq t_{mono}$. The total time of the offline phase t_{off} is the time it takes to precompute (t_{pre}) and store (t_{str}) these variables. For simplicity, here we assume that the execution of precomputation and storage are sequential but in practice they may also be interleaved. We also assume that t_{off} is not critical and the time between two consecutive execution is long ($t_{btw} \geq 0$). Indeed, under those assumptions, the platform can seek energy-efficient solutions, either wait for a pre-configured amount of harvested energy to start an uninterrupted execution [33] or start precomputations when the system reached its maximum energy level[8].

3 Post-Quantum Signatures

Post-Quantum signatures are classified into four groups: hash-based signatures, lattice-based signatures, code-based signatures and multivariate quadratic signatures. In this section, we will give an overview of all these digital signature schemes, but we will only investigate hash-based and lattice-based signatures in more detail. All these signature schemes are promising candidates in the post-quantum era, but the hash-based and the lattice-based signatures have relatively slower execution time which we can accelerate with precomputation methods.

3.1 Hash-based Signatures

Hash-based constructions provide forward-secure signing schemes. Moreover, depending on the underlying hash function, they can also be provably secure even in the post-quantum era [32]. In this work, we implement the Winternitz one-time signature (W-OTS) scheme defined as in [17]. Compared to Lamport-Diffie signatures [30], Winternitz offers a trade-off between signature size (communication energy) and execution time (computation energy). We have used the standard SHA-256 because it is fast and it provides 84-bits and 128-bits post-quantum security of pre-image and collision resistance respectively. Any secure hash function that has at least 256-bit outputs (eg. Keccak) can also replace the SHA-256 in our implementation.

In the Winternitz signature scheme, one generates a hash chain $Y = h(h(h(..h(X))))$ with the secret key X and the public key Y. A signature of message μ is created by selecting intermediate values on the hash chain using portions of μ. The verifier can, using μ, check if those intermediate values indeed reveal the public key Y at the end of the chain. The one-way character of the hash protects the secret key. Hash-based signatures are one-time signatures, since disclosing hash chain is equivalent to disclosing the secret key. In what follows, we derive a formal definition of these operations.

The key idea behind the Winternitz is to sign ω-bit portions of the message simultaneously. Hence, the Winternitz parameter $\omega \geq 2$ determines the trade-off between the execution time and the signature size. The number of required hash computations grow exponentially while the signature and key size reduce linearly. The parameters t_1, t_2, and t of the Winternitz signature are defined as

$$t_1 = \left\lceil \frac{n}{\omega} \right\rceil, t_2 = \left\lceil \frac{\lfloor \log_2 t_1 + 1 + \omega \rfloor}{\omega} \right\rceil, t = t_1 + t_2 \qquad (1)$$

The secret key X is t-blocks of 256-bit hash output values $(x_{t-1}, ..., x_1, x_0)$ generated from a single random seed and the public key Y is t blocks of 256-bit hash output values $(y_{t-1}, ..., y_1, y_0)$ generated using hash chains of $2^w - 1$ length where $y_i = h^{2^w-1}(x_i), 0 \leq i \leq t-1$. The scheme divides the message (or its hash) into t_1 blocks of ω-bits $(b_{t-1}, ..., b_{t-t_1})$ and also computes the checksum of t_2 blocks of ω-bits $(b_{t_2-1}, ..., b_0)$. Then, depending on the values of these blocks (from 0 to $2^w - 1$), it calculates the signature σ $(s_{t-1}, ..., s_1, s_0)$ with hash chains ranging from 0 to $2^w - 1$ as

$$\sigma = (s_{t-1}, ..., s_1, s_0) = (h^{b_{t-1}}(x_{t-1}), ..., h^{b_1}(x_1), h^{b_0}(x_0)) \tag{2}$$

To verify the signature σ, one has to first use the message (or its hash) to compute the values of t_1 and t_2 blocks of ω-bits $(b_{t-1}, ..., b_0)$. Then, if the signature is valid, applying the remaining hash chains of $2^\omega - 1 - b_i$ $(0 \leq i \leq t-1)$ on the signature σ should reveal the public key Y $(y_{t-1}, ..., y_1, y_0)$.

Generating the secret and the public key requires t and $t(2^w - 1)$ hash computations respectively. The number of hash operations required to compute the signature depends on the message (or its hash) to be signed and is bounded with $[0,t(2^w - 1)]$. The length of the secret key, public key and the signature is $t \times n$ bits.

Standalone hash-based constructions offer one-time signatures that can only sign a single message. However, with Merkle trees [35] or Chaining, they can be further extended into schemes that can sign multiple messages. Merkle trees expand a one-time signature to a 2^H-time signature by using a hash-tree of height H. The root node of the hash-tree is a public value and the leaves are the one-time signatures. Together with each signature, the signer also sends an index value $(\leq 2^H)$ indicating the key position (leaf node), the path from the root node corresponding to that index, and the one-time public key. The verifier first checks if the index value is greater than the previous one and if the path is correct. Then, it validates the one-time signature.

Chaining is simply including the next public key to the message to be signed. One drawback of Chaining is that it enforces the verifier to validate signatures in an order. We opt to implement Chaining instead of Merkle tree as they both achieve similar signature size (even with the latest work on Merkle trees [13], [27], [28]) while Chaining does not limit the number of signature generations for a given device.

