GEOVANDRO CARLOS CREPALDI FIRMINO PEREIRA

MULTIVARIATE AND HASH-BASED POST-QUANTUM DIGITAL SIGNATURES

ASSINATURAS DIGITAIS PÓS-QUÂNTICAS MULTIVARIADAS E BASEADAS EM HASH

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

São Paulo
2015

GEOVANDRO CARLOS CREPALDI FIRMINO PEREIRA

MULTIVARIATE AND HASH-BASED POST-QUANTUM DIGITAL SIGNATURES

ASSINATURAS DIGITAIS PÓS-QUÂNTICAS MULTIVARIADAS E BASEADAS EM HASH

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Paulo Sérgio Licciardi Messeder Barreto

São Paulo
2015

Catalogação-na-publicação

Pereira, Geovandro Carlos Crepaldi Firmino
Multivariate and Hash-based Post-quantum Digital Signatures (Assinaturas Digitais Pós-Quânticas Multivariadas e Baseadas em Hash) / G. C. C. F. Pereira -- versão corr. -- São Paulo, 2015.
122 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Criptologia 2.Algoritmos 3.Segurança de redes I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Inúmeras foram as pessoas que cruzaram minha trajetória e que contribuíram de uma forma ou outra para que este trabalho de doutorado fosse concluído com êxito.

Lembrando que ”sem a música, a vida seria um erro”, agradeço à dupla Daft Punk por ter lançado seu fantástico álbum ”Random Access Memories” justamente em um momento que eu tanto precisava de novas vibrações para continuar. Guy-Manuel de Homem-Christo e Thomas Bangalter foram capazes de transcender a esse objetivo com maestria e perfeição.

Não pretendo cometer injustiças, então agradeço de forma geral às pessoas com quem me envolvi no SEMBEI, onde partilhei de discussões acaloradas sobre assuntos inimagináveis e que foram valiosos para me tornar uma pessoa mais completa. Dali também surgiram boas amizades.

Não posso deixar de agradecer meu amigo Paulo Barreto que me proveu as condições necessárias para que pudesse desenvolver o gosto e as habilidades do pensar científico, sempre de forma séria e voltada a resolver problemas do mundo real. Creio que isso constitua o bem mais valioso que levo dessa etapa. Ademais, sou grato por contar com sua amizade que inclui desde um convite sem motivos para comer um temaki, até um ombro amigo quando passei por situações, digamos, complicadas.

Vale também aqui mencionar uma pessoa que plantou em mim uma ”semente da vitória”, em alusão ao livro do Nuno Cobra. Foi à época do meu ensino médio que Adevaldo, um grande professor apaixonado pela matemática, foi capaz de me contaminar com essa paixão, compartilhando da sua energia e dedicação ao ensino da mesma. Sua participação foi determinante e me abriu uma nova perspectiva de vida.

Tive sorte de ter mais semeadores em minha trajetória. Menciono aqui a Lorena, uma grande amiga de graduação (e ainda dizem que na poli não tem mulher!) que me presenteou em algum dos meus 29 aniversários com o livro ”Fortaleza Digital”. Semente essa que me levou a gostar de criptografia e, de alguma forma iluminou meu caminho e permitiu que me adentrasse nesse assunto inicialmente obscuro.

Tenho imensa gratidão ao meu irmão Adriano, que está sempre compartilhando comigo os seus desafios vencidos, seja na culinária, seja na faculdade, ou seja com os 60 *km* percorridos de bike. Obrigado por sempre estar ao meu lado repassando essa energia de vida positiva, Nano!

Incondicionalmente, tenho de deixar documentado o amor e orgulho que tenho por meus pais. Olhando para trás, reconheço claramente que deram tudo o que podiam para crescer seus dois preciosos filhos. Guardo comigo uma bela imagem de seu Antônio, sempre dedicado ao trabalho, à honestidade, ao bom humor e a compartilhar seus conhecimentos sobre a natureza. A dona Ivone é onipresente em mim. É uma estrela que me segue, e está sempre lá para esquentar-me com sua luz própria quando estou com frio e nunca me deixa desanimar.

RESUMO

Os esquemas convencionais de assinatura digital mais usados na atualidade têm sua segurança ameaçada com a possibilidade da construção de um computador quântico de grande porte. Ademais, tais esquemas não têm se mostrado completamente adequados para uso em plataformas com recursos computacionais extremamente escassos. Surge então a necessidade da busca por alternativas que satisfaçam as condições de segurança a médio e longo prazo, além de apresentarem desempenho razoável quando poucos recursos computacionais estão disponíveis.

Este trabalho obtém assinaturas digitais pós-quânticas multivariadas quadráticas e baseadas em hash mais eficientes e tem o intuito de torná-las práticas em cenários como Internet das Coisas e Redes de Sensores Sem Fio (RSSF), caracterizados por apresentarem dispositivos com recursos computacionais limitados. No contexto de assinaturas multivariadas quadráticas, descreve-se uma nova técnica que tenta minimizar o principal gargalo desses esquemas, o grande tamanho de chaves. A nova técnica explora certos anéis matriciais com estrutura compacta. Mostra-se que alguns dos anéis analisados não são seguros (um dos ataques apresenta tempo polinomial), enquanto outros anéis menos compactos aparentam não sofrer ataque polinomial, mas infelizmente ainda não são adequados para uso em dispositivos muito restritos.

Por outro lado, descreve-se um método para obter assinaturas digitais baseadas em hash que fornece redução das assinaturas para $\approx 2/3$ do tamanho original do esquema *multi-time* Merkle-Winternitz. De fato, o tamanho das assinaturas constitui o principal gargalo desses esquemas. A melhoria também acarreta uma redução em $\approx 2/3$ nos tempos de execução (geração de chave, geração de assinatura e verificação de assinatura) e no consumo de energia para essas operações quando executadas em um microcontrolador AVR tipicamente usado em Redes de Sensores Sem Fio, o ATmega128L. Este resultado torna-se promissor para implantação de assinaturas baseadas em hash no cenário de Internet das Coisas.

Palavras-chave: Criptologia. Assinaturas Digitais. Plataformas Restritas. Redes de Sensores Sem Fio. Criptografia Pós-Quântica.

ABSTRACT

The conventional digital signature schemes widely used today may have their security threatened with the possibility of the rising of a large quantum computer. Moreover, such schemes are not entirely suitable for utilization on very constrained-resource platforms. Therefore, there is a need to look at alternatives that present reasonable security in the medium and long term, in addition to attaining acceptable performance when few resources are available.

This work provides more efficient multivariate and hash-based post-quantum digital signatures and targets the deployment in scenarios like Internet of Things and Wireless Sensor Networks where the typical devices are very resource-constrained. In the context of multivariate quadratic digital signatures we describe a new technique that attempts to minimize the main drawbacks of these schemes, the large key sizes. The new technique explores certain structured compact matrix rings. Some of the analyzed matrix rings are not secure (one of the attacks runs in polynomial time). Other less compact matrix rings are investigated and they apparently do not suffer a polynomial time attack, but unfortunately are still far from deployment on very constrained platforms.

On the other hand, this work describes a method for hash-based signatures providing a $\approx 2/3$ reduction of the signature sizes in the Merkle-Winternitz multi-time signature scheme. In fact, the signature sizes constitute the main bottleneck of these schemes. The improvement also leads to a $\approx 2/3$ reduction in the run times (key generation, signing and verifying) and in energy consumption for all these operations on an AVR ATmega128L microcontroller, typically found in Wireless Sensor Networks. This result is much more promising for the deployment in an IoT scenario.

Keywords: Cryptology. Digital Signatures. Constrained Platforms. Wireless Sensor Networks. Post-Quantum Cryptography.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

USP	University of São Paulo
$\mathbb{F}_q$	Finite field with q elements.
$\mathbb{F}_q^{p\times p}$	Ring of $p \times p$ matrices whose entries are taken from $\mathbb{F}_q$.
$\mathcal{S}$	Private Map or Linear Transform of variables.
$\mathcal{T}$	Private Linear Transformation of polynomials in $\mathbb{F}_q^{m\times m}$.
t_{ij}	Entry of row i and column j in matrix T.
$\mathcal{Q}$	Private Quadratic transformation in $\mathbb{F}_q^{n\times n\times m}$.
$\mathcal{T} \circ \mathcal{S}$	Composition of two maps T and S, written from right to left, i.e. $T(S(x))$.
$\mathcal{Q} \circ \mathcal{S}$	Composition of a linear transform S and a quadratic one Q, written from right to left, i.e. $Q(S(x)) = SQ^{(k)}S^T, 1 \leq k \leq m$.
$Q^{(k)}$	The superscript between parenthesis (k) is used to enumerate matrices or polynomials.
$\boldsymbol{x}$	A bold character denotes a row vector and $\boldsymbol{x}^T$ the respective column vector.
x_i	The i-th entry of a vector $\boldsymbol{x}$.
ASM	Assembly language.
DM	Davies-Meyer compression function.
EIP	Extended Isomorphism of Polynomials.
sk	Secret Key.
pk	Public Key.

ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman key exchange protocol
GNFS	Generalized Number Field Sieve
GPU	Graphics Processing Unit
IoT	Internet of Things.
IP	Isomorphism of Polynomials.
MMO	Matyas-Meyer-Oseas compression function.
MQ	Multivariate Quadratic.
PKC	Public Key Cryptography.
WSN	Wireless Sensor Networks.

1 INTRODUCTION

The requirements and restrictions imposed to *software* libraries are becoming increasingly drastic due to the continuous reduction of devices' available resources. This, indeed, is a trend with the emergence of the *Internet of Things* (IoT), where each object in the real world is endowed with processing and communication capabilities. The underlying resources may offer little processing power and storage in order to enable its large-scale deployment at a low cost. A great number of applications require authentication services including health monitoring, software updates, M-factor authentication in financial online transactions, and many others. A security framework for such an environment should be lightweight enough to not compromise the main applications, and consume only a small relative amount of energy given that many devices are battery powered.

Indeed, among the desired cryptographic primitives for the typical security services, the bottleneck is certainly constituted by the asymmetric cryptographic algorithms. There are few asymmetric alternatives in terms of the computational underlying problem that security relies on. RSA, DSA, and ECDSA standards (GALLAGHER; KERRY, 2013; UNE; KANDA, 2007), for example, are mainly based on the problems of the *integer factorization*, *discrete logarithm* and *elliptic curve discrete logarithm*. In 1997, however, Peter Shor showed that a sufficiently large quantum computer solves these problems in polynomial time, which would make the current standards obsolete (SHOR, 1997). After Shor's discovery, the cryptosystems that rely on problems conjectured yet intractable under quantum attacks were promoted to the post-quantum realm.

Thus, unfortunately, current standards do not belong to this realm, and seeking for post-quantum alternatives is currently a quite intensive area of research (BERNSTEIN; BUCHMANN; DAHMEN, 2009; BARRETO et al., 2014).

Amid the most promising alternatives for digital signatures, Multivariate Quadratic (MQ) public key schemes have attracted researchers' attention in the last two decades for two reasons. First, they seem to resist to attacks by large quantum computers and second, they have reasonable efficiency and simpler underlying operations, i.e. operations based on small finite fields, thus avoiding multiple precision arithmetic. Their main drawback is the large key sizes, having about 10 KiB for the most recent works (PETZOLDT; BUCHMANN, 2009). Thus, a particular attention on reducing these sizes is highly relevant.

Also, post-quantum hash-based signatures is another topic with many advantages. A very special property is that their security is only based on a secure hash function which is also necessary for the other digital signature schemes. And all the other signatures rely on additional assumptions such as *one-way trapdoor functions* (NAOR; YUNG, 1989). Another advantage is that there are a plenty of secure hash functions in the literature, and in the case in which some adopted underlying hash function is considered insecure, it could be easily replaced by another one. Thus, relying on a very small assumption makes hash-based signatures very special. Furthermore, hash functions are usually fast and demand small amount of memory when compared to asymmetric primitives, being suitable for constrained devices. The drawback of hash-based signatures is the signature sizes, and a particular focus on this issue is deserved.

Post-quantum digital signatures based on lattices also present some advantages. They involve fast algorithms for generating and verifying a signature, based on less complex operations including vector and matrix arithmetic. On the other hand, the main challenge is the associated key sizes and signatures, for example 128 KiB in (LYUBASHEVSKY, 2012). In this sense, there is an effort of improving these sizes to an acceptable level in order to make these schemes ready to use (BERNSTEIN; BUCHMANN; DAHMEN, 2009; BARRETO et al., 2014).

It is worth to mention that there is a recent new post-quantum research topic based on elliptic curve isogenies (COUVEIGNES, 2006; STOLBUNOV, 2012). In 2014, De Feo, Jao, and Plut showed how to construct efficient and provable secure 1) key-exchange with forward secrecy, 2) encryption and 3) zero-knowledge identification primitives from the difficult of computing supersingular elliptic curve isogenies (DE FEO; JAO; PLÛT, 2014). Also in 2014, researchers extended the security of the new key-exchange protocol to a form of digital signature with strong designated verifier (SUN; TIAN; WANG, 2014). Subsequently, Jao and Soukharev came up with an alternative method of creating undeniable signatures with designated verifier (JAO; SOUKHAREV, 2014). Besides a full-fledged digital signature scheme has not been constructed yet based on this new primitives, there is much room for upcoming breakthroughs in this area.

1.1 Scenario

Now we describe two scenarios which will serve as a target substratum for this work, namely the Internet of Things and the Wireless Sensor Networks. These two scenarios have in common the presence of small limited-resource devices able to communicate themselves and establish truly interconnected networks.

Since the above mentioned devices may be accessible to the external world, users' data or online services may be exposed to malicious attacks including tampering, eavesdropping and attempts to modify the transmitted data and insert unauthorized messages into the network. Thus, there is a necessity of improving or building new security mechanisms for this environment since the current ones – mainly Public-Key Cryptography (PKC) algorithms may impose prohibitive use in this scenario (see Section 1.2). It is essential that they be efficient enough to not seriously compromise the performance of the main applications running on the devices.

This is not an easy task, mainly when trying to construct public-key cryptography, in particular digital signatures, which is much more expensive compared to symmetric cryptography.

1.1.1 Internet of Things

A recent paradigm under intensive development is the **Internet of Things**. In IoT, any real-world object may be equipped with computational capabilities provided by embedded systems. Depending on the application, these devices may not only exchange information locally with peers, but also globally with entities on the Internet. Most of them are also able to connect to the Internet, obtain new information, and update themselves autonomously. To mention a few practical examples, we have Internet-connected cameras that allow people to post pictures online with a single click; home automation systems that turn on the front porch light when someone leave work; and bracelets that share with friends how far someone have biked or run during the day.

A side-effect of the IoT interconectivity is the susceptibility to many network security vulnerabilities. Thus, attacks that have been primarily targeted against PCs could, suddenly, be launched against cars, mobile phones, *e-tickets*, RFIDs or even pacemakers.

The IoT scenario is typically characterized by having devices provided with very limited processing power – sometimes less than a 10 MHz of clock – and small storage resources – like 4 KiB of RAM. Further, this environment presents low bandwidth communication channels such as ZigBee in Personal Area Networks based on the IEEE 802.15.4 communication standard and also SMS which supports only a very limited message length. These channels present relatively higher transmission costs. For instance, in the case of SMS, the sender may be charged an amount of money to transmit each 140-byte message.

1.1.2 Wireless Sensor Networks

Wireless Sensor Networks are composed of small autonomous devices, also called sensor nodes, which are able to process and communicate data acquired from the environment in which they are deployed. Their low cost and rapidity of deployment make them particularly attractive for many applications requiring security services – climate and health monitoring, pollution detection, building automation, etc (ARAMPATZIS; LYGEROS; MANESIS, 2005; PUCCINELLI; HAENGGI, 2005). Since sensor nodes are usually deployed in large scale, manufacturing costs become one of the main concerns for the designers and the underlying resources tend to be scarce.

In WSNs the energy consumption is more expensive for transmitting than processing a bit of data (MARGI et al., 2010). The transmission of data in WSN devices is expensive because the radio has to be turned on, spending a relatively higher amount of energy to this task. This becomes more drastic given that data is transmitted through small packets by the typical operating systems, e.g. a 29-byte payload is the default for TinyOS operating system. The transmission of messages larger than the payload size would incur increased time of radio turned on, trying to send all the packets, increasing the battery drain which reduces sensor's lifetime. Therefore, cryptographic solutions must have low cost in terms of transmission overheads in addition to processing and storage.

In short, there are two main sources of power consumption for deploying security services in sensor nodes. The one dedicated to PKC *computations* itself and the one related to *communication* of the higher level protocols using a security mechanism, e.g the Elliptic Curve Diffie-Hellman key exchange with authentication provided by the Elliptic Curve Digital Signature Algorithm ECDH-ECDSA used in SSL/TLS (DIERKS; RESCORLA, 2008).

The candidate device selected for evaluating our solutions is the AVR low-power microcontroller ATmega128L from Atmel which is typically embedded in MICAz sensor nodes, widely employed in Wireless Sensor Networks. ATmega128L has an 8-bit CPU and operates at 7.37 MHz clock speed (MEMSIC, 2012). It provides 4 KiB of SRAM and 128 KiB ROM (flash) storage.

1.2 Motivation

An important concern in the context of digital signatures for IoT is the question of what digital signature schemes will remain secure and efficient in medium to long terms.

We begin the motivation by first describing some application contexts where digital signatures are important. One scenario to be considered consists of *software updates*. Today's applications running on smartphones and PCs continuously receive many software updates over time – many bugs are found and then fixed, security flaws are frequently discovered and new features are often introduced. According to Yahoo Aviate, Android users have an average of 95 applications installed per phone (SAWERS,). If we assume that each application is monthly updated, an average user device would receive $1,140$ software updates per year, which is a relevant amount of traffic and may be easily susceptible to attacks by malicious entities trying to impersonate the software providers and execute malicious code on users' devices.

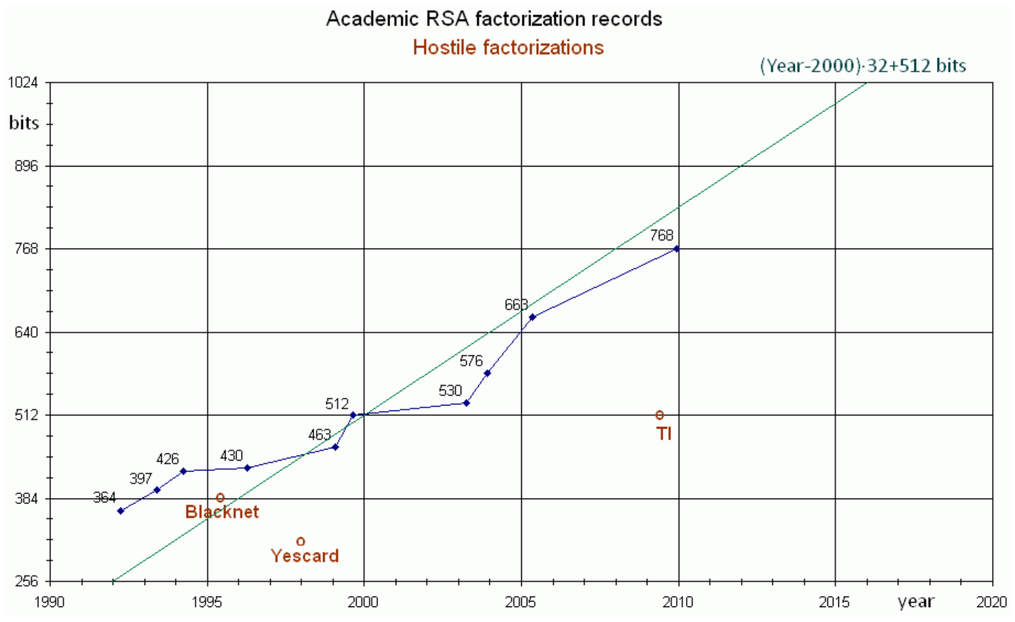
One sensitive application that receives software updates is described in the following example. The recent Tesla Model S car is enabled for receiving over-the-air software updates using a WiFi connection. So after receiving a new update the car's owner just have a brand new car with additional features. Tesla declares that the Model S is not a car but a sophisticated computer on wheels (HIRSCH, 2015). One important question that arises in the above scenario is: how a user's device knows that the software updates are authentic so that it can trust them? If there was no origin authentication a terrorist could pretend to be Tesla and remotely send a malicious software update targeting a car accident.

Another example where authenticity becomes important is in the context of Software Defined Networks (SDN). In SDN, network devices like routers and switches do not need to execute routing algorithms anymore, they typically receive an already optimized routing table for its network computed by a trusted controller. This kind of technology can be easily deployed in conventional wired network devices given the computational resources available. But trying to bring SDN to Wireless Sensor Networks for example may constitute an issue given the scarcity of computational resources available like 4KiB of RAM and 7.37 MHz clock in a typical MICAz sensor node (MEMSIC, 2012). The recent work of Oliveira *et. al.* provides an implementation of SDN over WSNs consuming 3.2 KiB of RAM and does not include cryptographic primitives to achieve end-to-end authenticity (OLIVEIRA; MARGI; GABRIEL, 2014). If no authentic control messages are used in the name of the trusted controller, new attacks could be launched. See for example the following survey describing WSN-related attacks (MARGI et al., 2009)).

Fortunately, there are cryptographic services that provide origin authentication, e.g. *digital signatures*. As described in Section 1, the most widely employed algorithms for digital signatures are the RSA, DSA and ECDSA (GALLAGHER; KERRY, 2013; UNE; KANDA, 2007) relying their security on the computational problems of, respectively, integer factorization, discrete logarithm and elliptic curve discrete logarithm. We now argue that these standards are not fully suitable for use on constrained devices in a medium to long terms.

Although integer factorization is believed to be hard, many cryptanalytic efforts including distributed computation and use of dedicated GPUs have had some progress over the years. Surprisingly, the advances are faster than the usual assumption that the cryptanalytic progress halves the computational complexity every 18 months by the action of Moore's law. Figure 1 (GRIEU, 2015) illustrates some milestones on factoring integers (in particular, RSA numbers) using mainly the *General Number Field Sieve* (GNFS) technique (LENSTRA et al., 1993).

Figure 1: Progress on factoring RSA numbers



Source: (GRIEU, 2015)

According to Figure 1, the blue points represent the academic progress which is the most effective effort (see for example (BERNSTEIN; LANGE, 2014) which employs dedicated hardware methodologies for this task). The red points illustrate the non-academic groups progress including hackers and intelligence agencies. So far, non-academic groups have always been some years behind academic efforts. The green line gives a linear estimate of the academic factoring progress. Notice that in the 4-year period of $\approx$ 2005 – 2009 the moduli factorized jumped from 663 to 768 bits (in blue), a 105-bit displacement, which is far ahead the expected effect of Moore's law for a 4-year period. Besides the current factoring record is a 768 bits modulus, by the end of 2009, Kleinjung *et. al.* (KLEINJUNG et al., 2010) assume that it is not unreasonable to expect that 1024-bit RSA moduli, still used in many applications, can be factored well within this decade by an academic effort. According to the green line estimate in Figure 1 this factorization would happen by 2016. Other authors also share similar opinion and consider that using an approach of combined GNFS with the Moore's law action this factorization is expected to occur somewhere around 2018 (BRENT, 2000).

In order to prevent against these cryptanalytic advances, RSA key sizes have to be increased. The current recommendation by NIST is to use at least 2048-bit keys until 2030 offering 112 bits of symmetric security (GALLAGHER; KERRY, 2013). The drawback of RSA is that there is a known effect where each time the key size is doubled the security level is increased by a factor smaller than two and the decryption becomes 6–7 times slower (COFFEY, 2012). Thus, sticking on RSA even considering a classical scenario (no quantum computers involved) is not a great deal in any context. In addition in the case of a quantum scenario it turns out to be even worse for RSA when large quantum computers are built becoming immediately insecure (SHOR, 1997) .

Even the alternative standards to RSA such that DSA and ECDSA are also not very promising considering a classical scenario. Antoine Joux initiated a mini-revolution in 2012, by successive record-breaking in solving the discrete logarithm problem defined over fields of small characteristic (JOUX, 2012; JOUX, 2013a; JOUX, 2013c). One main resulting effect is that supersingular binary elliptic curves are not anymore recommended for cryptographic applications (GALBRAITH, 2013; JOUX, 2013b). This makes the search for alternative digital signatures a relevant effort.

It is also important to mention that on constrained platforms, these standards (RSA, DSA and ECDSA) do not profit a natural instantiation completely avoiding a cryptographic co-processor, given their complex underlying operations, e.g. multiple precision arithmetic for medium and big finite fields.

Among the existing post-quantum digital signatures, many advances have been observed in three lines: schemes based on hash functions (MERKLE, 1979), on multivariate quadratic equations ($\mathcal{MQ}$) over finite fields (WOLF; PRENEEL, 2005) and on lattices (GÜNEYSU; LYUBASHEVSKY; PÖPPELMANN, 2012). The searching for post-quantum alternatives a quite intensive area of research (BERNSTEIN; BUCHMANN; DAHMEN, 2009; BARRETO et al., 2014).

From the above analysis it becomes clear that alternatives to today's digital signature standards are required in two aspects: underlying computational problem and efficiency in order to deliver satisfactory authenticity services on resource constrained devices in medium and long term.

1.3 Goals

This work targets to improve the efficiency drawbacks of post-quantum digital signatures making them more adequate for constrained embedded devices typical of IoT and WSNs providing practical alternatives to today's standards. In order to tackle this goal this work will develop new optimization techniques for two promising post-quantum digital signature schemes, namely multivariate quadratic ($\mathcal{MQ}$) signatures and hash-based signatures.

Shortly, this thesis intends to give a step forward to practicality of the two mentioned digital signatures and, to achieve that, the work will involve alleviating their main drawbacks: multivariate signature key sizes and hash-based signature sizes, and validating the practicality of the constructions through implementation for a very constrained device.

1.4 Methodology

In order to achieve the above described goals, it is necessary to conduct a literature review concerning the construction of non-conventional digital signatures, in order to better understand the state-of-the-art of these cryptosystems, what are the relevant attacks, and how to select suitable parameters that resist them. Based on the review, new approaches will be proposed for parametrizing the cryptographic schemes by introducing a particular structure appropriate for compact representation.

The new approaches may introduce new shortcuts in turn for the attacks due to the additional structure inserted. This fact must and will be taken into account by the presented approaches and the possible attacks will be analyzed in order to obtain the proper parameters.

The proposed cryptosystems must improve (when applicable) some of the following relevant metrics for the target platforms:

- smaller key sizes,
- smaller signature sizes,
- smaller code size (ROM),
- smaller dynamic memory (RAM),
- acceptable run times for signature generation and verification procedures,
- reduced energy consumption.

It is worth mentioning that in the Internet of Things scenario, these metrics may not be necessarily compared with conventional cryptographic schemes for the case that any of them are not even deployable in the target platform. So the comparison would be in relation to the available resources in the scenario bottlenecks. Indeed, conventional schemes may not be even remotely deployable in some of the involved platforms, with which a comparison would not make sense, merely by impossibility. Instead, the concrete metrics for these platforms, that determine the semantic of the "reduced" memory consumption and of the "acceptable" times, refer to the goal of offering the desired cryptographic services in the available storage, and notoriously exiguous, of ROM and RAM, without compromising the main application with unacceptable perceptual run times due to the characteristic low speed of embedded devices, i.e. the end user benefited by the offered cryptographic services, should not feel a discomfort greater than he would feel with conventional schemes in typical platforms with more resources.

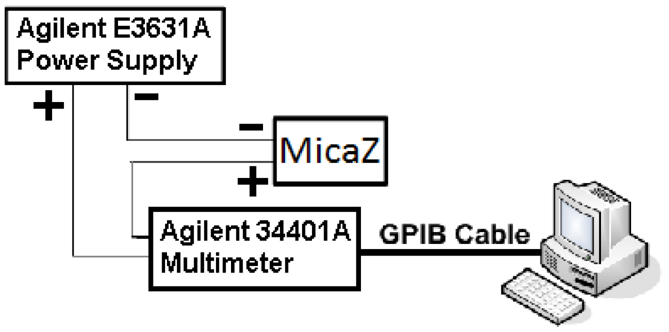
To evaluate the memory-related, run time, and energy consumption metrics we will provide a software implementation of the target algorithm candidate mainly in C language, and Assembly language for critical procedures, for the MICAz wireless sensor node whose characteristics were described in Section 1.1.2. For RAM and ROM memory evaluation it will be used the values estimated by the avr-gcc compiler and the program will be compiled along with the TinyOS operating system. The memory consumed by the TinyOS will be discounted from the overall measured.

The MICAz sensor node is adequate for this work since besides of being a very constrained device it can be also used to conduct energy measurements of the algorithms following the methodology described below.

Energy measurement. When IoT devices have limited lifetime, which is the case of wireless sensors powered by battery, energy consumption becomes an important metric to take into account by the algorithms. Therefore, we also investigate energy consumption of our implementation running on a MICAz sensor. In order to obtain an accurate measurement of the energy consumption, we perform direct measurements on the target platform.

A Agilent E3631A (AGILENT, 2010) power supply is configured to provide a constant voltage to power the target device. The Agilent 34401A (AGILENT, 2012) digital multimeter (DMM) is used to measure the current flow as the different hardware subsystems become active/inactive during the tasks' execution. Figure 2 shows a block diagram describing our measurement setup: a GPIB (General Purpose Interface Bus – IEEE 488 standard) cable is used to connect the Agilent 34401A DMM to a computer running the software LabView (NATIONAL INSTRUMENTS, 2012), which collects and records the measurement samples. The reading rate of DMM is configured to 60 Hz.

Figure 2: Measurements setup



Source: Adapted from (SANTOS et al., 2010)

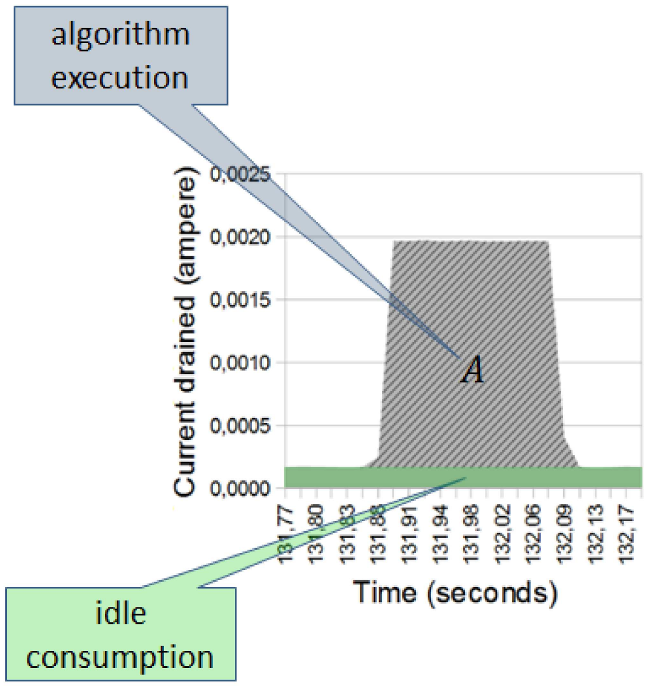
We run each algorithm and measure the current drained. We obtain the charge via time integration of the current and, since voltage is constant, we have the energy consumption. We also measure the system idle current drained (i.e., no particular tasks being executed), obtaining the threshold that is deduced from the measurements. Therefore, the system's overall energy consumption is given by the energy consumed by each task added to the energy consumed when the system is idle. This methodology is described in (SANTOS et al., 2010).

The typical fingerprint of the current sampling graphic is shown in Figure 3. The operating system idle consumption, i.e. no tasks running besides the OS, is shown in green and the current-related to the algorithm execution is identified in grey (region A). In order to separate the actual algorithm current drain consumption the overall area is subtracted from the green area for a given time interval.

For our experiments we run the target operation a sufficient number of times at once and divide the total energy amount by the number of iterations to get the consumption of a single execution of the operation. This calculation is conducted using a Python script that receives the LabView output file (current drained vs time) and then computes the energy consumption in mJ. Afterward, we divide this value by the number of iterations the target operation was executed.

In addition, the operation run time can also be obtained from the script using the

Figure 3: Current sampling of the operation



Source: Author

same methodology of dividing the total time detected by the script by the number of iterations. We have checked that this methodology matches with the benchmarked run times using the TinyOS timer in software.

1.5 Original Contributions

In the context of $\mathcal{MQ}$ signatures, a different approach to parametrize UOV/Rainbow signature schemes is described, namely, adopting private keys that display a certain blockwise ring symmetry which is preserved when computing the associated public key. Thus, in contrast to previous techniques in the literature, the new construction builds both private and public keys as collections of elements from a suitable more compact matrix ring – hence the scheme name, *RingUOV/RingRainbow*. As a result, smaller keys are achieved compared to the current techniques (PETZOLDT; BUCHMANN, 2009; PETZOLDT; BULYGIN; BUCHMANN, 2010a; PETZOLDT et al., 2011), but unfortunately a negative result was obtained when adopting very compact matrix rings (a polynomial time attack is found). Subsequently, a less compact matrix ring is described which provides at most twice smaller keys than the previous in the literature, but this improvement is not sufficient for practical deployment on constrained embedded platforms, and the overall security still casts doubt.

On the other hand, in the context of hash-based signature schemes, this thesis contributes with an efficient signature scheme that not only yields shorter signatures than the previous state of the art, but also enables faster key generation, signature generation and verification for the same security level and word size parameters. The resulting scheme is also less energy consuming. We provide a practical implementation for a very constrained microcontroller, the ATmega128L (@7.37MHz, 4KiB SRAM and 128KiB ROM), and argue that the resulting scheme is very suitable for constrained platforms typical of the IoT, as well as similar or related scenarios, like wireless sensor networks and intelligent habitats and environments. Even for this type of platform the resulting signing times are about $0.3s$, verification times $0.1s$ and the RAM consumption is about $2.5 - 3KiB$. Interestingly, in this particular platform the energy consumption is proportional to the run time of the algorithms, so energy is reduced by a proportional factor. A software library in C language was produced and is considered an important by-product of this work.

During his PhD investigations, the author has also achieved results in subjects not directly related to post-quantum digital signatures. They are briefly described in Appendix 5.1. These works are extensions of the author's MSc research regarding elliptic curve cryptography. In (PEREIRA et al., 2013), an efficient framework is designed for secure SMS transmission over embedded devices using asymmetric cryptography based on elliptic curves. In another work (BARRETO et al., 2015), the authors discovered some weaknesses in many elliptic curves widely adopted in the literature, which are related to subgroup security in pairing-based cryptography. In addition to pinpointing these weaknesses, the authors were able to suggest curves for replacement having small computation overheads compared to the previous weak curves.

1.6 Organization

Chapter 2 describe some basic concepts related to digital signatures. Then, Chapter 3 reviews the related $\mathcal{MQ}$ digital signatures in the literature and describe a new flexible technique for reducing key sizes, called RingUOV/RingRainbow. The security of the new constructions is also analyzed with practical experiments involving the known attacks.

Chapter 4 focuses on reducing signature sizes in hash-based schemes and evaluating their practicality on very constrained platforms through software implementation and benchmarks.

Finally, we make our conclusions and suggest open problems and future works in Chapter 5.

2 BACKGROUND AND CONCEPTS FOR DIGITAL SIGNATURES

Many of the definitions and notations herein follow the book of Hankerson *et. al.* (HANKERSON; MENEZES; VANSTONE, 2006) and the book chapter (BARRETO et al., 2014).

One-way function. Let $f : A \to B$ be a function. f is said to be one-way if:

1. Given $x \in A$, it is computationally easy to compute $y = f(x)$.
2. Given $y \in_f B$, it is computationally hard to compute $x \in A$ such that $f(x) = y$.