3.2 Applying Precomputation Methods for Hash-based Signatures

Precomputation can significantly accelerate the signature generation time of hash-based signatures. The main idea behind this optimization is to store some of the intermediate states of the hash chain during the offline phase and to start the chain from the closest possible node during the online phase. There is an obvious trade-off between the memory and the execution time of this operation. If there is enough memory space, we could precompute and store all intermediate

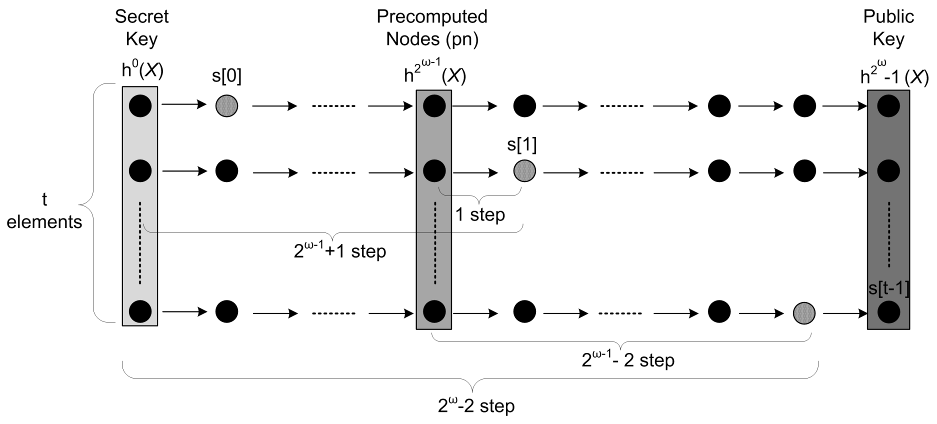

Fig. 3. Precomputation of intermediate nodes for the Winternitz scheme

states of the hash chain. Then, the online phase simply becomes selecting and loading the appropriate values from the memory.

Figure 3 illustrates an example scenario of how precomputation can accelerate the signature generation. If we assume that the entire hash-chain is generated online, then the computation of $s[0]$, $s[1]$, and $s[t-1]$ requires 1, $2^{\omega-1}+1$, and $2^{\omega}-2$ evaluations of the hash function respectively. If we precompute the intermediate hash-chain step of $h^{2^{\omega-1}}$, the number of evaluation steps for $x[1]$ and $x[t-1]$ shortens to 1 and $2^{\omega-1}-2$ respectively. The more intermediate steps we store, the shorter the number of evaluation steps become on average.

Figure 4 highlights the operation flow of several modes (strategies) for two consecutive hash-based signature executions. To compute the signature of a message, we have to generate the one-time secret key (sk_i) for the current session and the one-time public key (pk_{i+1}) for the next session (due to chaining). Therefore, the required operations of a single execution are: generating one-time keys, computing the signature of the message and the next public key, and transmitting those values to the verifier. The generic flow (Figure 4(a)) does not store any information between two consecutive runs. Hence, it recomputes the current keys for every session. This can be omitted with an optimized execution (Figure 4(b)) in which the processor stores the secret key of the next session. Now, before generating a signature, the processor can load the precomputed secret key and perform signing. Converting the monolithic execution mode to a partitioned mode (Figure 4(c), (d)) further reduces the latency. In the partitioned mode, the processor can precompute a number of one-time keys (say j keys) during the offline phase and later use it for the future signature requests. The processor can stay in the online phase as long as there is a precomputed key. Once the system uses all precomputed keys, it has to return back to the offline phase. This operation effectively removes the generation of the keys from the latency path, resulting in a shorter execution time. Finally, we can apply the precomputation of intermediate nodes (Figure 4(d)) to maximize the speed.

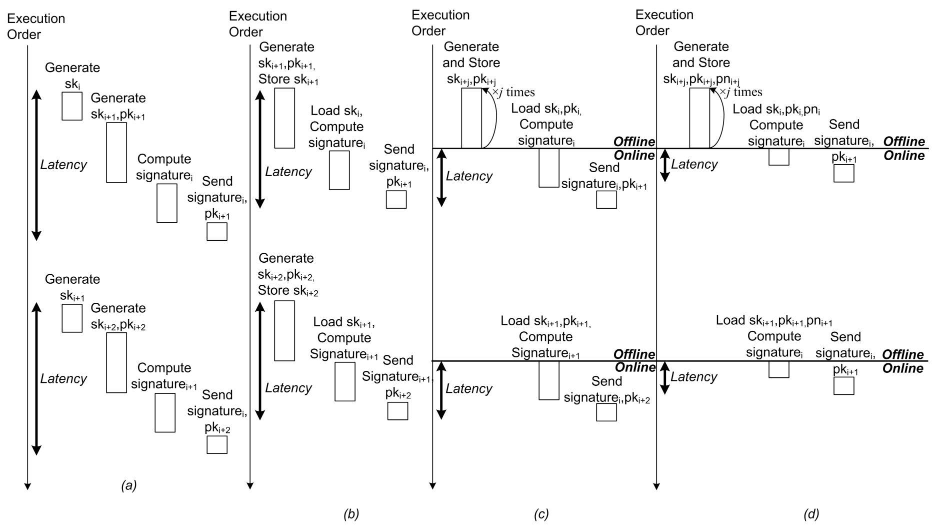
Execution Order
Generate sk_i
Generate sk_{i+1}, pk_{i+1}
Latency
Compute signature$_i$
Send signature$_i$, pk_{i+1}
Generate sk_{i+1}
Generate sk_{i+2}, pk_{i+2}
Compute signature$_{i+1}$
Send signature$_i$, pk_{i+2}
(a)
Generate sk_{i+1}, pk_{i+1}, Store sk_{i+1}
Load sk_i, Compute signature$_i$
Send signature$_i$, pk_{i+1}
Generate sk_{i+2}, pk_{i+2}, Store sk_{i+2}
Load sk_{i+1}, Compute Signature$_{i+1}$
Send Signature$_{i+1}$ pk_{i+2}
(b)
Generate and Store sk_{i+j}, pk_{i+j}
$\times j$ times
Load sk_i, pk_i, Compute signature$_i$
Offline
Online
Send signature$_i$, pk_{i+1}
Load sk_{i+1}, pk_{i+1}, Compute Signature$_{i+1}$
Send signature$_i$, pk_{i+2}
(c)
Generate and Store $sk_{i+j}, pk_{i+j}, pn_{i+j}$
Load sk_i, pk_i, pn_i Compute signature$_i$
Send signature$_i$, pk_{i+1}
Load $sk_{i+1}, pk_{i+1}, pn_{i+1}$ Compute signature$_i$
(d)

Fig. 4. Computation modes for (a) generic, (b) optimized, (c) partitioned, and (d) partitioned-with-precomputing execution

As usual, precomputation comes with a memory overhead. For the mode *(a)* it is enough just to store the root of the secret key which is 256-bits where as mode (b) requires $t \times 256$-bits of storage. As we precompute j secret and public keys, mode (c) needs $t \times j \times 256 \times 2$ amount of memory bits and if the system utilizes pn precomputed node blocks, then the memory cost of mode (d) becomes $t \times j \times 256 \times (2 + pn)$-bits.