In this definition, $\in_f$ means that y is "randomly" chosen in $f(A)$, where "randomly" means here that the probability of obtaining a value y is exactly:

$$\frac{|\{x \in A, f(x) = y\}|}{|A|}$$

One-way trapdoor function. Let $f : A \to B$ be a function. f is said to be one-way trapdoor if:

1. f is a one-way function.
2. There is a secret piece of information s such that, given $y \in_f B$ and s, it is easy to compute $x \in A$ such that $f(x) = y$.

One-way hash function. A one-way hash function f is a one-way function with the additional property that it can map arbitrarily large inputs onto much smaller ones of a fixed size. Thus, a hash function is many-to-one, implying that the existence of collisions is unavoidable.

Cryptographic hash functions are used in security applications such as digital signatures, identification schemes, key derivation, among others. Formally, a hash function $f : \{0,1\}^* \rightarrow \{0,1\}^n$ takes as input an arbitrarily long string m and returns a fixed string r of size n, the *hash value* (that is, $r = h(m)$).

Since the image $\{0,1\}^n$ of f is a subset of $\{0,1\}^*$, it is easy to see that more than one message is mapped to the same hash value (or digest). Some applications require that it be computationally infeasible for an attacker to find two random messages that generate the same digest; one such example is that of digital signatures, in which the hash of messages are signed, not the messages themselves.

The primary properties that a hash function $f : \{0,1\}^* \longrightarrow \{0,1\}^n$ should possess are preimage resistance, second-preimage resistance and collision resistance.

- *Preimage resistance:* Let r be a known digest. Then, it should be infeasible to find a value m such that $f(m) = r$.
- *Second preimage resistance:* Let m be a known message. Then, it should be infeasible to find m' such that $m' \neq m$ e $f(m') = f(m)$.
- *Collision resistance:* It should be infeasible to find a pair $m, m' \in \{0,1\}^*$ such that $m' \neq m$ and $f(m') = f(m)$.
- *Undetectability* Some signature schemes consider an additional property called undetectability. A function is called undetectable if an adversary cannot distinguish the output from f with a uniform distribution over its range. In practice, this means that, given some n-bit element, an adversary is not able to tell whether f was applied to it or not.

Another desirable property for practical applications, is that the hash function be efficient (speed, memory, energy, etc.) when implemented on various computing platforms (hardware and/or software). It is easy to see that a function that is collision resistant is also second preimage resistant, but the reciprocal is not necessarily true.

Digital signature. Digital signatures were started by Diffie and Hellman (DIFFIE; HELLMAN, 1976) where a signature scheme based on a *trapdoor one-way function* was provided. The intuition is that given a message m as being the image of some function f hard to invert, compute $f^{-1}(m)$ using s. Digital signature schemes can be used to provide the following basic cryptographic services:

- data integrity: the assurance that data has not been altered by unauthorized or unknown means.
- origin authentication: the assurance that the source of data is as claimed.
- non-repudiation: the assurance that an entity cannot deny previous actions or commitments.

Ideally, a digital signature scheme should be *existentially unforgeable under chosen-message attack*. This is a very important notion of security introduced by Goldwasser et al. (GOLDWASSER; MICALI; RIVEST, 1988). It asserts that an adversary who is able to obtain some entity A's signatures for any message of its choice is unable to successfully forge A's signature on a single different message.

Below we give the description of a generic digital signature scheme as a set of four algorithms (Setup, KeyGen, Sign, Verify):

- Setup: with an input 1^k for a certain security parameter k, the output is a set params of parameters for the signature scheme attaining the desired security level.
- KeyGen: Given a set params of parameters, generates the key pair (pk, sk).

- Sign: given a message m and the signer's private key sk, computes the signature σ for m with the key sk.
- Verify: given a message m, a signature σ and a public key pk, the output is 1 if σ is a valid signature for m with the key pk, and 0 otherwise.

Pseudorandom function family (PRF) The notion of a pseudorandom function family was proposed by Goldreich *et. al.* (GOLDREICH; GOLDWASSER; MICALI, 1986). Pseudorandom functions exist if one-way functions exist, which in turn exist if secure digital signatures exist (ROMPEL, 1990, Lemma 1). So pseudorandom functions exist if secure digital signatures exist.

A pseudorandom function family is a length preserving family of functions defined as

$$F(n) = \{f_k : \{0,1\}^n \to \{0,1\}^n \mid k \in \{0,1\}^n\}$$

where the key k selects a specific function member from the family. A PRF has the property that its output can not be distinguished in polynomial time from a function which is chosen randomly from the set of all functions with the same input and output length.

In the original construction, the iterative application of f_k consists of using its output as input for the next iteration and the key remains fixed, and the notation $f_k^i(x)$ denotes that the function is iterated i times on input x using key k for the first iteration. Thus, we have $f_k^2(x) = f_k(f_k(x))$ and $f_k^0(x) = x$.

There is an alternative notation that appears in the literature for the iteration $f_k^i(x)$ where the output of the function is used as key for the next iteration instead of as input, i.e. $f_k^2(x) = f_{f_k(x)}(x)$ and $f_k^0(x) = x$.

There are two main desired properties for PRFs:

- *second key resistance*: given key k and preimage x, it is hard to find a key $k' \neq k$ such that $f_k(x) = f_{k'}(x)$.
- *key collision resistance*: given preimage x, it is hard to find two distinct keys k, k' such that $f_k(x) = f_{k'}(x)$.

Group. A group G is a structure (G,$\star$), constituted by a set $G = a, b, c...$ endowed with a binary operation ($\star$), and its elements satisfy the following properties:

i Closure: $\forall a, b \in G, a \star b \in G$

ii Associativity: $\forall a, b, c \in G, (a \star b) \star c = a \star (b \star c)$

iii Neutral Element: $\exists e \in G$, such that: $a \star e = e \star a = a, \forall a \in G$

iv Inverse Element: $\forall a \in G, \exists a^{-1}$ such that: $a \star a^{-1} = a^{-1} \star a = e$.

Note that the first property comes from the definition of the binary operation. In the case that $\forall a, b \in G$ we have $a \star b = b \star a$ and the group is said to be *commutative* or *abelian* (in honor of Niels Abel who first worked with these structures).

Ring. A ring A is an abelian group endowed with a second binary operation ($*$) which is distributive over the underlying abelian group binary operation, and satisfies:

i A is an abelian group with respect to addition (and thus, A has a neutral element, denoted 0, and $\forall x \in A$ has an additive inverse, denoted by $-x$,

ii multiplication is associative and distributive with respect to addition,

iii $x * y = y * x, \forall x, y \in A$,

iv there exists $1 \in A$ such that $1 \neq 0$ and $x * 1 = 1 * x = x, \forall x \in A$. This identity element is unique.

Matrix Ring. Another important concept used in Chapter 3 is the definition of *matrix rings*. A ring R is called a matrix ring if, for some ring K and some positive integer n,

$$R \cong K^{n\times n}. \tag{2.1}$$

where $K^{n\times n}$ denoted the set of matrices $n \times n$ with components in K.

3 MULTIVARIATE QUADRATIC DIGITAL SIGNATURES

Multivariate Public Key Cryptosystems (MPKC) constitute one of the main public key families considered potentially resistant against the powerful quantum computers. The security of MPKC schemes is based upon the difficulty of solving a random non-linear system of equations over finite fields. In particular, in most cases, such schemes are based upon multivariate systems of *quadratic* (instead of higher degrees) equations because of computational advantages. Quadratic equations can always be obtained from higher degree equations by applying degree reduction. The generic underlying problem is known as Multivariate Quadratic Problem or $\mathcal{MQ}$ Problem, and it was shown to be NP-complete by Patarin (PATARIN; GOUBIN, 1997). MPKC has been developed more intensively in the last two decades. It was shown that, in general, encryption schemes were not as secure as it was believed to be, while digital signatures can be considered viable.

3.1 Preliminaries and Notation

For this chapter we define the hash function that computes documents' digests that will be in fact signed, by $H : \{0,1\}^* \rightarrow \mathbb{F}_q^m$. The expression $\mathbf{x} = [x_1, \ldots, x_n]$ will denote a row vector whose components are the variables x_i and, $\mathbf{x}^{\mathrm{T}}$ will be the transpose of $\mathbf{x}$, which in turn is a column vector.

Matrices will be denoted by capital letters. And more generally, for a finite set of matrices, an element in the k-th matrix of this set, located at the i-th row and j-th column will be denoted by $P_{ij}^{(k)}$.

The idea behind MPKC is to define a trapdoor one-way function whose image is a non-linear random-looking system of multivariate equations over a finite field. A typical MPKC public key is given by a set $\mathcal{P}$ of polynomials having the form

$$\mathcal{P} = \{p_1(x_1, \dots, x_n), \cdots, p_m(x_1, \dots, x_n)\}$$

where each p_k is a non-linear polynomial (usually quadratic) in variables $(x_1, \cdots, x_n)$:

$$p_k(x_1, \dots, x_n) := \sum_{1 \le i \le j \le n} P_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} L_i^{(k)} x_i + c^{(k)}, 1 \le k \le m \tag{3.1}$$

where all the coefficients and variables are in $\mathbb{F}_q$.

In order to make the previous definition simpler, we will adopt vector notation, which is closer to practical implementations:

$$p_k(\mathbf{x}) := \mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}} + L^{(k)} \mathbf{x}^{\mathrm{T}} + c^{(k)}, 1 \le k \le m \tag{3.2}$$

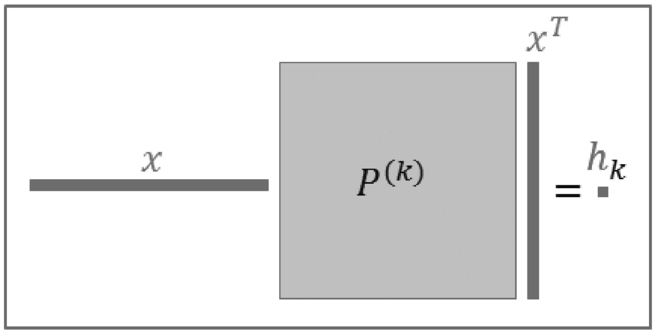
where $P^{(k)} \in \mathbb{F}_q^{n \times n}$ is a $n \times n$ matrix, whose entries are the coefficients of the quadratic terms in $p_k(x_1, \dots, x_n)$. $L^{(k)} \in \mathbb{F}_q^n$ is a vector whose entries are the coefficients of the linear terms in $p_k(x_1, \dots, x_n)$ and $c^{(k)}$ denotes the constant term of $p_k(x_1, \dots, x_n)$. Figure 4 illustrates the pure quadratic transformation (or map) $\mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}}$ whose evaluation provides a certain element of the finite field denoted by $h_k \in \mathbb{F}_q$.

In simple terms, the $\mathcal{MQ}$ problem consists of determining whether or not a given system of multivariate quadratic equations over a finite field admits a solution over that field. A formal definition for the search version of the $\mathcal{MQ}$ Problem is given as follows.

Definition 1 (**$\mathcal{MQ}$ Problem**). *Solve a uniformly random generated system $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x}) = \cdots = p_m(\boldsymbol{x}) = 0$, where each p_i is quadratic in variables $\boldsymbol{x} = [x_1, \dots, x_n]$ and defined as in Equation 3.1. All coefficients and variables are in $K = \mathbb{F}_q$, the finite field*

Figure 4: Pure quadratic map or transformation

$$x\,P^{(k)}\,x^T = h_k \quad (k = 1, \ldots, m)$$



Source: Author

with q elements.

In other words, the target of the $\mathcal{MQ}$ Problem is to find a solution **x** for a given map $\mathcal{P} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$. In 1979, Garey and Johnson proved (GAREY; JOHNSON, 1979, page 251) that the decision variant of the $\mathcal{MQ}$ Problem over finite fields is NP-complete. Unfortunately, the result of Garey and Johnson provides only a worst-case difficulty for the decisional $\mathcal{MQ}$ problem (BELLARE; GOLDWASSER, 1994). Actually, the complexity of the $\mathcal{MQ}$ problem strongly depends on the relation between n and m, and the actual structure of the polynomials.

In this context, it is worth to mention that the proposed $\mathcal{MQ}$ signature schemes in the literature do not rely their security exactly on the original $\mathcal{MQ}$ Problem. Actually, the security of the one-way trapdoor function constructed in practice, which does not employ completely random systems but adds some structure to them, is based on a related problem called the Isomorphism of Polynomials Problem or *IP* Problem, proposed by Patarin (PATARIN, 1996) which has the following definition:

Definition 2 (**Isomorphism of Polynomials Problem**). *Let $m, n \in \mathbb{N}$ be arbitrarily fixed. Further denote $\mathcal{P}, \mathcal{Q} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ two multivariate quadratic maps and $\mathsf{T} \in \mathbb{F}_q^{mxm}$, $\mathsf{S} \in \mathbb{F}_q^{nxn}$ two bijective linear maps, such that $\mathcal{P} = \mathsf{T} \circ \mathcal{Q} \circ \mathsf{S}$. Given $\mathcal{P}$ and $\mathcal{Q}$, find T and*

S.

In other words, the IP Problem goal is to find T and S for a given pair $(\mathcal{P}, \mathcal{Q})$. Note that, originally, S was defined as an affine instead of linear transformation (PATARIN; GOUBIN; COURTOIS, 1998). But, Braeken *et al.* (BRAEKEN; WOLF; PRENEEL, 2005, Sec. 3.1) noticed that the constant part is not important for the security of certain $\mathcal{MQ}$ schemes and thus can be omitted.

3.2 Construction of $\mathcal{MQ}$ Key Pairs

Generically, a typical $\mathcal{MQ}$ private key is constituted of two linear transformations T and S along with a quadratic transformation $\mathcal{Q}$. Note that $\mathcal{Q}$ will present certain particular trapdoor structure. Two distinct trapdoor structures are presented in Section 3.3 for the UOV and Rainbow signature schemes (KIPNIS; PATARIN; GOUBIN, 1999; DING; SCHMIDT, 2005), which are the most promising ones.

The trapdoor structure will allow the signer to easily solve the public $\mathcal{MQ}$ system in order to generate valid signatures. The public key is simply given by the composition $\mathcal{P} = \mathsf{T} \circ \mathcal{Q} \circ \mathsf{S}$. For some signature schemes it is not necessary to explicitly use the map T, since it is reduced to the identity (BERNSTEIN; BUCHMANN; DAHMEN, 2009, Chapter 6).

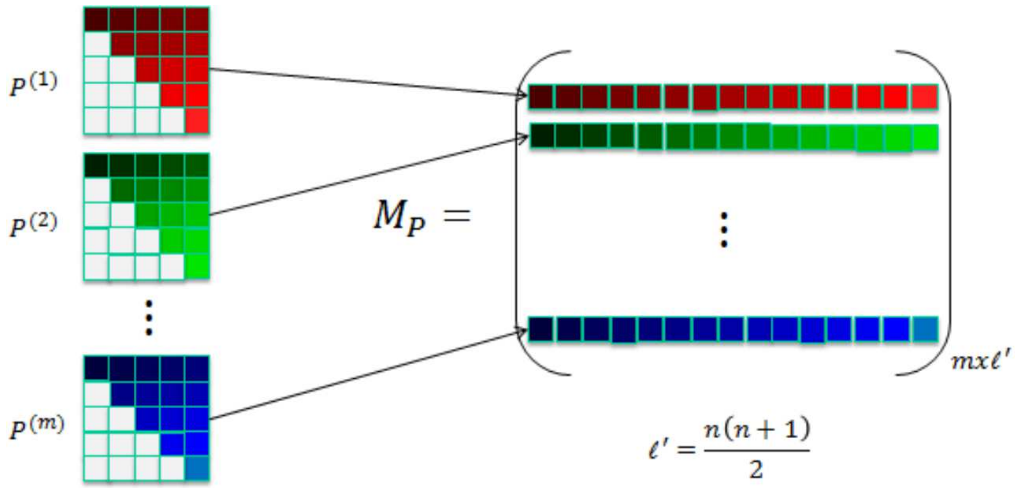
The main difference among distinct $\mathcal{MQ}$ signature schemes falls in the trapdoor structure of $\mathcal{Q}$. Since public keys have the same structure in most schemes, verifying a signature follows the same procedure, in other words, just checking if a given signature $\mathbf{x}$ is a solution of a public quadratic system $p_k(\mathbf{x}) = h_k, 1 \leq k \leq m$. For other trapdoor constructions the reader can see for example (WOLF; PRENEEL, 2005).

It is worth to mention an obvious optimization in the public matrices $P^{(k)}$ defined over fields of odd characteristic, that provides a reduction by a factor about two in the space representation. From the definition of the summation of the quadratic part of

$p_k(x_1, \ldots, x_n)$ (Equation 3.1), the coefficient of the term $x_i x_j$ is $P^{(k)}_{ij} + P^{(k)}_{ji}$, thus, one can update the coefficient $P^{(k)}_{ij}$ with the value $P^{(k)}_{ij} + P^{(k)}_{ji}$ and the coefficient $P^{(k)}_{ji}$ with zero for $i \leq j \leq n$, what makes the matrix $P^{(k)}$ upper triangular, thus storing about half of the size of the full matrix.

After turning matrices $P^{(k)}$ into upper triangular form, one is able to define a unique public matrix called the *public matrix of coefficients*, denoted M_P. Each row of M_P is given by the linearization of the coefficients of each upper triangular matrix $P^{(k)}$, by simply transporting them to a row vector. Figure 5 illustrates this construction. The form M_P will be useful for viewing the set of public matrices as a single one.

Figure 5: Public matrix of coefficients M_P



Source: Author

Now we present the UOV signature scheme and its more efficient generalization, the Rainbow signature scheme.

3.3 UOV and Rainbow $\mathcal{MQ}$ Signatures

One of the main still secure $\mathcal{MQ}$ signature families is the Unbalanced Oil and Vinegar (UOV) construction which was proposed by Patarin (KIPNIS; PATARIN; GOUBIN, 1999). The name Oil and Vinegar came from the fact that variables $(u_1, \cdots, u_n)$ of a certain quadratic private system can be separated in two subsets $O = (u_1, \cdots, u_m)$ and $V = (u_{m+1}, \cdots, u_n)$, in such a way that the oil and vinegar variables are never mixed in the quadratic terms (like the disposition of the oil and vinegar mixture in a salad dressing).

Formally, the trapdoor structure consists of defining a purely quadratic map, called the *central map*, $\mathcal{Q} : \mathbb{F}^n \to \mathbb{F}^m$ with the following structure

$$\mathcal{Q} = \{f_1(u_1,\ \ldots,\ u_n), \ldots, f_m(u_1,\ \ldots,\ u_n)\}$$

and

$$f_k(u_1, \ldots, u_n) := \sum_{1 \le i \le j \le n} Q^{(k)}_{ij} u_i u_j \equiv \mathbf{u} Q^{(k)} \mathbf{u}^{\mathrm{T}} \tag{3.3}$$

The central map has a special restriction in its polynomials $f_k(u_1, \ldots, u_n)$. It is imposed that a certain part of its coefficients be zeros. The set of variables $\mathbf{u}$ is divided in two subsets, the one of vinegar variables u_i with $i \in V = \{1, \cdots, v\}$ and the one of oil variables u_i with $i \in O = \{v+1, \cdots, n\}$ of $m = n - v$ elements. The restriction on the polynomials f_k is that they have no term combining any two oil variables. That assures that we do not have quadratic (or crossed) terms in oil variables. Thus, we only have terms combining the following sort of variables $v \times v$ and $o \times v$.

Patarin showed that given the above construction one can fix arbitrary values for the vinegar variables, resulting in a linear system in the oil variables. This remaining linear system will have a solution with high probability, i.e. $1 - 1/q$, and can be solved using gaussian elimination with complexity $\mathcal{O}\left(n^3\right)$. The structure of the private polynomials is the following:

$$f_k(u_1, \cdots, u_n) := \sum_{i,j \in V,\ i \le j} Q^{(k)}_{ij} u_i u_j + \sum_{i \in V,\ j \in O} Q^{(k)}_{ij} u_i u_j \tag{3.4}$$

In order to generate a signature $\mathbf{x} \in \mathbb{F}_q^n$ of a given message M, particularly of its hash $h = H(M) \in \mathbb{F}_q^m$, the signer is required to invert the map $\mathcal{P}(\mathbf{x}) = \mathcal{Q}(\mathcal{S}(\mathbf{x})) = h$.

Defining $\mathbf{x'} = \mathbf{x} \cdot S$, one first solves the multivariate system, $\mathbf{x'}Q^{(k)}\mathbf{x'}^{\mathrm{T}} = h_k$, $1 \leq k \leq m$, finding $\mathbf{x'}$. Finally, the signature $\mathbf{x} = \mathbf{x'}S^{-1}$ is computed.

As explained before, the structure of the matrices $Q^{(k)}$ allows to efficiently solve the $\mathcal{MQ}$ system, by choosing v vinegar variables at random and then solving the resulting system for the remaining m oil variables. If the linear system has no solution, repeat the process by choosing new vinegar variables until it has a valid solution. The signature generation is illustrated in Algorithm 3.3.1. The verification procedure is also illustrated in Algorithm 3.3.2.

Algorithm 3.3.1 Signature Generation

```
Require: To sign a message $M$ solve the equation $P(x) = T \circ Q \circ S(x) = h$, where
    $h = H(M)$
Ensure: A valid signature $\mathbf{x}$
1: Compute the hash $h = H(M) \in F_q^m$ with a secure hash function $H : \{0,1\}^* \rightarrow \mathbb{F}_q^m$
2: Compute $y = T^{-1}(h)$
3: Randomly choose $(u_1, \cdots, u_v) \overset{\$}{\leftarrow} \mathbb{F}_q^v$
4: Compute $\mathbf{x'}$ such that $\mathbf{x'} = Q^{-1}(y)$ (using Gaussian Elimination). If no solution is
    found go to step 3
5: Compute $\mathbf{x} = S^{-1}(\mathbf{x'})$
6: The signature of $M$ will be $\mathbf{x} \in \mathbb{F}_q$, in which $Q^{-1}(y)$ means computing one among
    the possible preimages of $y$
```

Algorithm 3.3.2 Signature Verification

```
Require: Receives a signature $\mathbf{x} \in \mathbb{F}_q$ and public key $\mathcal{P}$, this algorithm checks if
    $P(\mathbf{x}) = 0$
Ensure: Valid or Invalid.
1: For each $1 \leq k \leq m$, check that $\mathbf{x}P^{(k)}\mathbf{x}^{\mathrm{T}} = 0 \in \mathbb{F}_q$.
2: If $\mathbf{x}P^{(k)}\mathbf{x}^{\mathrm{T}} \neq 0$ for any $k$ return Invalid.
3: return Valid.
```

Verification. A signature $\mathbf{x}$ for $h = H(M)$ is valid, if and only if, all polynomials p_k constituting the public key have their evaluation satisfied, i.e. $p_k(x_1, \cdots, x_n) = \mathbf{x}P^{(k)}\mathbf{x}^T = h_k$, $1 \leq k \leq m$.

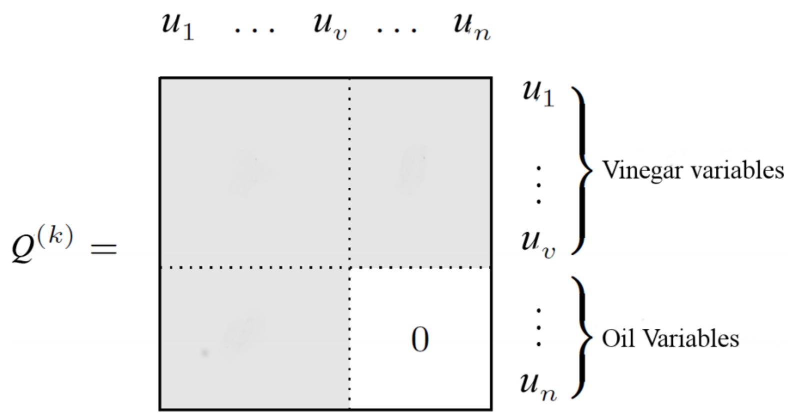
The consistency of the verification $P(\mathbf{x}) \overset{?}{=} h$ is described next

$$\begin{aligned}
p(\mathbf{x}) &= \mathbf{x}P\mathbf{x}^{\mathrm{T}} \\
&= \mathbf{x}(Q \circ S)\mathbf{x}^{\mathrm{T}} \\
&= \mathbf{x}(SQS^{\mathrm{T}})\mathbf{x}^{\mathrm{T}} \\
&= (\mathbf{x}'S^{-1})(SQS^{\mathrm{T}})(\mathbf{x}'S^{-1})^{\mathrm{T}} \\
&= \mathbf{x}'(S^{-1}S)Q(S^{\mathrm{T}}(S^{-1})^{\mathrm{T}})\mathbf{x}'^{\mathrm{T}} \\
&= \mathbf{x}'IQI\mathbf{x}'^{\mathrm{T}} \\
&= \mathbf{x}'Q\mathbf{x}'^{\mathrm{T}} \\
&= h.
\end{aligned}$$

Historically, UOV signatures were proposed to fix a security problem in the original *Oil and Vinegar* (OV) construction (PATARIN, 1997), for which the number of vinegar and oil variables are the same, i.e. balanced oil and vinegar. Unfortunately that construction was shown insecure (KIPNIS; SHAMIR, 1998). Next, a new modification was introduced to make it secure by unbalancing the amount of each subset ($v > m$), what originated the Unbalanced Oil and Vinegar (UOV) signature (KIPNIS; PATARIN; GOUBIN, 1999). Figure 6 illustrates the structure of each UOV private quadratic transformation.

In order to hide the trapdoor structure at polynomials f_k, an invertible linear transformation $\mathsf{S} \in \mathbb{F}_q^{n\times n}$ is applied to the right of $\mathcal{Q}$. So the resulting public map is $\mathcal{P} = \mathcal{Q}\circ\mathsf{S}$. The private key is given by the pair $sk := (\mathcal{Q}, \mathsf{S})$ and the public key is composed of polynomials $\mathcal{P} := p(x_1, \cdots, x_n) = \{p_1(x_1, \cdots, x_n), \cdots, p_m(x_1, \cdots, x_n)\}$. So, it becomes clear that the security of the system is not directly based on the $\mathcal{MQ}$ Problem and indeed recovering the private key is related to the difficult to decompose $\mathcal{P}$ in $\mathcal{Q}$ and S, in other words, to solve the IP Problem. For the UOV scheme, $v > o$, (KIPNIS; PATARIN;

Figure 6: UOV central map



Source: Adapted from (CZYPEK; HEYSE; THOMAE, 2012)

GOUBIN, 1999) showed that a specific attack has a complexity of roughly $O\left(q^{v-o-1}o^4\right)$, when $v \approx o = n - v$. This means, that if o is not too large (< 100) and a given fixed field of size q, then $v - o$ should be large enough, but also not too large, to ensure the security of the scheme.

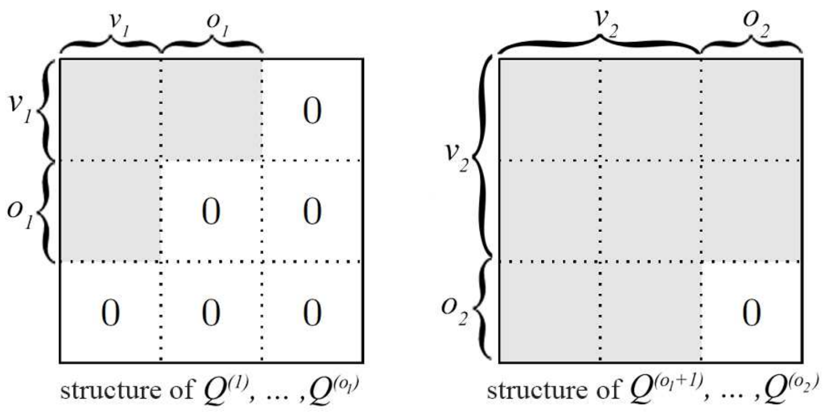
However, one must notice that in this scheme the message to be signed is a vector in $\mathbb{F}_q^o$ and the signature is a vector in $\mathbb{F}_q^{o+v}$. This means that the signature is at least twice the size of the message and with a large $v + o$ the system becomes less efficient.

An improved variant of the UOV scheme is the Rainbow (DING; SCHMIDT, 2005) signature scheme. Rainbow was proposed by Ding and Schmidt, whose main advantage are the shorter signature footprints attained compared to UOV (THOMAE, 2012a, Section 3).

The basic idea behind Rainbow signature is to separate the m private UOV equations in smaller ranges or bands and partitioning the variables accordingly, in other words, each band has its own *oil* and *vinegar* variables. After a band is processed, all of its variables become the vinegars for the next band and so on until the last band is processed.

Typically the central map is divided in only two bands, since this configuration has been shown the most suitable in the sense that it avoids certain structural attacks and keeps the signatures reasonably short (THOMAE, 2012a).

Figure 7: Rainbow central map $\mathcal{Q}$ with two bands


v_1
o_1
v_1
o_1
0
0
0
0
0
0
structure of $Q^{(1)}, \ldots, Q^{(o_1)}$
v_2
o_2
v_2
o_2
0
structure of $Q^{(o_1+1)}, \ldots, Q^{(o_2)}$

Source: Adapted from (CZYPEK; HEYSE; THOMAE, 2012)

Rainbow central map $\mathcal{Q}$ with two bands for example is divided in two layers as shown in Figure 7 where v_1 and o_1 are the number of vinegars and oils of the first layer and v_2 and o_2 are the number of vinegars and oils of the second layer. Note that $v_2 = o_1 + v_1$.

The signature procedure is similar to UOV procedure, choosing vinegars at random for the first band in order to be able to compute the oil variables as in UOV. Then, these computed variables (vinegars plus oils) are used as vinegars for the next layer.

To give an idea of the efficiency of these signature schemes, a comparison among the different parameter sizes will be provided at the end of the Section 3.9.3.

3.4 The Cyclic UOV Signature

An interesting step toward the reduction of UOV/Rainbow key sizes was made by means of the Cyclic UOV/Rainbow constructions (PETZOLDT; BUCHMANN, 2009; PETZOLDT; BULYGIN; BUCHMANN, 2010a). Petzoldt *et. al.* noticed the existence of a linear relation between part of the public quadratic map and the private quadratic map.

That relation was exploited in order to construct key pairs in a different unusual way being able to reduce the public key size. The idea is to first generate the quadratic part of the public key with a desired compact structure and then computing the quadratic part of the private key by using the linear relation.

Therefore, it is possible to obtain shorter public keys while the private ones remain random looking and without any apparent structure. The structure suggested by Petzoldt *et. al.* was the one that uses circulant matrices, that is the origin of the name Cyclic UOV (PETZOLDT; BUCHMANN, 2009). Circulant matrices are very compact, since they can be represented by simply their first row. Thus, the public key can be stored in an efficient manner, apart from some advantages in processing like Karatsuba and Fast Fourier Transform (FFT) techniques.

Cyclic UOV keys are constructed as follows. Firstly, one generates an invertible linear transform $\mathsf{S} \in \mathbb{F}_q^{n\times n}$, where $S_{ij} \overset{\$}{\leftarrow} \mathbb{F}_q, 1 \leq i, j \leq n$ and, from S, one computes the aforementioned linear relation and denoted by $A_{UOV} := \alpha_{ij}^{rs}$:

$$\alpha_{ij}^{rs} = \begin{cases} S_{ri} \cdot S_{si} i = j \\ S_{ri} \cdot S_{sj} + S_{rj} \cdot S_{si}, \text{ otherwise} \end{cases}$$

In order to illustrate how the public and private matrices of coefficients, M_P and M_F, are related, we have initially Figures 8 and 9 that separate the proper parts of these matrices.

Figure 8: Cyclic UOV – **Public** matrix of coefficients

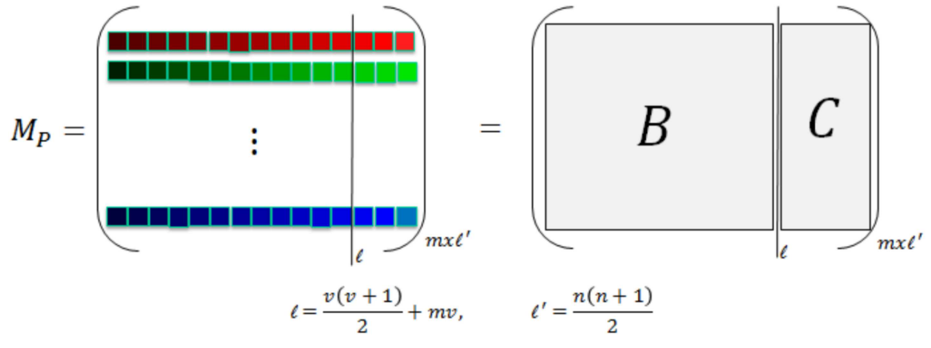
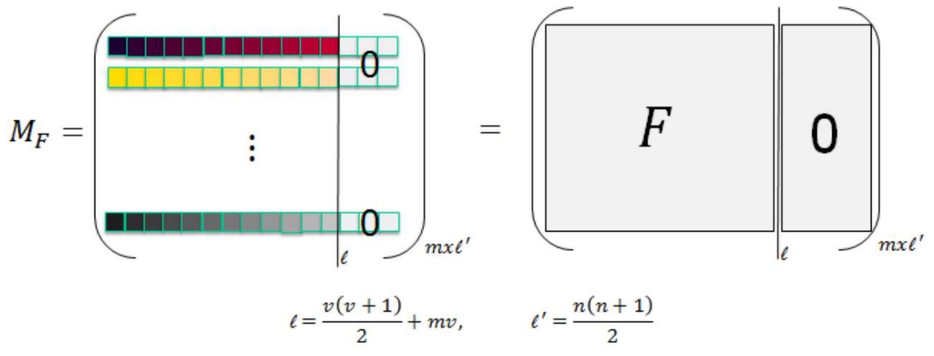


Figure 9: Cyclic UOV – **Private** matrix of coefficients



$M_F =$
0
0
ℓ
$m x \ell'$
$=$
F
0
ℓ
$m x \ell'$
$\ell = \frac{v(v+1)}{2} + mv, \qquad \ell' = \frac{n(n+1)}{2}$

Source: Author

Blocks B of M_P and F of M_F obey the relation $B := F \cdot A_{UOV}(S)$. Thus, for the key generation, one may first generate matrix M_P with B with circulant structure and then computing $F := B \cdot A^{-1}_{UOV}(S)$. That methodology was able to reduce UOV public key size in about 6 times for the security level of 80 bits.

As mentioned above, $\mathcal{MQ}$ signatures have been developed more intensively in the last two decades. Many constructions have been proposed toward key size reduction which is the main disadvantage today.

3.5 A New Approach to Reduce UOV Key Sizes Exploiting Matrix Rings

Differently from the original UOV/Rainbow constructions the new idea does not consider keys as a set of generic matrices containing random elements. Actually, it views each matrix as a block matrix whose entries are another block submatrices from a certain matrix ring of compact representation. Matrix ring operations that preserve the compact structure allow that the block-oriented structure propagates from the private key to the public key. In practice, in order to apply the composition $P^{(k)} = SQ^{(k)}S^T$, when S and $Q^{(k)}$ are block matrices as suggested, multiplication is performed by blocks and not simply as independent rows and columns. Thus, when two blocks from the ring are multiplied, the structure is preserved in the resulting block of the public key. In addition to multiplication, the operations of sum, subtraction, inversion and transposition must also preserve the structure by blocks, which is not satisfied by triangular matrices for example as we will see later in this section.