3.3 Lattice-based Signatures

The family of lattice-based signatures is another promising group of candidates in the post-quantum era. Currently, lattice-based signatures that utilize Fiat-Shamir paradigm [21] yields the most efficient constructions. These constructions first introduce an identification scheme and then transform it to signatures.

The main challenge of lattice-based signatures is to minimize the signature size as well as public and secret keys. Figure 5 illustrates recent work on practical lattice-based digital signature schemes using the Fiat-Shamir transformation. Lyubashevski *et al.* proposed the majority of the body of work in this field and hence we will refer these schemes as Lyubashevski-like constructions. The figure shows that over a couple of years, using several optimization techniques such as, matrix to polynomial reduction [31], changing the basis of lattice problems [47], optimizing the parameter set [24], compression [24], [6], [5], and more efficient sampling [18] reduced the signature size down to 5Kb. This size will probably even become smaller in the future as there is currently no theoretical limit on how small they eventually be.

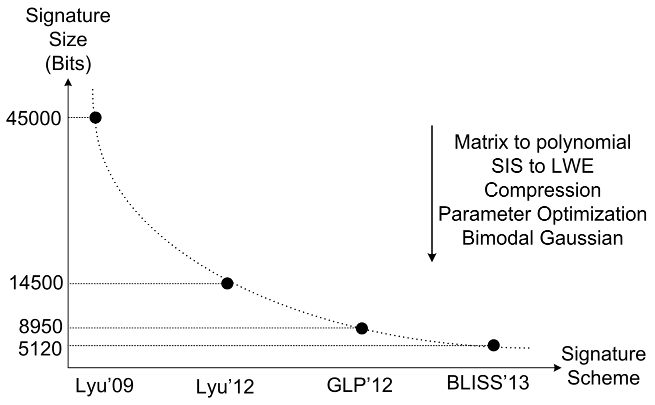

Fig. 5. Practical Lattice-based Digital Signature Schemes

In this paper, we will focus on the basic signature scheme presented in [24] due to its conceptual simplicity and reasonable signature size. However, the optimization methods that we propose apply to all Lyubashevski-like constructions and we can extrapolate our results to other lattice-based signature schemes as well.

Algorithm 1 The basic signature scheme of [24]

```
1: procedure KEY GENERATION($a, s_1, s_2, t$)
2:     $s_1, s_2 \leftarrow rand(R_1^{p^n})$
3:     $a \leftarrow rand(R^{p^n})$
4:     $t \leftarrow as_1 + s_2$
5: end procedure
6: procedure SIGNING($s_1, s_2, \mu, z_1, z_2, c$)
7:     $y_1, y_2 \leftarrow rand(R_k^{p^n})$
8:     $c \leftarrow H(ay_1 + y_2, \mu)$
9:     $z_1 \leftarrow s_1c + y_1, z_2 \leftarrow s_2c + y_2$
10:    if $z_1$ or $z_2 \notin R_{k-32}^{p^n}$ go to step 7
11: end procedure
12: procedure VERIFICATION($z_1, z_2, c, \mu, t,$)
13:    Validate iff
14:        $z_1,\ z_2 \in R_{k-32}^{p^n}$
15:        $c = H(az_1 + z_2 + tc, \mu)$
16: end procedure
```

The listing in 1 presents the basic signature scheme of Güneysu *et al.* [24]. We have implemented this signature scheme with the parameter Set I. All elements except the message μ are defined as polynomials of degree n and the coefficients of these polynomials are modulo p. The underlying operations of the signature

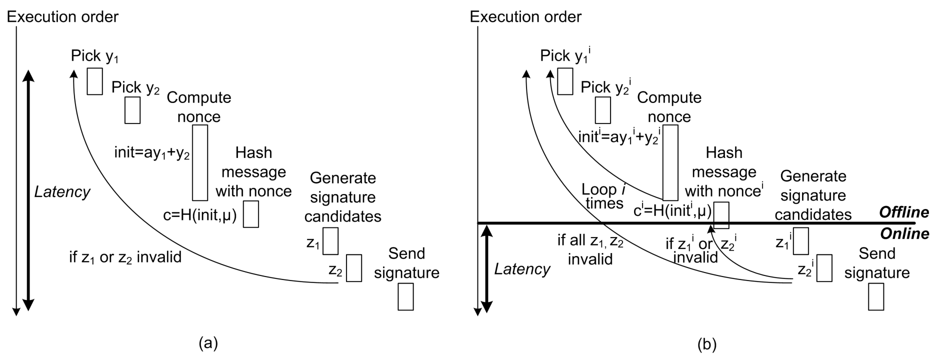

Fig. 6. Monolithic (a) and the partitioned (b) computation modes for the GLP signature scheme

scheme are picking uniformly random polynomials ($rand()$), polynomial multiplication, polynomial addition, and hashing (H).

Key Generation consists of picking the secret key s_1, s_2 with small coefficients (either -1,0 or 1), setting a parameter a, and generating the public key t. Then, the signer uses the unique masking polynomials y_1, y_2 and generates the nonce $ay_1 + y_2$. This value is concatenated with the message μ and hashed. Now, the signature z_1 and z_2 can be generated using the secret key s_1, s_2, the output of the hash (c) and the masking polynomials. If the coefficients of z_1 and z_2 are not bounded with $(-k, +k)$, the signing process has to restart by picking another two masking polynomials. $Verification$ starts with checking if the coefficients of the signature polynomials are bounded accordingly. Then, to validate a signature, the verifier can check if the equation $c = H(az_1 + z_2 + tc, \mu)$ holds.

3.4 Applying Precomputation Methods for Lattice-based Signatures

There are two different types of precomputation methods for lattice-based signatures. The first one refers to an arithmetic trick on the multiplicative operands during the Number Theoretic Transform (NTT) [2]. The previous work reports the effects of generating these operands on-the-fly vs. precomputing them on reconfigurable hardware [4]. In this work, we study its effects on microcontrollers. We will also discuss another precomputation method that requires latency-optimized algorithmic transformation for a partitioned execution.

Figure 6 illustrates the operation flow of monolithic *(a)* and partitioned execution *(b)*. To generate the signature, the monolithic execution first processes all steps of the signature scheme such as picking masking polynomials (y_1,y_2), calculating nonce values ($init$), hashing the output (c), and generating the signature candidate (z_1,z_2). Then, it checks if the candidate is valid and repeats the same process if necessary. In contrast, the partitioned execution selects a number of masking polynomials (y_1^i,y_2^i), then it computes and stores multiple nonce values. These coupons are used to create the signature during the online phase. If all attempts fail to generate a single valid signature, the signer has to start all over and compute new coupons. However, given enough memory space to store coupons, the probability of failure becomes arbitrarily small. The partition is especially useful for our application scenario because it outsources the most complicated operation, NTT-based multiplication of ay_1, to the offline phase. The online phase operations consist of simple operations like hash, sparse polynomial multiplication of sc_1, polynomial addition, and coefficient comparison.

3.5 Other Signature Schemes

Code-based and multivariate quadratic (MQ) signature schemes can also be used in the post-quantum era. Code-based McEllice encryptions follow the equation $x = mG \oplus e$. The public key is the generator matrix G of a t-bit error correcting code that encodes the message m and generates the signature x by adding a t-bit error vector e. The decoding process computes the syndrome s as $s = Hx^t$ using the secret parity matrix H. Then, the message can be recovered with bit-flipping algorithms [22] or using more complex decoding techniques [7]. The nature of the coding algorithms makes this family of crypto-systems very suitable for the application scenario; encoding is typically a simple process while decoding is much more complex. However, they do not additionally accelerate with precomputation as all the operations have dependency on the input message. Currently, the most efficient and secure code-based encryption method is a McEllice-based construction with a Quasi-Cyclic (QC) Moderate Density Parity Check (MDPC) code using 4801-bits of public and secret keys [36]. Recent work shows that encryption with QC-MDPC codes takes 42 ms on an ARM Cortex-M4 microcontroller [34]. The McEllice-based public key encryption can be transformed to a digital signature [15] but such an implementation utilizing the QC-MDPC is not available yet.

MQ signatures also have a fast signature generation rate. F is m quadratic polynomials of n variables, referred to as central map. The polynomials of F are chosen to make it easily invertible. However, the public key P is a hard-to-invert mapping computed as $P = T \circ F \circ G$ where T and G are two additional maps that obfuscates the structure of F. The secret keys are T, F, and G. To sign a message m, we evaluate it respectively over T^{-1}, F^{-1}, and G^{-1}. Hence, these operations depend on m and can not be precomputed without the full knowledge of the input message. Verification is done by checking if the evaluation holds over P. Typically, MQ signature generation and the verification is fast, but the public key sizes are impractical (over hundred KB). Recent work on MQ signature scheme reduces the public key size significantly to 9kB[38]. On an ATxMega128a1 microcontroller clocked at 32MHz, it takes 110.20 ms to compute such a signature [16]. However, one should approach these constructions with caution as previous optimizations with similar features have a track record of successful cryptanalysis [20], [46].

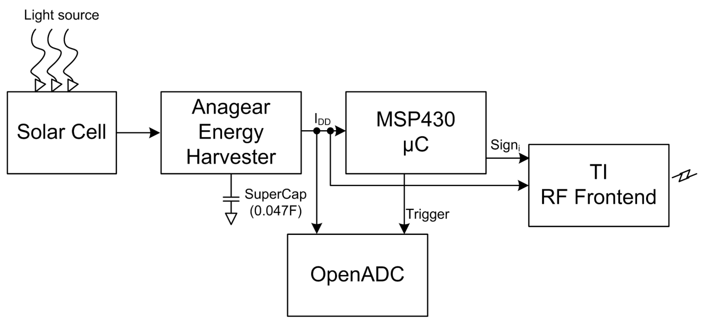

Fig. 7. The block diagram of the research platform

4 Platform

In this work, we used our research prototype with energy harvesting and measurement capabilities. This section gives a brief overview but we refer the interested readers to [37] for a detailed description of our complete setup. Figure 7 shows a block diagram of the building blocks of our system. A Photovoltaic cell converts the energy from photons into electricity and transmits it to the Anagear ANG 1010 energy-harvesting board. Attached to the Anagear, a low-leakage supercapacitor can accumulate an energy-level up to 710 mJ. This energy can activate the main processing unit of the system, a low-power 16-bit MSP microprocessor with 16KBs of SRAM and 128 KBs of Flash memory that executes the digital signature schemes. The RF Frontend of the system (CC2500 transceiver of Texas Instruments) also uses the harvested energy and it can send the signatures with 64-byte packets. The MSP microprocessor is also responsible to send trigger signals to the measurement unit, an OpenADC board with a Spartan-3 FPGA that can measure the energy consumption of both computation and communication with a high-precision.

5 Implementation Results

We used the C programming language to implement the digital signature schemes. We compiled the software codes with the `gcc 4.6.3` cross compiler with an optimization level of `O2` that minimizes the code size. The processor runs at 10 MHz and does not utilize a hardware multiplier. We used the RELIC 0.3.3 library [1] to realize the hash primitive and select the SHA-256 hash function as it provides an acceptable level of collision resistance (84-bits) in the post-quantum era. Even though the software-efficiency can potentially be improved with assembly level programming, we aim to show the relative savings of precomputation and argue that it will be similar also on an optimized implementation.

Here we summarize our conclusions from the implementation result. We derive these observations using our platform and the calibration factors of the results are depending on the experimental setup.