Let $\mathbb{F}_q^{p\times p}$ denote the ring of the generic random $p \times p$-matrices. Under the new block matrix construction, matrices S and $Q^{(k)}$ are constituted by $n' \times n'$ blocks of dimension $p \times p$, where $n' = n/p$.

$$S = \left(\begin{array}{c|c|c} S_0 & \cdots & S_{n'-1} \\ \hline \vdots & \ddots & \vdots \\ \hline S_{(n'-1)n'} & \cdots & S_{n'^2-1} \end{array}\right), \quad Q^{(k)} = \left(\begin{array}{c|c|c} Q_0^{(k)} & \cdots & Q_{n'-1}^{(k)} \\ \hline \vdots & \ddots & \vdots \\ \hline Q_{(n'-1)n'}^{(k)} & \cdots & Q_{n'^2-1}^{(k)} = O^{o\times o} \end{array}\right)$$

where each $S_i, Q_i^{(k)} \in \mathbb{F}_q^{p\times p}$. The resulting public key $P^{(k)} = S\,Q^{(k)}S^T$ is

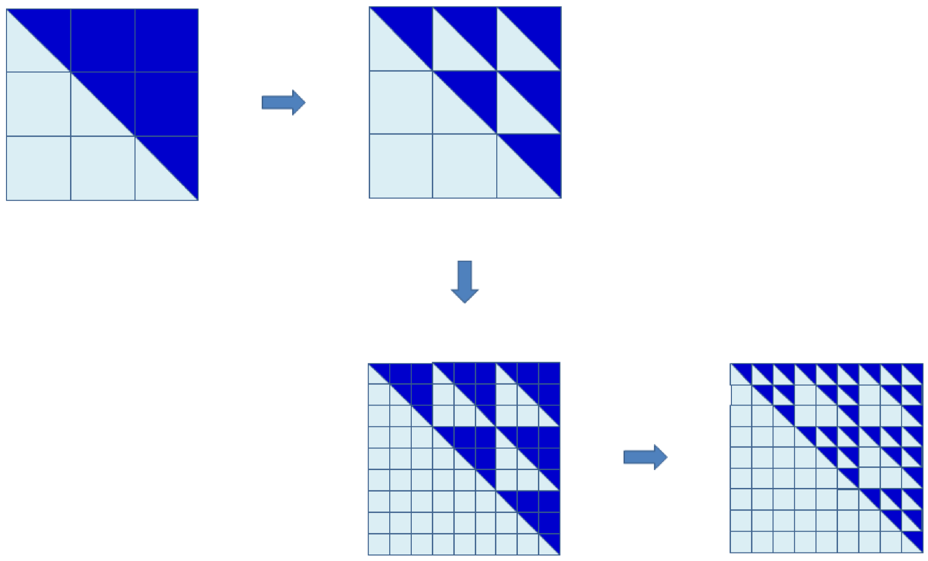
$$\left(\begin{array}{c|c|c} P_0^{(k)} & \cdots & P_{n'-1}^{(k)} \\ \hline \vdots & \ddots & \vdots \\ \hline P_{(n'-1)n'}^{(k)} & \cdots & P_{n'^2-1}^{(k)} \end{array}\right) = \left(\begin{array}{c|c|c} S_0 & \cdots & S_{n'-1} \\ \hline \vdots & \ddots & \vdots \\ \hline S_{(n'-1)n'} & \cdots & S_{n'^2-1} \end{array}\right)\left(\begin{array}{c|c|c} Q_0^{(k)} & \cdots & Q_{n'-1}^{(k)} \\ \hline \vdots & \ddots & \vdots \\ \hline Q_{(n'-1)n'}^{(k)} & \cdots & O^{o\times o} \end{array}\right)\left(\begin{array}{c|c|c} S_0 & \cdots & S_{n'-1} \\ \hline \vdots & \ddots & \vdots \\ \hline S_{(n'-1)n'} & \cdots & S_{n'^2-1} \end{array}\right)^T$$

Thus, at first sight, the more compact the ring structure $R \subset \mathbb{F}_q^{p\times p}$, the smaller the key sizes obtained.

It is possible to achieve even smaller sizes if each block submatrix is also from a compact ring, creating therefore a layered structure. Besides the property of compactness, another property a particular ring can have is more efficient underlying operations.

Figure 10 illustrates an apparently possible compact ring to be adopted, the upper triangular matrix ring: the most external matrix consists of $p \times p$ blocks; and each one of these are another triangular $p' \times p'$-matrix of smaller blocks, and so on, until reaching the most internal 1×1-block being a element of the underlying finite field (in the example, all the blocks have dimension 3×3). The lighter regions are zero elements; the total size occupied corresponds to the total area in darker color. An inherent problem of this matrix ring in our context is that when the operation $S^{\top}$ is applied the ring structure is broken in the resulting public key, thus losing the desired compactness. Therefore, our matrix rings candidates must be closed under transposition to be useful.

Figure 10: Multi-layered block matrix structure



Source: Author

Another obvious ring to be studied is the circulant matrix ring. Circulant matrices can be simply represented by their first row with p elements, which is by a factor p smaller than random matrices having p^2 independent elements. Thus, circulant matrices can be dealt and stored as vectors. In addition to the storage advantage, circulant matrices also offer opportunities for faster matrix arithmetic, including operations related to solving linear systems of equations of the type $Cx = y$, which is a highly used operation in multivariate digital signatures. The efficiency gain in circulant matrices is related to the fact that they belong to a *convolution ring*, which allows diagonalization and, thus, more efficient underlying operations (GULAMHUSEIN, 1973).

Convolution ring. Let R be a commutative ring and p a positive integer. The convolution ring of dimension p over R, denoted $C_p(R)$, is the set of the circulant matrices $p \times p$ with entries in R.

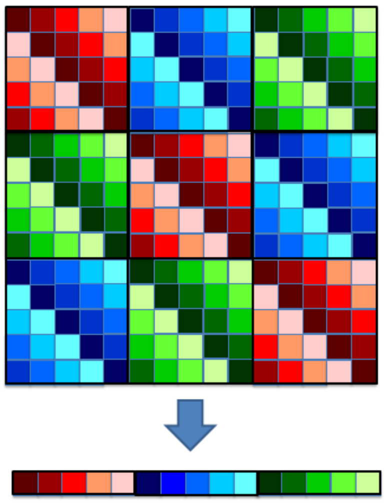
In general, a circulant $p \times p$-matrix is illustrated as the following

$$C = \begin{bmatrix} c_0 & c_{p-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{p-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{p-2} & & \ddots & \ddots & c_{p-1} \\ c_{p-1} & c_{p-2} & \cdots & c_1 & c_0 \end{bmatrix}_{p\times p}$$

where we adopt $c_i \in \mathbb{F}_q$.

In order to further reduce the key sizes, one can think about reproducing the above circulant structure for more than one layer. The ring R itself can be a convolution ring (of dimension q) over some other ring R', resulting in a convolution ring in two layers $C_p(C_q(R'))$, which is abbreviated as $C_{p,q}(R')$. More generally, we write $C_{p_1,\cdots,p_d}(R) := C_{p_1}(\ldots C_{p_d}(R)\ldots)$, a convolution ring of d layer and dimension $p_1 \cdots p_d$ over some ring R. The innermost ring will be usually a finite field $\mathbb{F}_q$. Figure 11 illustrates the circulant matrix ring in two layer, $C_3(C_5(R))$, with $p = 3$ and $q = 5$.

Figure 11: 2-layer circulant matrix ring



Source: Author

Note that instead of $15^2 = 225$ elements, the block matrix above needs only $3\times5 = 15$ elements to be stored and reconstructed later.

In the case where the innermost ring is a finite field, and if this field has a p_j-

th primitive root of unity ($j = 1, \ldots, d$), then the ring elements $C_{p_1,\cdots,p_d}(\mathbb{F}_q)$ can be diagonalized simultaneously. Although, multiplication and inversion in convolution rings have complexity no more than quadratic in the ring dimension, diagonalizable matrices reduce this complexity even further to $O(p \log p)$ in the ring dimension, $p = p_1 \cdots p_d$. Unfortunately, we will show a negative result for the circulant matrix ring in Section 3.7.

3.6 Circulant UOV and Rainbow Signatures

The new construction over the circulant matrix ring is easily plugged into $\mathcal{MQ}$ signature schemes such as UOV and Rainbow, resulting in the first variants of them that we call here RingUOV and RingRainbow. In the case of RingRainbow, it is possible to adopt an additional block structure for each layer, consisting of a particular non-trivial symmetry among distinct layers. Using this trick, a further reduction factor of $\approx 2p^2$ could be achieved. Public keys based on a layered circulant structure could get one order of magnitude smaller than the best previous sizes in the literature if shown secure.

Unfortunately, after a deeper security analysis the circulant matrix ring was shown to present too much structure so that one of the attacks performed in polynomial time. Afterward, we found a less compact matrix ring with an apparent better resistance to the attack (yet exponential), giving keys smaller than in the literature but much larger than the achieved from the circulant matrix ring. Next section describes the detected security problem on circulant matrices.

3.7 A Vulnerability on Simultaneously Diagonalizable Matrix Rings

Concerning the apparent advantage of the circulant rings, or more generally, of the simultaneously diagonalizable rings, we found that these rings are not adequate for cryptographic applications.

We have conducted practical experiments based on the reconciliation attack. In particular, the computational complexity observed from algorithms F_4 and F_5 for Groebner bases computation revealed a very efficient behavior for the circulant convolution ring. In fact, a polynomial behavior was observed – compared to an exponential asymptotic behavior exhibited against a plain Rainbow scheme. The same kind of behavior appears when the circulant ring is changed by others simultaneously diagonalizable rings, such as the negacyclic, cyclosymmetric, bisymmetric and others having linear number of independent elements.

The observed behavior can be explained with the following considerations. A generic random linear system with m equations and n variables has, usually, $O(n^2)$ distinct quadractic terms. However, if the blocks of the system come from a simultaneously diagonalizable ring, there exists a unique matrix which diagonalizes all the blocks. Meaning that it is possible to obtain from the public system, an equivalent system with all blocks in diagonal form. But the diagonal matrices constitute themselves a ring, and the elements of these blocks interact very weakly with each other because blocks are extremely sparse, thus reducing the number of quadratic distinct terms of the $\mathcal{MQ}$ system of equations corresponding to just $O(n)$. For example, a UOV system with parameters $(v = 2m, o = m)$ and $m \times m$ blocks would have, at most, only $6m = 2n$ distinct quadratic terms.

Unfortunately, this behavior is unavoidable, due to the own nature of the simultaneously diagonalizable rings. Consequently, the cost of the reconciliation attack is drastically smaller than the expected, except if p is very small, say $p = \mathcal{O}(\lg m)$. In this case, however, it is not possible to reduce key sizes by a big factor as initially desired.

It is worth mentioning that the same vulnerability would be present in any other $\mathcal{MQ}$ scheme, not just the ones from the UOV family, since the attack does not depend on the structure of the secret trapdoor, but only on the public key with blocks coming from a simultaneously diagonalizable ring.

In short, the "obvious" choice of the circulant rings and similar ones, that frequently occurs in lattice-based and code-based cryptosystems, was shown not to be adequate for $\mathcal{MQ}$ schemes.

3.8 RingRainbow/RingUOV over Alternative Matrix Rings

The weakeness of the simultaneously diagonalizable rings reminds a vulnerability detected in anomalous elliptic curves (SATOH, 1998; FREY; RÜCK, 1994), which immediately suggests the question: what rings are secure and efficient for $\mathcal{MQ}$ cryptosystems? Naturally, the generic ring $\mathbb{F}_q^{p \times p}$ is adequate in terms of security, since it directly represents the plain UOV/Rainbow schemes, in analogy to totally random elliptic curves. However, this trivial ring is clearly inefficient, as random elliptic curves.

Another possibility found, which partially answers the above question, is to adopt the centrosymmetric matrix ring. Centrosymmetric matrices are not simultaneously diagonalizable, differently from the circulant matrices; in fact, they are not even commutative, but since they constitute a ring, the composition $\mathcal{Q} \circ \mathcal{S}$ preserves the block structure, and gives a reduction of a constant factor ≈ 2. Thus, if a multi-layer structure could be securely chosen, one would get a total reduction of a factor $\approx 2^{\ell}$ for ℓ layers.

3.9 The Centrosymmetric Matrix Ring

Here we introduce a ring class that satisfies the ring conditions. One may call the resulting construction CentrosymmetricUOV/-Rainbow. Let J be the following matrix (called *exchange matrix*):

$$J = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

That is, J contains 1's in the secondary diagonal, and 0 in all other entries. The matrix J is an involution ($J^2 = I$) and symmetric ($J^T = J$).

Definition 3. *A* centrosymmetric *matrix is a matrix A that commutes with the exchange matrix J.*

Figure 12 shows the typical structure of a centrosymmetric matrix. Equal colors indicate identical elements of $\mathbb{F}_q$. The interest on centrosymmetric matrices for $\mathcal{MQ}$ cryptosystems arises from the Theorem 1.

Theorem 1. *Centrosymmetric matrices constitute a closed ring under the transposition operation.*

Proof. It is sufficient to show that the sum, product, neutral elements (additive and multiplicative), inverse elements (additive e multiplicative) and the transpose of centrosymmetric matrices are, in turn, centrosymmetric, i.e. commute with J.

In the following, for the pertinence properties, O denotes the zero matrix and I the identity matrix, both of appropriate dimensions. A and B are generic centrosymmetric:

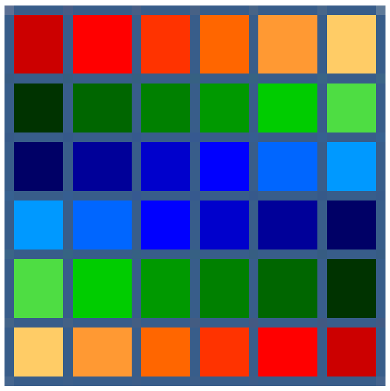
- (Pertinence of the sum): $(A + B)J = AJ + BJ = JA + JB = J(A + B)$.

- (Pertinence of the product): $(AB)J = A(BJ) = A(JB) = (AJ)B = (JA)B = J(AB)$.
- (Pertinence of the neutral elements): $OJ = JO$, $IJ = JI$.
- (Pertinence of the additive inverse): $(-A)J = -(AJ) = -(JA) = J(-A)$.
- (Pertinence of the multiplicative inverse, considering non-singular A): $AJ = JA \Leftrightarrow A^{-1}(AJ)A^{-1} = A^{-1}(JA)A^{-1} \Leftrightarrow (A^{-1}A)JA^{-1} = A^{-1}J(AA^{-1}) \Leftrightarrow JA^{-1} = A^{-1}J$.
- (Pertinence of the transpose): $A^TJ = (J^TA)^T = (JA)^T = (AJ)^T = J^TA^T = JA^T$.

It can be trivially proved one more property, that a scalar multiple of a centrosymmetric matrix is also centrosymmetric: $(cA)J = c(AJ) = c(JA) = J(cA)$. This further establishes that, more than a ring, centrosymmetric matrices constitute an *algebra* over $\mathbb{F}_q$, closed under transposition. $\square$

Experiments with the reconciliation attack were also conducted for the centrosymmetric ring. The resulting behavior of the attack is apparently exponential. This is not surprising, since the associated $\mathcal{MQ}$ system contains yet $O(n^2)$ distinct terms.

Figure 12: 6×6-Centrosymmetric matrix



Source: Author

Centrosymmetric matrices contain only a fraction $f(p) = \lfloor (p^2 + 1)/2 \rfloor / p^2$ of independent elements. More precisely, a Centrosymmetric $p \times p$-matrix contains

$\lfloor (p^2+1)/2 \rfloor$ independent elements. In the version with ℓ layers, supposing the simplest case where $m = p^\ell$, the public keys will have $|pk| = (p\lfloor (p^2+1)/2 \rfloor)^\ell$ elements of $\mathbb{F}_q$. This means that when using a centrosymmetric structure of ℓ-layers the key size reduction factor is about 2^ℓ compared to a random-matrix ring.

Example 1: Adopting $m' = p = \ell = 3$, a RingUOV public key, built with a centrosymmetric ring over $\mathbb{F}_{2^4}$ for the security level 2^{80} contains 3375 elements of the finite field, occupying only 1688 bytes $\approx$ 1.65 KiB. In contrast, the best result in the literature (Table 1) has keys of $\approx$ 10.2 KiB. The reduction in this example is by a factor 6.

It can be noticed from the Example 1, that keys based on the centrosymmetric ring (if shown secure) would fit in some embedded platforms. On the other hand, another security problem was found for multi-layered rings including the centrosymmetric one. When many layers are introduced, the dependency among elements of different layers is very high, giving a large amount of dependent elements in the full matrix. The attack is sensitive to this degree of dependency, i.e. the resulting degree of regularity decreases, improving the behavior of algorithms like F_5. The attacks are described in Section 3.9.2.

3.9.1 Security of CentrosymmetricUOV

In order to provide a fair comparison between UOV and Rainbow schemes, considering memory consumption and run time, firstly, one needs to choose parameters for the same security level. Therefore, we review the last attacks and the choices of parameters.

3.9.2 Parameter Selection for the UOV Signature

Direct Attack. In order to forge a single signature, an attacker would have to solve the related public key system of m quadratic equations in n variables over the field $\mathbb{F}_q$.

The most common approach to find a solution is to first guess v variables randomly. This yet preserves a solution with high probability. The best form to solve the remaining multivariate quadratic system with m equations and variables is to guess an appropriate number of additional variables (called *tradeoff*) and then use the F_5 algorithm, to compute the related Groebner basis (see the technique *Hybrid Approach* of Bettale et al. (BETTALE; FAUGÈRE; PERRET, 2009; BETTALE; FAUGÈRE; PERRET, 2012)). In short, first the *degree of regularity* d_{reg} is computed. For semi-regular sequences, which generic systems of equations are assumed to be, the degree of regularity is defined as the index of the first non-positive coefficient of the Hilbert series $S_{m,n}$:

$$S_{m,n} = \frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1 - z)^n}, \tag{3.5}$$

where d_i is the degree of the i-th equation and $S_{m,n}$ is a rational function in the variable z. The degree of regularity allows to get a precise knowledge of the complexity of Groebner basis computation for semi-regular systems of equations. In this case, d_{reg} decreases as m increases. Thus, the more the system is overdetermined, more quickly its Groebner basis can be computed. Then, the complexity of solving a zero-dimensional system (semi-regular) using the F_5 algorithm (BETTALE; FAUGÈRE; PERRET, 2009, Prop. 2.2) is

$$O\left(\left(m\binom{n + d_{reg} - 1}{d_{reg}}\right)^{\alpha}\right), \tag{3.6}$$

where α is called the linear algebra constant, $2 \leq \alpha \leq 3$. This work will adopt $\alpha = 2$ targeting an upperbound for the security.