– Communication requires more energy per unit time than computation
– Precomputation can reduce the latency and the run-time energy consumption by an order of magnitude
– GLP is faster and requires less energy than Winternitz
– The parameter ω can be selected to optimize either execution time or energy-efficiency
– Post-Quantum signatures have comparable time/energy results with pre-quantum signatures
– Proposed methods extend the operation hours of the system and increase the number of signature generations for a given energy level

5.1 Impact of Proposed Methods on Single Executions

Figures 8 and 9 report the energy consumption and the execution time of the post-quantum signature schemes. The results separately show the costs of offline operations, online operations, and the signature transmission. For hash-based signatures, the run-time energy and the latency of the monolithic mode (strategy (b) in Section 3.2) is equal to the sum of all these operations where as the partitioned modes (strategy (c), and (d) in Section 3.2) do not include the offline phase. $pre = 7$, $pre = 3$, $pre = 1$, and $pre = 0$ correspond to chains with 7, 3, 1, and 0 precomputed nodes respectively (ommitting the first node, secret key). Ideally, we would like to store the complete hash chain but the size of the SRAM is 16KB and it can not contain more than 7 precomputed nodes for $\omega = 8$ and 3 precomputed nodes for $\omega = 4$. For $\omega = 8$ (resp., $\omega = 4$), the total run-time energy of strategy (b), (c), and (d) is 100.69 mJ (resp., 20.86 mJ), 32.16 mJ (resp., 12.05 mJ), and 8.5 mJ (resp., 9.54 mJ). The latency of these strategies are 52.29 s (resp., 6.98 s), 15.53 s (resp., 2.15 s), and 2.22 s (resp., 0.69 s).

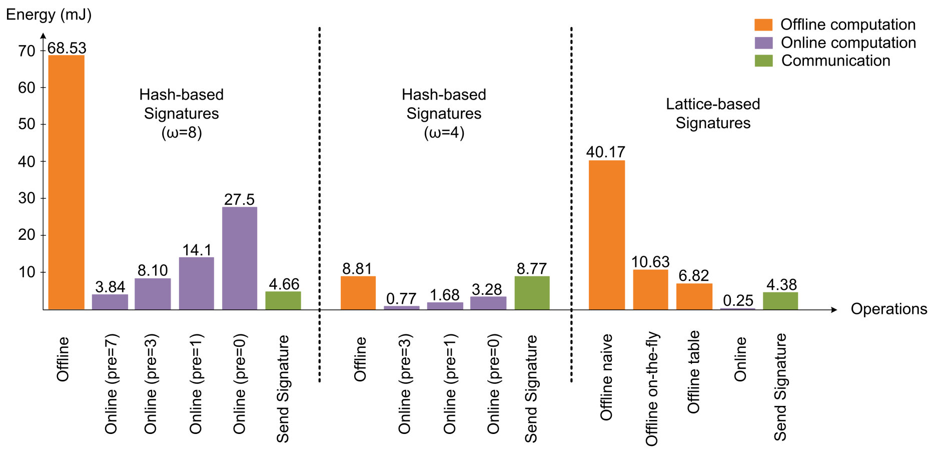

Fig. 8. The energy consumption measurements of signature schemes

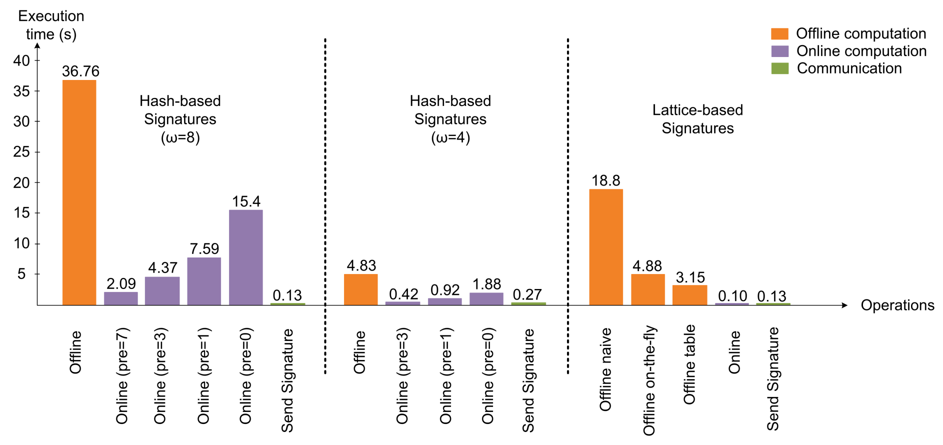

Fig. 9. The execution time measurements of signature schemes

As expected, the communication energy reduces linearly while the computation energy grows exponentially with ω. Likewise, the execution time and the energy of the online phase scales down linearly with the increase in the precomputed nodes. The experiments also reveal that the extra operations of strategy (d), which is storing intermediate nodes of the hash chains, require minimal calculations (approximately %1 increase) during the offline phase. Hence, we conclude that the cost of precomputation is marginal especially compared to its saving and given more memory space, the precomputation can further reduce the execution time and the energy of the online phase.

For lattice-based signatures, the run-time energy and latency of the partitioned execution does not include the offline phase, where as the monolithic execution involve all these operations. The energy consumption of the offline phase depends largely on how we perform the NTT operation. The naive implementation computes the powers of multiplicative operands of NTT from scratch at each iteration while the on-the-fly configuration uses the outer-loop iteration trick in [2]. $Offline$ *table* refers to computing and storing all required powers of the multiplicative operands during design-time, and later loading the appropriate values during NTT [40]. Even with an optimization of $6\times$, the offline phase of GLP signature requires a significant amount of energy and the execution time is several seconds. In contrast, the system can quickly calculate the online phase which brings a further optimization of $27\times$, making the total latency 230 ms. Likewise, the run-time energy drops down to 4.63 mJ. Note that the results show the computation of one signature candidate, the average number of trials depends on the parameter set. The parameters that we use requires a repetition rate of 7 on average but this number reduces to 1.6 with more recent constructions [18].

Table 3 summarizes the impact of the proposed methods for signature generation. On our experimental setup, generating and transmitting a signature candidate using GLP requires a minimum of 4.63 mJ and 0.23 s whereas it takes

Table 3. The best cases for the signature generations

Signature Scheme	**GLP**	**Winternitz** $\omega = 8$	**Winternitz** $\omega = 4$
Optimizations	Precomp. Operands Off/on partition	Precomp. Chain Nodes Off/on partition	
Offline phase (mJ/s)	6.82/3.15	68.53/36.76	8.81/4.83
Online phase (mJ/s)	0.25/0.10	3.84/2.09	0.77/0.42
Transmission (mJ/s)	4.38/0.13	4.66/0.13	8.77/0.27
Online Total (mJ/s)	**4.63/0.23**	**8.5**/2.22	9.54/**0.69**

8.5 mJ and 0.69 s with the best cases of Winternitz. With the available memory and the transmission infrastructure, hash-based signatures with $\omega = 8$ is more energy-efficient while $\omega = 4$ has a lower latency. BLISS improvements over GLP signatures enable a reduction in transmission energy/time while moving to a more sophisticated microprocessor enable a reduction in transmission and computation energy/time of Winternitz. Since a monolithic ECDSA signature operation takes 91 mJ and 12.5 s to compute on the same platform[37], we can conclude that post-quantum signatures are quite competitive, if not better suited for real-time applications with energy constraints.