Recently, Thomae et al. showed that it can be done even better than simply guessing v variables randomly (THOMAE; WOLF, 2012). Computing these v variables through the linear systems allows to solve a quadratic system of $m - \left\lfloor \frac{v}{m} \right\rfloor$ equations and the same number of variables afterward.

In (BETTALE; FAUGÈRE; PERRET, 2012), the authors show the best results for solving a generic non-linear system of equations, including $\mathcal{MQ}$ systems, over a finite field $\mathbb{F}_q$ with $q > 2$ and $log(q) \ll n$, obtaining asymptotic complexity $2^{n(3.31-3.62\lg q^{-1})}$ for the *Hybrid Approach* method and using, mainly, the F_5 algorithm.

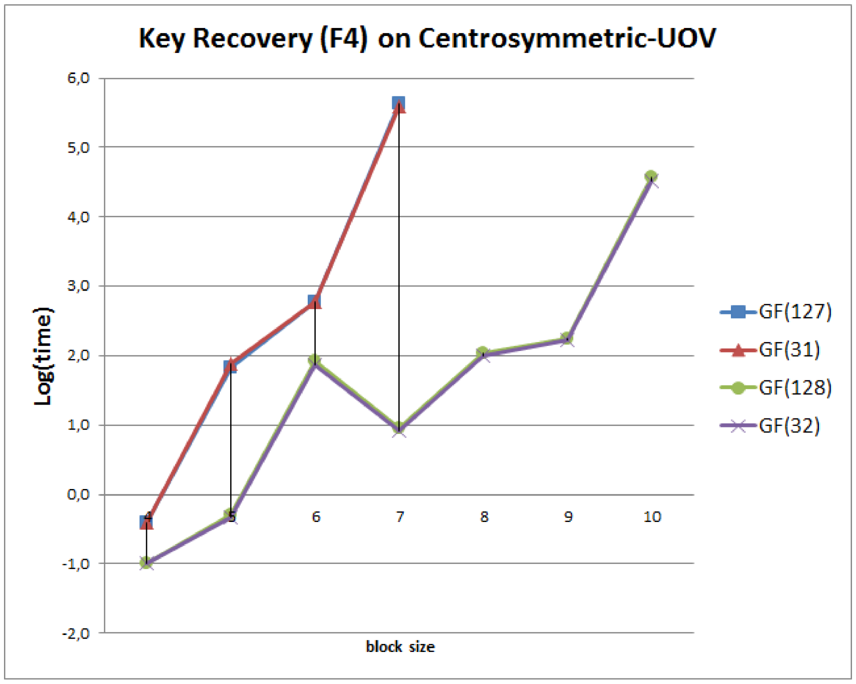
Key recovery attacks. There are two known key recovery attacks at the moment. The first one is a purely algebraic attack called the *reconciliation attack* (BERNSTEIN; BUCHMANN; DAHMEN, 2009, Sec. 5.5, pg 226). In order to obtain the private key S, solving $\binom{k+1}{2}m$ quadratic equations in kv variables for an optimal parameter $k \in \mathbb{N}$ is required. The second attack is a variant of the *Kipnis-Shamir* attack against the UOV scheme (KIPNIS; SHAMIR, 1998). Its general complexity is $O\left(q^{v-m-1}m^4\right)$. Notice that $v = 2m$ is very conservative in order to prevent this attack, thus, a smaller v can be chosen since that m is sufficiently large. For the case $k \geq 2$, even the reconciliation attack will not gain considerable advantage from choosing a smaller v, and the same occurs for the direct attack.

3.9.3 Practical Experiments

In addition to the above-mentioned theoretical results, we have conducted experiments using Magma with the two most relevant attacks against the Centrosymmetric variant of the original UOV (KIPNIS; PATARIN; GOUBIN, 1999) construction: the *(equivalent) key recovery* and the *signature forgery*. The idea of the key recovery attack is that the attacker tries to recover an equivalent private key for a given public key in order to sign messages on behalf of the legitimate owner of that public key. In practice, the attack consists of solving a system of equations constructed with the public key and the knowledge of the structure of the private key.

In the case of the signature forgery attack, the attacker simply tries to solve the public system for a desired hash of some message. The attacks are carried out by tailoring an equivalent system of quadratic equations, which can be solved by using effi-

Figure 13: Key recovery attack against our CentrosymmetricUOV construction

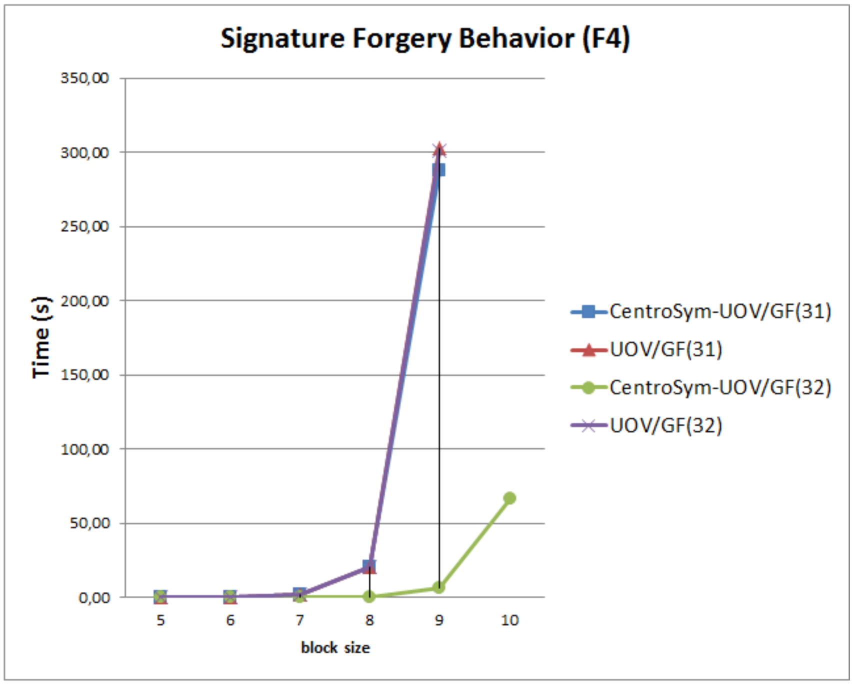


Source: Author

cient algorithms to this purpose (i.e. the so-called Groebner Bases algorithms (BOEGE; GEBAUER; KREDEL, 1986)). Both key recovery and signature forgery attacks in Magma against our CentrosymmetricUOV are analyzed.

In order to illustrate these results, we show in Figure 13 that both prime and binary fields provide an exponential behavior of the attack. Note that the binary case is experimentally a bit less secure, so we will give more emphasis on prime fields. In addition, Figure 14 compares the behavior of the signature forgery against plain UOV and Centrosymmetric UOV for prime and binary fields. The experiments also confirm that, even though the Centrosymmetric UOV over binary fields still displays an exponential behavior against such attacks, it remains less secure than prime field constructions. It also shows that the CentrosymmetricUOV behaves quite similarly to plain UOV for prime fields for one single layer. On the other hand, from our experiments, when more than one layer is introduced, the degree of regularity decreases, so this unexpected behavior casts doubt on the security of this construction.

Figure 14: Signature forgery attack against our Centrosymmetric UOV construction



Source: Author

One explanation is that there are too many dependent elements in a multi-layered structure that makes the attack sensible to this variation.

Table 1 illustrates a comparison of the new RingRainbow/RingUOV and other variants UOV/Rainbow already published. The parameters for RingRainbow/RingUOV refer to the security level $2^{102} - 2^{110}$, although the practical levels for some of them may be smaller due to the recent (*generalized Rainbow Band Separation attack*) with good keys and other variants (THOMAE, 2012a; THOMAE, 2012b). It can be noticed that both private and public keys are much smaller for RingRainbow/RingUOV than for previous proposals in the literature, but as it was argued the new constructions do not present satisfactory security.

3.10 Conclusions about $\mathcal{MQ}$ Signatures

This work on reducing MQ key sizes showed the following items:

Table 1: Parameter comparison of $\mathcal{MQ}$ signatures for a target security levels between $2^{102} - 2^{110}$ depending on the construction. The signing key *sk* and the public key *pk* are given in KiB. The hash size $|hash|$ and the signature sizes $|sig|$ are given in bits.

construction	$\lvert sk\rvert$	$\lvert pk\rvert$	$\lvert hash\rvert$	$\lvert sig\rvert$	ref.
Rainbow($\mathbb{F}_{2^4}$, 30, 29, 29)	75.8	113.4	232	352	(PETZOLDT; BULYGIN; BUCHMANN, 2010b)
Rainbow($\mathbb{F}_{31}$, 25, 24, 24)	59.0	77.7	232	392	(PETZOLDT; BULYGIN; BUCHMANN, 2010b)
CyclicUOV($\mathbb{F}_{2^8}$, 26, 52)	14.5	76.1	208	624	(PETZOLDT; BULYGIN; BUCHMANN, 2010a)
NC-Rainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	25.5	66.7	384	672	(YASUDA; SAKURAI; TAKAGI, 2012)
Rainbow($\mathbb{F}_{2^8}$, 29, 20, 20)	42.0	58.2	272	456	(PETZOLDT; BULYGIN; BUCHMANN, 2010b)
CyclicLRS($\mathbb{F}_{2^8}$, 26, 52)	71.3	13.6	208	624	(PETZOLDT; BULYGIN; BUCHMANN, 2011)
UOVLRS($\mathbb{F}_{2^8}$, 26, 52, 26)	71.3	11.0	208	624	(PETZOLDT; BULYGIN; BUCHMANN, 2011)
CyclicRainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	19.1	10.2	208	344	(PETZOLDT; BULYGIN; BUCHMANN, 2010a)
RingRainbow($\mathbb{F}_{31}$, 135, 15 ,30)	1.2	2.1	225	900	this work (circulant ring)
RingUOV($\mathbb{F}_{31}$, 246, 41)	0.9	0.7	205	1435	this work (circulant ring)
RingUOV($\mathbb{F}_{31}$, 246, 41)	4.4	4.7	204	610	this work (centrosymmetric ring)

Source: Author

- a new general method, called RingUOV/RingRainbow, for the construction of the key pairs for the UOV/Rainbow signatures is proposed, which is characterized by partitioning the matrices representing the keys into more compact blocks belonging to a certain matrix ring.

- In contrast to the literature, where only the public key or only the private key are reduced, RingUOV/RingRainbow reduces *simultaneously* both the private and pubic key sizes.

- A general construction was designed, adopting a secure ring that allows to reduce the keys by a factor α, or a factor $\approx \alpha^{\ell}$, using recursively the ring structure in ℓ layers.

- A totally insecure ring class was found, i.e. the ring produced by simultaneously diagonalizable matrices, with respect to the reconciliation attack.

- A less compact ring class of non-simultaneously diagonalizable matrices (in particular, the centrosymmetric matrices) was found, where the reconciliation attacks were shown not to be exponential for the one-layer construction but when more layers are adopted the attack performs better, and the resulting security does not cast confidence.

The above mentioned results do not give a good perspective on achieving the goals of this thesis regarding the search for good candidates to be deployed in very constrained platforms. We consider this a negative research result. Fortunately, we will give better results toward this direction from the realm of hash-based signatures.

4 HASH-BASED DIGITAL SIGNATURES

Internet of Things (IoT) purports to connect a vast range of equipments via the Internet, as long as the underlying device is connected directly or indirectly to anyone supporting IP protocol. This includes extremely constrained platforms, with as little as 64 KiB ROM and 4 KiB RAM as certain SIM cards. While this is commonly enough for symmetric primitives (hash functions, block and stream ciphers, and even richer constructions like authenticated encryption with associated data), it may push most asymmetric primitives beyond the available computational resources on such platforms. Yet, securing a typical Internet of Things scenario requires, at the very least, a basic public-key infrastructure (PKI) able to support public-key encryption and digital signatures, which are themselves based on asymmetric primitives.

While encryption can be attained with fairly modest resources by adopting lattice-based schemes (HOFFSTEIN; PIPHER; SILVERMAN, 1998) or code-based schemes (MISOCZKI et al., 2013), offering even the most basic functionality of digital signatures on the most stringent platforms is no easy task. Obvious and more exotic candidates alike suffer from the extreme lack of computational resources on some of those platforms, which currently seem to be at, or already beyond, the bare minimum needed to establish a full-fledged PKI. This lack of an efficient signature functionality constitutes a serious hindrance to the very concept of the IoT, especially if resorting to more expensive processors or co-processors is not an option.

Hash-based signatures, which originally appeared somewhat too far-fetched for actual deployment, turned out to be a very promising tool for the aforementioned scenario. On the one hand, their main drawback – which was a very long key generation time, have been for the most part successfully addressed in recent research works (BUCHMANN et al., 2007). On the other hand, practical considerations like the actual signature size and consequent bandwidth occupation, as well as leakage-resilience, have also been addressed, with very promising results (BUCHMANN; DAHMEN; HÜLSING, 2011; EISENBARTH; MAURICH; YE, 2013; HÜLSING, 2013b; ROHDE et al., 2008). Although they do constitute true digital signatures in the sense of public-key cryptosystems, such schemes are based on entirely symmetric primitives, which are readily available on constrained platforms, are typically very efficient, and appear to resist attacks mounted even with the help of quantum computers, to the extent that hash-based signatures have been promoted to the category of quantum-resistant, or post-quantum, cryptosystems. Yet, given the extreme scarcity of resources one finds in IoT processors, full establishment of a secure environment for realistic applications requires that all cryptographic features be made as lightweight as possible, and hash-based signatures are no exception. It therefore makes sense to look for the most efficient constructions rather than sticking with proofs of concept.

Also in a quantum scenario, quantum computers do not harm the security of hash-based signature – the best known quantum attack is Grover's algorithm – allowing exhaustive search in a set with n elements in time square root of n using $\lg n$ space (GROVER, 1996; GROVER, 1997). While other signature schemes rely on additional intractability assumptions to generate a signature, a hash-based solution only needs a secure hash function for the same procedure. Therefore the security of the scheme holds with the hash function used. If the latter one gets compromised, it can be replaced. The underlying scheme is not affected.

Some hash-based schemes even reduce the need for a collision-resistant hash function to one that only needs to withstand attacks on the second-preimage (DAHMEN et al., 2008). As an example practical attacks against the collision-resistance of the MD5 function are known, but one still does not know about relevant attacks on the second-preimage property.

4.1 Preliminaries

Universal One-Way Hash Functions (UOWHF, "woof"). A universal hashing (in a randomized algorithm) refers to selecting a hash function at random from a family of hash functions with the following property:

Target Collision-Resistance (TCR). Given an element x in the domain (chosen by the adversary), when a member h is chosen at random from a hash family H, it is infeasible to find a different domain element x' which collides with x, i.e. $h(x) = h(x')$. The security notion of TCR is stronger than second-preimage resistance, but weaker than collision resistance.

Merkle Trees or Hash Trees (Wikipedia, 2015). A hash tree or Merkle tree is a tree in which every non-leaf node is labeled with the hash of the labels of its children nodes. Hash trees are useful because they allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains. Demonstrating that a leaf node is a part of the given hash tree requires processing an amount of data proportional to the logarithm of the number of nodes of the tree; this contrasts with hash lists, where the amount is proportional to the number of nodes. The concept of hash trees is named after Ralph Merkle who patented it in 1979 and introduced them in (MERKLE, 1979).

4.2 One-time Signature Schemes

A one-time signature scheme is characterized by the fact that a key pair can only be used once to sign one single message. One-time signature schemes first appeared in the works of Diffie and Lamport (DIFFIE; HELLMAN, 1976; LAMPORT, 1979) and Rabin (DEMILLO et al., 1978, Chapter "Digitalized signatures").

Merkle (1979) proposed a technique to transform a one-time signature scheme into a multi-time signature scheme able to sign an arbitrary but fixed number of messages. We now review some of the most relevant one-time and multi-time hash-based signature schemes in the literature.

4.2.1 Lamport-Diffie One-time Signature Scheme

The Lamport-Diffie one-time signature scheme *(LD-OTS)* is as old as public-key cryptography (it was proposed within its founding seminal paper (DIFFIE; HELLMAN, 1976)). A more complete description of the scheme was given in (LAMPORT, 1979). Since its original proposal, *LD-OTS* has not suffered any relevant attacks, making its security very reassuring. An important advantage of *LD-OTS* is that its security assumption is only the existence of a one-way function. As main drawbacks we mention the large private and public key sizes and the necessity of authenticating every public key for each message to be signed. Fortunately, there are many recent improvements addressing these issues.

We first give the intuition behind *LD-OTS* with a toy example for signing a single bit message. Let b be the bit to be signed, and f be a length-preserving one-way function. Firstly, the signer simply generates two private strings of n bits, x_0 and x_1, and computes the one-way function on them $y_0 = f(x_0)$ and $y_1 = f(x_1)$. The private key is composed of the private strings $\langle x_0, x_1 \rangle$ and the public key by their evaluation by f, $\langle y_0, y_1 \rangle$. Then, the public key is (authentically) delivered to a verifier.

The signature generation is quite simple, if the bit b to be signed is zero, the signer reveals x_0 to the verifier, otherwise x_1 is revealed. Under the reception of x_b, the verifier computes $f(x_b)$ and checks if it is equal to the respective y_b. It is clearly difficult to forge a signature in this construction since it involves to invert the one-way function f.

Notice that for signing a fixed-size n-bit message, a storage of $2n^2$ bits is required for each private and public keys. For example, to achieve a security of at least $\approx 2^{128}$, the one-way function must have at least 128 bits. Therefore, the private and the public keys must have at least $2 * 128^2 = 32768$ bits (4 KiB).