5.2 Impact of the Proposed Methods on the System

Apart from latency and run-time energy optimizations of online operations with the precomputed coupons, we observed two advantages of using the precomputation with partitioned execution modes. First, it extends the availability of the system. Second, it provides more service (more signature) for the given energy level. In the remainder of this section, we will describe how we arrived to these conclusions.

The energy accumulated by the system depends on many factors such as the quality and the size of the photovoltaic cell and the supercapacitor, the geolocation of the board and its relative position to the light sources, the timing of the sunrise/sunset, and the weather condition. Therefore, rather than reporting our ad-hoc results, we refer to the typical energy levels detailed in [8]. Figure 10 shows these results extrapolated onto our setup. The supercapacitor reaches its maximum energy level at 12PM on the first day until the light intensity diminishes at 07.30PM. Since the system can not store more energy during this interval, the *excess energy* is wasted unless we spend it. The precomputation operations of the offline phase is an ideal candidate to consume energy during this period.

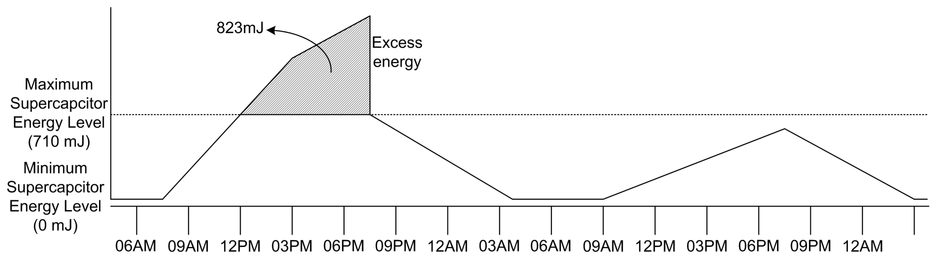

Fig. 10. The typical energy levels on the supercapacitor for two consecutive days (extrapolated from [8])

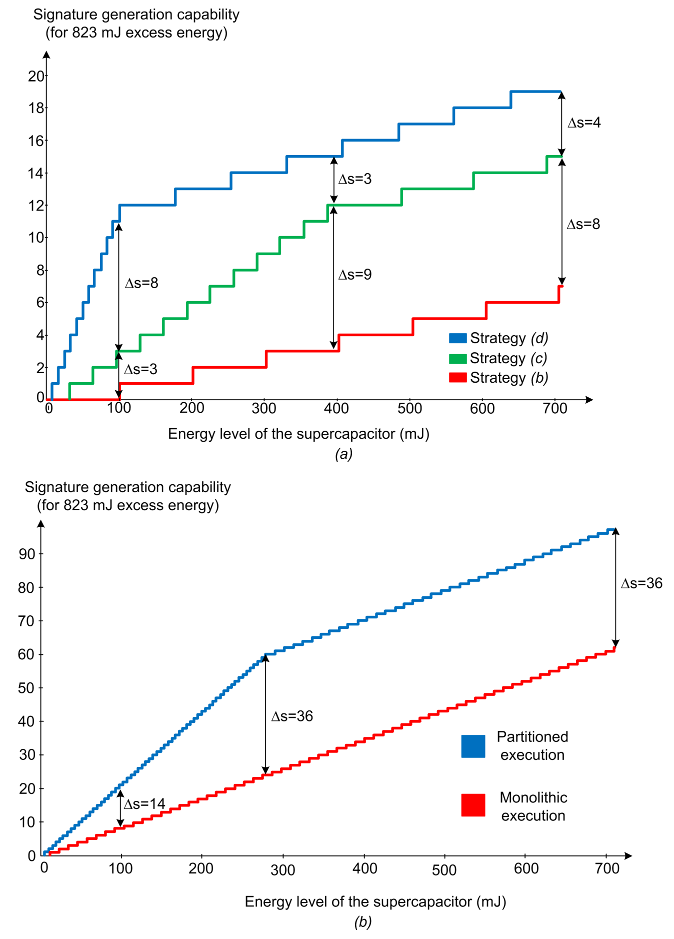

Fig. 11. Energy of the system vs. the total number of computable signatures for (a) hash-based and (b) lattice-based signatures

Next, we investigate how different approaches utilize the excess energy and the impact of offline/online partitioning on the system output. Figure 11 quantifies the energy-efficiency of the precomputation for hash-based (with $\omega = 8$) and lattice-based signatures respectively. These plots reflect the potentials of the system right after 7.30PM on the first day until the next excess energy interval occurs. The x-axis represents the energy level of the supercapacitor from 0 to the maximum energy level of 710 mJ and the y-axis shows the total number of signatures that system can generate for the corresponding energy level. For an excess energy of 823 mJ, the processor can perform the precomputation operations of 12 hash-based signatures. Hence, the strategy (d) and (c) (see Section 3.2 for an overview of these strategies) require significantly less energy to compute the first 12 signatures. Then, strategy (b) and (c) become equivalent but (d) still requires less energy to perform a signature due to its precomputation of the intermediate nodes. Δs corresponds to the difference of signature generation capability between different strategies. From these results we can clearly observe the effect of our optimization methods and deduce that its impact is much more important for critical energy levels. For example, if a burst signing request comes when the energy level is 100 mJ, the strategy (d), (c), and (b) can generate 11, 3, and 0 signatures respectively. This also shows that there are certain times (energy intervals) where the system can only generate a signature using an optimized strategy.