Next, we give a more complete and formal description of *(LD-OTS)*. Let n be a positive integer, the security parameter of *LD-OTS*. *LD-OTS* uses a one-way function

$$f : \{0,1\}^n \to \{0,1\}^n,$$

and a cryptographic one-way hash function (no need for collision-resistance here)

$$G : \{0,1\}^* \to \{0,1\}^n.$$

LD-OTS key pair generation: The signature key x consists of $2n$ bit strings of length n chosen uniformly at random:

$$x = (x_0[0], x_0[1], \ldots, x_{n-1}[0], x_{n-1}[1]) \quad \in_R \quad \{0,1\}^{(n,2n)}.$$

The verification key y is

$$y = (y_0[0], y_0[1], \ldots, y_{n-1}[0], y_{n-1}[1]) \quad \in \quad \{0,1\}^{(n,2n)},$$

where $y_i[b] = f(x_i[b])$, $0 \le i \le n-1$, $b = 0, 1$.

LD-OTS signature generation: The digest d of a message M is signed using the signature key x. Let $d = G(M) = (d_0, \ldots, d_{n-1})$. The signature of d is

$$\sigma = (\sigma_0, \ldots, \sigma_{n-1}) = (x_0[d_0], \ldots, x_{n-1}[d_{n-1}]) \quad \in \quad \{0,1\}^{(n,n)}.$$

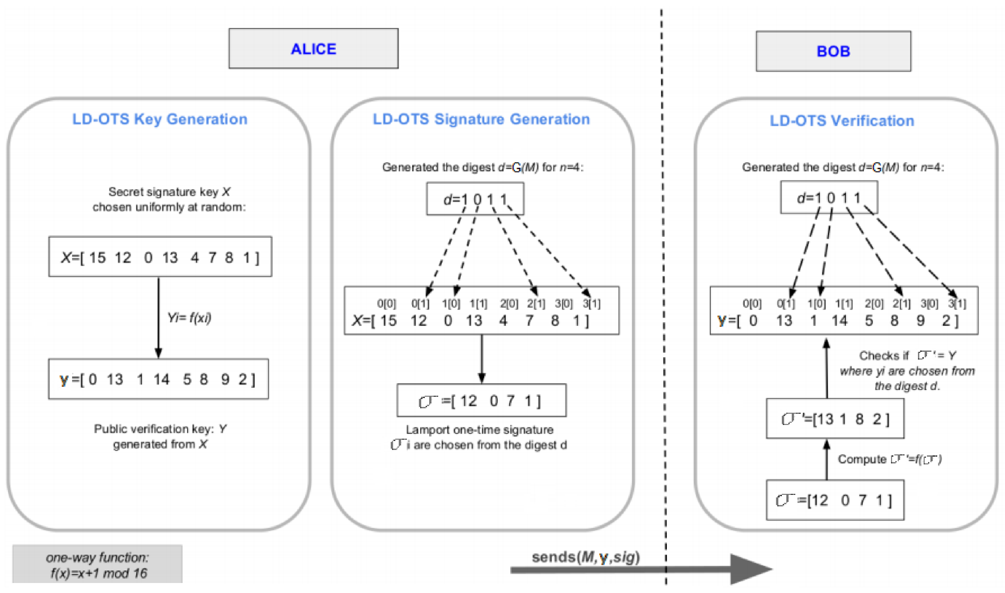
The signature σ is a sequence of n bits strings, each one of length n, and the signature elements are chosen from the key x according to the bits of the message digest. Then, for each bit d_i of d, it selects the corresponding string of length n from the private key x, that is, the algorithm selects $x_i[d_i]$. To verify a signature, the verifier first computes the message digest $d = G(M) = (d_0, \ldots, d_{n-1})$ and selects $y_i[d_i]$ from y, the verification (public) key. Then, it checks whether

$$\sigma = (f(\sigma_0), \ldots, f(\sigma_{n-1})) = (y_0[d_0], \ldots, y_{n-1}[d_{n-1}]).$$

If this holds, the signature is accepted as valid; otherwise it is rejected. Observe that signature verification requires n evaluations of G while signing requires no evaluation of G.

Figure 15 illustrates the Lamport scheme. In the example, let's assume that the function $f(x) = x + 1 \bmod 16$ is a one-way function. For the public key generation, $2n$ evaluations of the one-way function are required, one for each element of x. For signature verification, n evaluations of the one-way function are required, one for each element of σ.

Figure 15: Lamport-Diffie OTS scheme example.


ALICE
BOB
LD-OTS Key Generation
Secret signature key X chosen uniformly at random:
X=[15 12 0 13 4 7 8 1]
Yi= f(xi)
y=[0 13 1 14 5 8 9 2]
Public verification key: Y generated from X
LD-OTS Signature Generation
Generated the digest d=G(M) for n=4:
d=1 0 1 1
0[0] 0[1] 1[0] 1[1] 2[0] 2[1] 3[0] 3[1]
X=[15 12 0 13 4 7 8 1]
σ=[12 0 7 1]
Lamport one-time signature
σi are chosen from the digest d
LD-OTS Verification
Generated the digest d=G(M) for n=4:
d=1 0 1 1
0[0] 0[1] 1[0] 1[1] 2[0] 2[1] 3[0] 3[1]
Y=[0 13 1 14 5 8 9 2]
Checks if σ'= Y where yi are chosen from the digest d.
σ'=[13 1 8 2]
Compute σ'=f(σ)
σ=[12 0 7 1]
one-way function: f(x)=x+1 mod 16
sends(M,y,sig)

Source: (BARRETO et al., 2014)

4.2.2 Winternitz One-time Signature Scheme

Winternitz proposed an improvement to Lamport's one-time signature scheme, reducing the size of the signatures, additionally to the public and private key sizes. The basic idea of the Winternitz OTS is to sign several bits at once.

The Winternitz one-time signature scheme (*W-OTS*) was invented just after Merkle presented his thesis in 1979 but was first publicly described in (MERKLE, 1988). Its formal security proof in the standard model was given later and is based on a secure collision-resistant hash function (or a universal one-way hash function). The security proof is a consequence of the work of Hevia and Micciancio (HEVIA; MICCIANCIO, 2002) which is more general and proves the security for many one-time signatures using graph-based equivalences. The dedicated security proof for *W-OTS* was first given in (DODS; SMART; STAM, 2005).

W-OTS uses a one-way function

$$f : \{0,1\}^n \to \{0,1\}^n$$

and a cryptographic hash function

$$G : \{0,1\}^* \to \{0,1\}^m,$$

where n, m are positive integers. W-OTS uses a tradeoff parameter w which is the *number of bits to be signed simultaneously*. Larger values for w result in smaller signature keys and longer times for signing and verification. A comparative analysis of the run times and key sizes in terms of parameter w for the original *W-OTS* is found in (DODS; SMART; STAM, 2005).

W-OTS key pair generation: Given parameter $w \in \mathbb{N}$, the private key is

$$X = (x_0, \ldots, x_{L-1}) \in_{\mathcal{R}} \{0,1\}^{(n,L)},$$

where the x_i are chosen uniformly at random. The size L is computed as $L = \ell_1 + \ell_2$, where

$$\ell_1 = \left\lceil \frac{m}{w} \right\rceil, \qquad \ell_2 = \left\lceil \frac{\lfloor log_2 \ell_1 \rfloor + 1 + w}{w} \right\rceil.$$

The verification key

$$Y = (y_0, \ldots, y_{L-1}) \in \{0,1\}^{(n,L)}$$

is computed by applying the one-way function f to each element of the signature key $2^w - 1$ times:

$$y_i = f^{2^w-1}(x_i), \qquad for \quad i = 0, \ldots, L-1.$$

In order to minimize storage requirements, in the case of implementation for constrained devices, the use of a pseudorandom number generator (PRNG) described in (BUCHMANN et al., 2006; NIST, 2012) is suggested. This PRNG enables the recovery of all signature keys based only on the initial seed SEED_0, $\text{SEED}_{in} \rightarrow (\text{RAND}, \text{SEED}_{out})$.

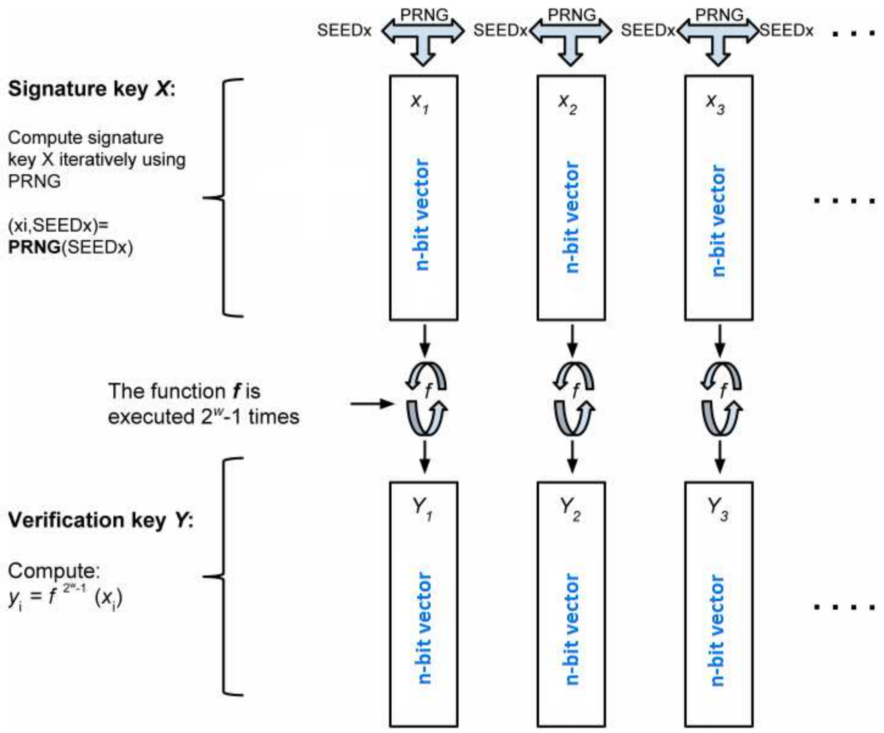
Figure 16 shows an example of a key pair generation in the Winternitz signature scheme, using a PRNG and a one-way function. The PRNG computes $PRNG(\text{SEED}_x) \rightarrow (x_i, \text{SEED}_x)$. This scheme produces smaller signature keys than Lamport's, but it increases the number of one-way function evaluations from 1 to $2^w - 1$ for each element of the key signature.

W-OTS signature generation: To generate a signature, first compute the message digest $d = G(M) = (d_0, \ldots, d_{m-1})$. If necessary, add zeros to the left of d, so as to make the bitlenght of d divisible by w. Then, d is split into ℓ_1 binary blocks of size w, resulting in $d = (d_0 \| \ldots \| d_{\ell_1 - 1})$, where $\|$ denotes concatenation. The d_i blocks are represented as integers in base-w, thus belonging to $\{0, 1, \ldots, 2^w - 1\}$. Now, a checksum c is computed as

$$c = \sum_{i=0}^{\ell_1 - 1} (2^w - 1 - m_i).$$

Since $c \leq \ell_1 2^w$, the length of the binary representation of c is less than $\lfloor \log_2 \ell_1 2^w \rfloor + 1 = \lfloor \log_2 \ell_1 \rfloor + w + 1$. If necessary, add zeros to the left of c in order to make the bitlength

Figure 16: Winternitz key pair generation using a PRNG


PRNG
SEEDx
Signature key X:
Compute signature key X iteratively using PRNG
(xi,SEEDx)= PRNG(SEEDx)
x1
x2
x3
n-bit vector
The function f is executed 2^w-1 times
f
Verification key Y:
Compute: y_i = f^{2^w-1}(x_i)
Y1
Y2
Y3

Source: Adapted from (BARRETO et al., 2014)

of string c divisible by w. Then, the extended string c can be divided into ℓ_2 blocks $c = (c_0 \| \dots \| c_{\ell_2 - 1})$ of length w. Let $b = d \| c$ be the concatenation of the extended string d with the extended string c. Thus $b = (b_0 \| b_1 \| \dots \| b_{L-1}) = (d_0 \| \dots \| d_{\ell_1 - 1} \| c_0 \| \dots \| c_{\ell_2 - 1})$. The signature is computed as:

$$\sigma = (\sigma_0, \dots, \sigma_{L-1}) = (f^{2^w - 1 - b_i}(x_0), f^{b_1}(x_1), \dots, f^{2^w - 1 - b_{L-1}}(x_{L-1})).$$

W-OTS signature verification: To verify signature $\sigma = (\sigma_0, \dots, \sigma_{L-1})$ of message M, first calculate $(b_0, \dots, b_{L-1})$ in the same way as in the signature generation; then, compute

$$\sigma_i' = f^{b_i}(\sigma_i), \qquad for \quad i = 0, \dots, L-1.$$

Finally, check whether

$$\sigma' = (\sigma_0', \dots, \sigma_{L-1}') = Y = (y_0, \dots, y_{L-1}).$$

If the signature is valid, then $\sigma_i = f^{b_i}(x_i)$ and, therefore.

$$f^{b_i}(\sigma_i) = f^{2^w-1}(x_i) = y_i$$

holds for $i = 0, 1, \ldots, L - 1$.

4.3 Merkle Signature Scheme

Merkle developed a meta-construction enabling for a multi-time signature scheme that uses one-time signatures and a binary tree (MERKLE, 1979). We call Merkle's construction the Merkle Signature Scheme (*MSS*). *MSS* uses a binary tree whose leaves are computed from one-time signature key pairs. The internal nodes of the tree are computed by simply concatenating their two children and hashing the result. The general public key will be the root of the tree. For instance, given a tree of height h, the public key consists of one single tree node (the root) that authenticates all the 2^h *OTS* key pairs (the leaves).

Merkle did not provide a security proof in his original work. In 2005, Coronado gave *MSS*'s first formal security proof in the standard model and it is based on the existence of collision resistant-hash functions (CORONADO, 2005). In 2008, Dahmen *et. al.* provided a slightly modified version of *MSS* with a security proof that requires the weaker assumption of second-preimage resistance from the underlying hash function (DAHMEN et al., 2008).

Most of the new recent MSS variants come with a security proof in the standard model and outperform RSA in many settings regarding run times. The main drawback of *MSS* is the signature size which to a large extent depends on the underlying one-time signature scheme used.

Merkle key generation. To generate the Merkle public key *(pub)*, which is the root of the Merkle tree, one must to first generate 2^h one-time key pairs in order to obtain all the leaves.

So a one-time signature algorithm generates the private keys $X[u]$ and the public keys $Y[u]$, for each leaf, enumerated as $u = 0, \ldots, 2^h - 1$. Algorithm 4.3.1 describes the procedure of a one-time key pair generation.

Algorithm 4.3.1 Winternitz one-time key pair generation (*Leafcalc*)

```
Require: Winternitz parameter $t$ and $w$; seed $\mathrm{SEED}_x$.
Ensure: verification key $Y$;
  for ($i = 0$, $i < L$, i++) do
      $(x[i], \mathrm{SEED}_x) = PRNG(\mathrm{SEED}_x)$;
      $\mathrm{y[i]} = f^{2^w-1}(x[i])$;
  end for
  $\mathrm{Y}=G(y[0]\|\ldots\|y[L-1])$;
  return Y;
```

Merkle public key generation (pub): Algorithm 4.3.2 generates the Merkle tree public key *pub*. The input values are: the initial leaf *leafIni* and tree height h. Each leaf node $node[u]$ of the tree receives the corresponding verification key $Y[u]$. The inner nodes of the Merkle tree $node[parent]$ contain the hash value of the concatenation of their left and right children, $node[left]$ and $node[right]$, respectively. Each time a leaf u is calculated and stacked in *stackNode*, the algorithm checks if the nodes at the top of the *stackNode* have the same height. If the nodes have the same height, the two nodes will be unstacked and the hash value of their concatenation will be pushed into *stackNode*. The algorithm terminates when the root of the tree is calculated.

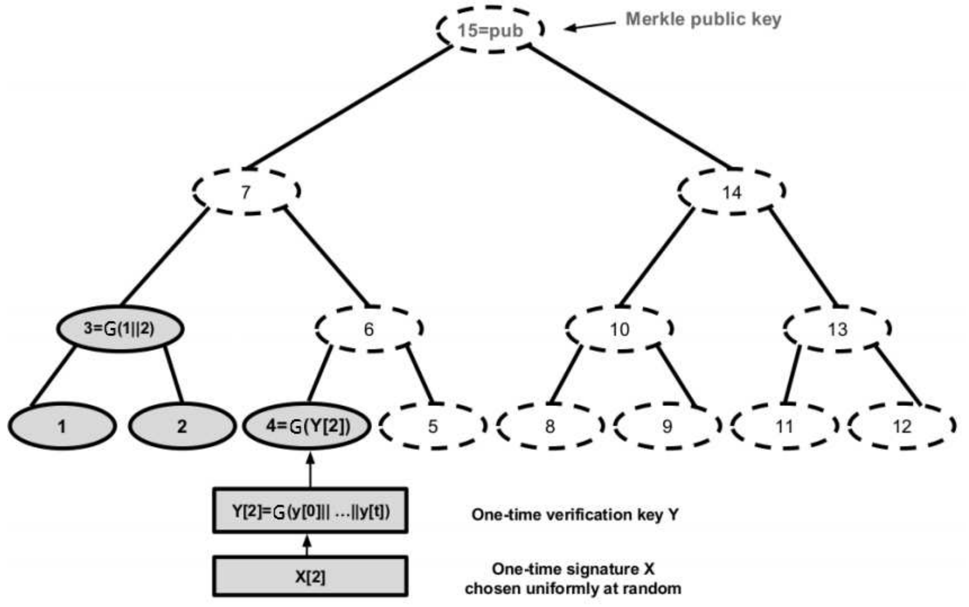
Figure 17 shows the order in which the nodes are stacked on the tree according to Algorithm 4.3.2. The nodes in gray represent the ones that have already been generated. For example, the 4-th node generated (leaf $u = 2$) received $Y[2]$. The 3-rd node is the hash result of the concatenation of the nodes 1 and 2.