The energy required for lattice-based signatures (Figure 11 (b)) is much lower than its hash-based counterpart. The excess energy enables to precompute necessary values for the next 71 signatures, but the available memory space can only store 60 of them. Therefore, after the 60^{th} generation, it takes more energy to compute next signatures. For critical energy levels (eg. 100 mJ) the partitioned execution makes a significant improvement on the signature generation capability. However, maximum difference of 36 occurs around moderate energy levels following the 60^{th} generation and it remains fixed for the remainder of the energy spectrum. Note that, this implementation uses an $Offline\ table$ precomputation method, and it shows the signature candidate generation.

6 Related Work

There have been several implementations of post-quantum digital signatures. However, most of them are optimized for execution time, do not quantify the impact on energy, and none have the notion of precomputation with the offline/online phases. Güneysu *et al.* introduced the GLP signatures in [24] and report a baseline implementation on FPGAs. Then, it was optimized towards execution time with assembly level coding for high-end multiprocessors [25] and to lightweight microcontrollers [34], [9]. Recently, Pöppellmann proposed an FPGA implementation of BLISS which is an improved version of the GLP signature scheme [39]. On the other hand, hash-based signatures first implemented by Rohde et al. for constrained devices [43]. Then, several work proposed improved Winternitz and Merkle-tree constructions to reduce the execution time [19], [28].

Yet, the signing process (including the key generation) still is at the order of seconds.

7 Conclusions

Together with security assumptions, the principles of embedded systems may also change in time. This calls a rethinking of the design and optimization methodologies. In this paper, we investigated precomputation with partitioned execution modes as a potential optimization technique for post-quantum digital signatures. We later provided two proof-of-concept digital signature implementations utilizing these optimizations. We proved that today's embedded research platforms can effectively employ precomputation methodologies. We first showed that precomputation can save significant run-time energy and bring the execution time down to milliseconds range. Then, we highlight its impact on the overall system and demonstrate that it can increase the total number of signature computations for a typical use-case. Hence, we argue that in the emerging era of post-quantum real-time embedded systems, such optimizations will play a significant role. Precomputation methods will enable an improved real-time security with faster signatures and will provide energy-efficient solutions towards a greener future.

8 Acknowledgements

This work is supported in part by National Scientific Foundation (NSF) award no 1314598. The authors would like to thank Bilgiday Yuce for his extensive support on the energy-harvesting platform.

References

1. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an efficient library for cryptography. Homepage at http://code. google. com/p/relic-toolkit 163, 182 (2010)
2. Arndt, J.: Algorithms for programmers-ideas and source code (2004)
3. Ateniese, G., Bianchi, G., Capossele, A., Petrioli, C.: Low-cost standard signatures in wireless sensor networks: a case for reviving pre-computation techniques? In: Proceedings of the 20th Annual Network & Distributed System Security Symposium, NDSS 2013. NDSS2013, San Diego, CA (February 24-27 2013)
4. Aysu, A., Patterson, C., Schaumont, P.: Low-cost and area-efficient FPGA implementations of lattice-based cryptography. In: Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on. pp. 81–86 (June 2013)
5. Bai, S., Galbraith, S.: An improved compression technique for signatures based on learning with errors. In: Benaloh, J. (ed.) Topics in Cryptology CT-RSA 2014, Lecture Notes in Computer Science, vol. 8366, pp. 28–47. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-04852-9_2
6. Bansarkhani, R.E., Buchmann, J.: LCPR: High performance compression algorithm for lattice-based signatures and schnorr-like constructions. Cryptology ePrint Archive, Report 2014/334 (2014), http://eprint.iacr.org/

7. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). Information Theory, IEEE Transactions on 24(3), 384–386 (May 1978)
8. Bianchi, G., Capossele, A.T., Petrioli, C., Spenza, D.: AGREE: exploiting energy harvesting to support data-centric access control in WSNs. Ad Hoc Networks 11(8), 2625–2636 (2013)
9. Boorghany, A., Jalili, R.: Implementation and comparison of lattice-based identification protocols on smart cards and microcontrollers. Cryptology ePrint Archive, Report 2014/078 (2014), `http://eprint.iacr.org/`
10. Boyko, V., Peinado, M., Venkatesan, R.: Speeding up discrete log and factoring based schemes via precomputations. In: Nyberg, K. (ed.) Advances in Cryptology EUROCRYPT'98, Lecture Notes in Computer Science, vol. 1403, pp. 221–235. Springer Berlin Heidelberg (1998), `http://dx.doi.org/10.1007/BFb0054129`
11. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C., Moura, A. (eds.) LATIN'98: Theoretical Informatics, Lecture Notes in Computer Science, vol. 1380, pp. 163–169. Springer Berlin Heidelberg (1998), `http://dx.doi.org/10.1007/BFb0054319`
12. Brickell, E., Gordon, D., McCurley, K., Wilson, D.: Fast exponentiation with precomputation. In: Rueppel, R. (ed.) Advances in Cryptology EUROCRYPT 92, Lecture Notes in Computer Science, vol. 658, pp. 200–207. Springer Berlin Heidelberg (1993), `http://dx.doi.org/10.1007/3-540-47555-9_18`
13. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.Y. (ed.) Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 7071, pp. 117–129. Springer Berlin Heidelberg (2011), `http://dx.doi.org/10.1007/978-3-642-25405-5_8`
14. Buettner, M., Greenstein, B., Wetherall, D.: Dewdrop: An energy-aware runtime for computational RFID. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. pp. 197–210. NSDI'11, USENIX Association, Berkeley, CA, USA (2011), `http://dl.acm.org/citation.cfm?id=1972457.1972478`
15. Courtois, N., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) Advances in Cryptology ASIACRYPT 2001, Lecture Notes in Computer Science, vol. 2248, pp. 157–174. Springer Berlin Heidelberg (2001), `http://dx.doi.org/10.1007/3-540-45682-1_10`
16. Czypek, P.: Implementing Multivariate Quadratic Public Key Signature Schemes on Embedded Devices. Ph.D. thesis, Diploma Thesis, Chair for Embedded Security, Ruhr-Universität Bochum (2012)
17. Dods, C., Smart, N., Stam, M.: Hash based digital signature schemes. In: Smart, N. (ed.) Cryptography and Coding, Lecture Notes in Computer Science, vol. 3796, pp. 96–115. Springer Berlin Heidelberg (2005), `http://dx.doi.org/10.1007/11586821_8`
18. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J. (eds.) Advances in Cryptology CRYPTO 2013, Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer Berlin Heidelberg (2013), `http://dx.doi.org/10.1007/978-3-642-40041-4_3`
19. Eisenbarth, T., von Maurich, I., Ye, X.: Faster hash-based signatures with bounded leakage. In: Lange, T., Lauter, K., Lisonk, P. (eds.) Selected Areas in Cryptography

– SAC 2013, pp. 223–243. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2014), http://dx.doi.org/10.1007/978-3-662-43414-7_12

20. Faugere, J.C., Gligoroski, D., Perret, L., Samardjiska, S., Thomae, E.: A polynomial-time key-recovery attack on MQQ cryptosystems. Cryptology ePrint Archive, Report 2014/811 (2014), http://eprint.iacr.org/
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A. (ed.) Advances in Cryptology CRYPTO 86, Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer Berlin Heidelberg (1987), http://dx.doi.org/10.1007/3-540-47721-7_12
22. Gallager, R.: Low-density parity-check codes. Information Theory, IRE Transactions on 8(1), 21–28 (January 1962)
23. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 212–219. STOC '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/237814.237866
24. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: A signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems CHES 2012, Lecture Notes in Computer Science, vol. 7428, pp. 530–547. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-33027-8_31
25. Güneysu, T., Oder, T., Pöppelmann, T., Schwabe, P.: Software speed records for lattice-based signatures. In: Gaborit, P. (ed.) Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 7932, pp. 67–82. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-38616-9_5
26. Harari, E.: The non-volatile memory industry - a personal journey. In: Memory Workshop (IMW), 2011 3rd IEEE International. pp. 1–4 (May 2011)
27. Hülsing, A.: W-OTS+ shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A. (eds.) Progress in Cryptology AFRICACRYPT 2013, Lecture Notes in Computer Science, vol. 7918, pp. 173–188. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-38553-7_10
28. Hülsing, A., Busold, C., Buchmann, J.: Forward secure signatures on smart cards. In: Knudsen, L., Wu, H. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 7707, pp. 66–80. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-35999-6_5
29. Koyama, K., Tsuruoka, Y.: Speeding up elliptic cryptosystems by using a signed binary window method. In: Brickell, E. (ed.) Advances in Cryptology CRYPTO 92, Lecture Notes in Computer Science, vol. 740, pp. 345–357. Springer Berlin Heidelberg (1993), http://dx.doi.org/10.1007/3-540-48071-4_25
30. Lamport, L.: Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International Palo Alto (1979)
31. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) Advances in Cryptology ASIACRYPT 2009, Lecture Notes in Computer Science, vol. 5912, pp. 598–616. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10366-7_35
32. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 5086, pp. 54–72. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-71039-4_4

33. Mane, D., Schaumont, P.: Energy-architecture tuning for ECC-based RFID tags. In: Hutter, M., Schmidt, J.M. (eds.) Radio Frequency Identification, pp. 147–160. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-41332-2_10
34. von Maurich, I., Güneysu, T.: Towards side-channel resistant implementations of QC-MDPC McEliece encryption on constrained devices. In: Mosca, M. (ed.) Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 8772, pp. 266–282. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-11659-4_16
35. Merkle, R.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO 89 Proceedings, Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer New York (1990), http://dx.doi.org/10.1007/0-387-34805-0_21
36. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. pp. 2069–2073 (July 2013)
37. Pabbuleti, K., Mane, D., Schaumont, P.: Energy budget analysis for signature protocols on a self-powered wireless sensor node. In: Saxena, N., Sadeghi, A.R. (eds.) Radio Frequency Identification: Security and Privacy Issues, pp. 123–136. Lecture Notes in Computer Science, Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-13066-8_8
38. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small public keys and fast verification for Multivariate Quadratic public key systems. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems CHES 2011, Lecture Notes in Computer Science, vol. 6917, pp. 475–490. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23951-9_31
39. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems CHES 2014, Lecture Notes in Computer Science, vol. 8731, pp. 353–370. Springer Berlin Heidelberg (2014), http://dx.doi.org/10.1007/978-3-662-44709-3_20
40. Pöppelmann, T., Güneysu, T.: Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In: Hevia, A., Neven, G. (eds.) Progress in Cryptology LATINCRYPT 2012, Lecture Notes in Computer Science, vol. 7533, pp. 139–158. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-33481-8_8
41. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. Quantum Info. Comput. 3(4), 317–344 (jul 2003), http://dl.acm.org/citation.cfm?id=2011528.2011531
42. Ransford, B., Sorber, J., Fu, K.: Mementos: System support for long-running computation on RFID-scale devices. SIGARCH Comput. Archit. News 39(1), 159–170 (Mar 2011), http://doi.acm.org/10.1145/1961295.1950386
43. Rohde, S., Eisenbarth, T., Dahmen, E., Buchmann, J., Paar, C.: Fast hash-based signatures on constrained devices. In: Grimaud, G., Standaert, F.X. (eds.) Smart Card Research and Advanced Applications, Lecture Notes in Computer Science, vol. 5189, pp. 104–117. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85893-5_8
44. de Rooij, P.: Efficient exponentiation using precomputation and vector addition chains. In: De Santis, A. (ed.) Advances in Cryptology EUROCRYPT'94, Lec-

ture Notes in Computer Science, vol. 950, pp. 389–399. Springer Berlin Heidelberg (1995), http://dx.doi.org/10.1007/BFb0053453
45. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on. pp. 124–134 (November 1994)
46. Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography PKC 2012, Lecture Notes in Computer Science, vol. 7293, pp. 156–171. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-30057-8_10
47. V. Lyubashevksy: Lattice signatures without trapdoors. In: Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 738–755. EUROCRYPT'12, Springer-Verlag, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-29011-4_43
48. Whyte, W., Weimerskirch, A., Kumar, V., Hehn, T.: A security credential management system for V2V communications. In: Vehicular Networking Conference (VNC), 2013 IEEE. pp. 1–8 (Dec 2013)
49. Zhu, T., Mohaisen, A., Ping, Y., Towsley, D.: DEOS: Dynamic energy-oriented scheduling for sustainable wireless sensor networks. In: INFOCOM, 2012 Proceedings IEEE. pp. 2363–2371 (March 2012)