MSS signature generation. Scheme *MSS* allows the generation of 2^h signatures for a tree of height h. Suppose we want to sign u messages, for $u = 0, .., 2^h$. Each message $M[u]$ is signed with the one-time signature key $X[u]$ resulting in a signature

Algorithm 4.3.2 Merkle public key generation (CalcRoot)

```
Require: Leaf $leafIni$; tree height $h$; seed $\mathrm{SEED}_{in}$.
Ensure: The root of the tree $pub$.
  Create a stack $stackNode$.
  $\mathrm{SEED}_0 = \mathrm{SEED}_{in}$
  for ($u = leafIni$, $u < 2^h$, u++) do
      $(\mathrm{SEED}_x, \mathrm{SEED}[u+1]) = PRNG(\mathrm{SEED}[u])$;
      $node[u].digest = G(\mathrm{Leafcalc}(L, \mathrm{SEED}_x))$
      Push $node[u]$ in the stack $stackNode$
      while The nodes at the top of the $stackNode$ has the same height do
          Pop $node[right]$
          Pop $node[left]$
          Compute $node[parent].digest = G(node[left].digest \| node[right].digest)$
          if $node[parent].height{=}h$ then
              return ($node[parent]$)
          else
              Push $node[parent]$ into $stackNode$
          end if
      end while
  end for
```

Figure 17: Merkle public key generation (pub)



Source: Adapted from (BARRETO et al., 2014)

$\sigma[u]$.

The authentication path *Aut* is used to store the nodes in the path needed to authenticate leaf $Y[u]$, eliminating the need for sending the whole Merkle tree to the receiver.

The Merkle signature Σ consists of one-time signature $\sigma[u]$ for the leaf u, the corresponding verification key $Y[u]$, the index u (leaf index) and its authentication path, $Aut = (Aut[0], .., Aut[h-1])$. Therefore, the signature is

$$\Sigma = (u, \sigma[u], Y[u], (Aut[0], \ldots, Aut[h-1])).$$

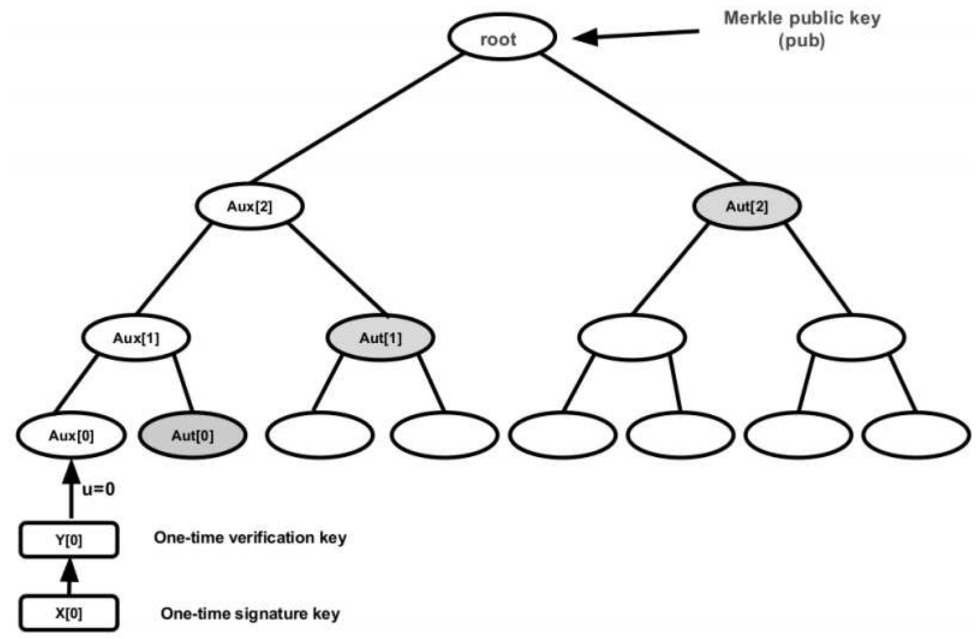
Notice that the one-time public key $Y[u]$ may not be transmitted, since it will be recomputed from the one-time signature $\sigma[u]$ and if the signature is legitimate, $Y[u]$ will be correctly authenticated in the Merkle tree.

The classic authentication path algorithm. The classic authentication path algorithm *(Path Regeneration Algorithm)* (MERKLE, 1979) computes node authentication path *Aut* for each tree leaf, needed to authenticate public key *pub* of the Merkle tree. This algorithm uses two stack variables, *Aut* and *Aux*. Stack *Aut* contains the current authentication path and stack *Aux* saves the next authentication nodes which will be needed. *Aut* is formed by the proper siblings at each level of the authentication path which connects the leaf to the root of the Merkle tree.

We now describe the computation of the authentication paths. The first authentication path is generated during the execution of Algorithm 4.3.2. The next authentication path is generated if a new signature is required. In Figure 18, the nodes in gray show the first authentication path *Aut* for the leaf $u = 0$.

Output and update phases: Algorithm 4.3.3 shows the steps for producing the authentication path for the next leaf u in the tree. The algorithm starts by signing the leaf $u = 0$; then, the leaf is updated in one unit, and the next authentication path is computed efficiently since only the nodes that change in the path will be updated.

Figure 18: First authentication path state in Algorithm 4.3.2



Source: Adapted from (BARRETO et al., 2014)

Algorithm 4.3.3 The Path Regeneration Algorithm (MERKLE, 1979)

```
Require: Tree height $h$; seed $\mathrm{SEED}_{in}$.
Ensure: One authentication path $Aut$.
  for ($u = 0$, $u < 2^h$, $u$++) do
      for ($j = 0$, $j < h$, $j$++) do return $Aut$ of leaf $u$
          A signature with leaf $u$ is done
          for ($j = 0$, $j < h$, $j$++) do
              if $(u + 1)/(2^j) = 0$ then
                  Update $Aut[j] = Aux[j]$
                  $node_{Ini} = (u + 1 + 2^j) \oplus 2^j$.
                  $Aux[j] = \mathrm{CalcRoot}(node_{Ini}, j, \mathrm{SEED}_{in})$.
              end if
          end for
      end for
  end for
```

Algorithm 4.3.3 updates authentication nodes by executing function

$$CalcRoot(node_{Ini}, h, \mathrm{SEED}_{in}).$$

The function *CalcRoot* executes Algorithm 4.3.2 for node $node_{Ini}$. After 2^h rounds, the value of the selected node will be computed.

MSS signature verification. The signature verification consists of two steps (BERNSTEIN; BUCHMANN; DAHMEN, 2009, Chapter "Hash-based Digital Signature Schemes"): in the first step, signature σ is verified by using the one-time verification key $Y[i]$ and the underlying one-time signature; in the second step, the public key of the Merkle tree is reconstructed by the verifier which uses the authentication path $(p[0], \ldots, p[h])$ and leaf i. Index i is used to decide the order in which the authentication path is reconstructed. Initially, for leaf i, $p[0] = G(Y[i])$. For each $j = 1, 2, \ldots, h$; the node $p[h]$ is computed using the condition $(\lfloor i/(2^{j-1}) \rfloor \equiv 1 \bmod 2)$ and the recursive formula

$$p[j] = \begin{cases} G(Aut[j-1] \,||\, p[j-1]) & \text{if } \lfloor i/(2^{j-1}) \rfloor \equiv 1 \bmod 2; \\ G(p[j-1] \,||\, Aut[j-1]) & \text{otherwise.} \end{cases}$$

Finally, if the value $p[h]$ is equal to the public key *pub*, the signature Σ is valid.

One possible drawback of *MSS* is being stateful, which means that one cannot have a signing key shared across multiple devices and the scheme may be vulnerable to "restart attacks" where the scheme is forced to re-use a secret key, compromising the security. This may be a problem in applications where the same key should be shared between several devices or threads, for example TLS using load balancing. On the other hand, MSS is highly suitable in the context of constrained embedded systems where each device takes care of its own private key.

4.4 CMSS – An Improved Merkle Signature Scheme

The *CMSS* scheme is a variant of the *MSS* scheme with reduced private key size, key pair generation time, and signature generation time (BUCHMANN et al., 2006). *CMSS* is able to sign up to 2^{40} documents with a reasonable time for all the operations key generation, signature generation and signature verification when compared to RSA and ECDSA.

In the *CMSS* scheme, two *MSS* authentication trees are used, a subtree and a main tree, each one with 2^j leaves, where $j = h/2$. Thus, the number of possible signatures can be increased in relation to *MSS*. Note that *MSS* becomes impractical for $h > 25$ since private keys are too large and the key pair generation takes too much time. For example, to generate 2^{20} signatures, two trees with 2^{10} leaves each are generated with *CMSS*, while with *MSS*, a single tree with 2^{20} leaves is constructed. Therefore, key generation time is reduced.

In order to improve signature generation time, *CMSS* uses Szydlo's treehash algorithm (SZYDLO, 2004), which is more efficient for constructing authentication paths which requires logarithmic space and time. This algorithm creates the paths in a sequential manner, i.e. the path for leaf $i + 1$ is created after the path for leaf i. Since to verify the authentication path one also needs to know the value of i, a one-time signature implemented using Szydlo's algorithm will reveal the time ordering of different issued signatures. In some applications this may be an issue. Szydlo's algorithm was improved in (BUCHMANN; DAHMEN; SCHNEIDER, 2008), whose purpose is to balance the number of calculated leaves in each authentication path update.

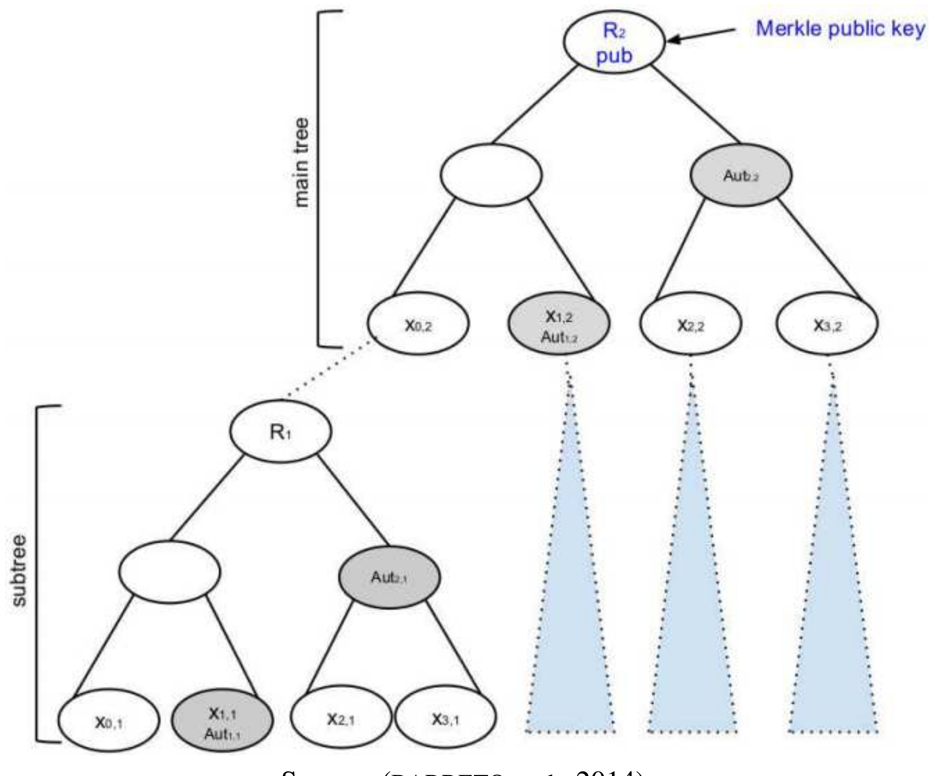
To reduce private key sizes, a pseudorandom number generator *PRNG* (NIST, 2012) is used, and only the seed of the *PRNG* is stored. By using a hash function of n bits and the Winternitz parameter L, the signature key will have $(L \cdot n)$ bits. Thus, one needs only to store a seed of n bits.

The *CMSS* public key is the root of the main tree. The messages are signed using the leaves of the subtree. After the first 2^j signatures have been generated, a new subtree is constructed and used to generate the next 2^j signatures.

CMSS Key generation: For key pair generation, the *MSS* key pair generation is called twice. The subtree and its first authentication path are generated. Then, the main tree and its first authentication path are computed.

The *CMSS* public key is the root of the main tree. *CMSS* uses Winternitz as the underlying one-time signature scheme. Figure 19 shows the *CMSS* scheme.

Figure 19: CMSS signature scheme



Source: (BARRETO et al., 2014)

CMSS signature generation: *CMSS* signature generation is carried out in various parts. First, the one-time signature of the message is computed using the leaf of a subtree. After that, the one-time signature of the subtree's root is computed using the leaf of the main tree. This signature will be recalculated in the next signature only if all the leaves of the current subtree have already been used. Then, the authentication path of both trees (main and subtree) is appended in the signature and the next authentication paths are computed. Thus, the next subtree is partially constructed, and the *CMSS* private key is updated.

CMSS verification: To verify the *CMSS* signature, checking the roots of both subtrees and both one-time signatures is required.

In (BUCHMANN et al., 2006) *CMSS* was shown to be competitive in practice, presenting a highly efficient implementation within the Java Cryptographic Service Provider FlexiProvider and showing that the implementation can be used to sign messages within Microsoft Outlook.

4.5 GMSS – Signatures with Virtually Unlimited Signature Capacity

The *GMSS* scheme was published in 2007 (BUCHMANN et al., 2007). It is another variant of the Merkle signature scheme, which allows for a virtually unlimited number of 2^{80} messages to be signed with one key pair. The basic construction of *GMSS* consists of a tree with T layers (subtrees). Subtrees in different layers may have different heights. To reduce the cost of signature time, *GMSS* distributes the cost of one signature generation across previous signatures and key generation. Thus, this scheme allows for the choice of different parameters w of Winternitz in different subtrees, in order to produce smaller signatures.

GMSS Key generation: For each subtree, the one-time key generation algorithm computes the signature keys and Algorithm 4.3.2 calculates the roots. The first authentication path of each subtree is stored during generation of the root. Then, the signatures Σ_τ of Merkle subtrees, for $\tau = 2, \ldots, T$ will be computed to be used in the first signature. Since those signature values change less frequently for the upper layers, the precomputation can be distributed over many stages, resulting in a significant improvement of the signing speed. To ensure a small size for private keys, only the seed for the *PRNG* needs to be stored.

GMSS signature generation: The root of a subtree is signed with the one-time signature key corresponding to the parent tree. $Root_\tau$ denotes the root of the tree τ. σ_τ denotes the one-time signature of $Root_\tau$, which is generated using the leaf l of parent τ. The message digest d is signed using the leaves on the deepest layer T.

The number of messages that can be signed with a *GMSS* key is $S = 2^{h_1+\ldots+h_T}$, where $h_1, \ldots h_T$ are the heights of the subtrees. The *GMSS* signature consists of:

- the leaf index u;
- the one-time signatures σ_d and $\sigma_{\tau_{i,j_i}}$ for $i = 2, \ldots, T$, $j = 0, \ldots, 2^{h_1+\ldots+h_{i-1}} - 1$.
- authentication paths $Aut[\tau_{i,j_i}, l_i]$ of leaves l_i, for $i = 1, \ldots, T$, $j = 0, \ldots, 2^{h_1+\ldots+h_{i-1}} - 1$.

During the signature generation roots $Root_{\tau_{i,1}}$ are also calculated, as are the authentication paths $Aut[\tau_{i,1}, 0]$ of trees $\tau_{i,1}$, for $i = 2, \ldots, T$. The signature generation is split into two parts. The first, online part, computes σ_d. The second, offline part, precomputes the authentication paths and one-time signatures of the roots required for the upcoming signatures.

GMSS verification: The GMSS signature verification is essentially the same as that of schemes *MSS* and *CMSS*: the verifier checks the one-time signatures σ_d and $\sigma_{\tau_{i,j_i}}$ for $i = 2, \ldots, T$ and $j = 0, \ldots, 2^{h_1+\ldots+h_{i-1}} - 1$. Therefore, she verifies the roots $Root_\tau$ for $\tau = 2, \ldots, T$, and the public key using the corresponding authentication path.

4.6 XMSS – eXtended Merkle Signature Scheme

The hash-based signature scheme XMSS is a variant of MSS and being the first practical forward secure signature with minimal security requirements (BUCHMANN; DAHMEN; HÜLSING, 2011). This scheme uses a function family F and a hash function family G. *XMSS* is efficient, provided that G and F are efficient. The parameters of *XMSS* are: $n \in \mathbb{N}$, the security parameter; $w \in \mathbb{N}(w > 1)$, the Winternitz parameter; $m \in \mathbb{N}$, the message length; $h \in \mathbb{N}$, the tree height; the one-time signature keys $x \in \{0, 1\}^n$, chosen randomly with uniform distribution; a one-way function family

$$F_n = \{f_K : \{0, 1\}^n \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n\},$$

and a hash function g_K, chosen randomly with uniform distribution from the family

$$G_n = \{g_K : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n.\}$$

The one-time signature key x is used to construct the one-time verification y, by applying the function family F_n. In (BUCHMANN; DAHMEN; HÜLSING, 2011) the family function used was $f_K(x) = G(Pad(K)||Pad(x))$, for a key $K \in \{0, 1\}^n$, $x \in \{0, 1\}^n$. $Pad(z) = (z||10^{b-|z|-1})$, for $|z| < b$, where b is the size of the hash function block.

The *XMSS* scheme uses a slightly modified version of the WOTS proposed in (BUCHMANN et al., 2011). This modification makes collision resistance unnecessary: the iterated evaluations of a hash function is replaced by a random walk through the function family F_n, as follows: for $K, x \in \{0, 1\}^n$, $e \in \mathbb{N}$, and $f_K \in F_n$, the function $f_K^e(x)$ is $f_K^0(x) = K$. For $e > 0$, the function is $f_K^e(x) = f_{K'}(x)$, where $K' = f_K^{e-1}(x)$.

Modified WOTS Key pair generation: First compute the Winternitz parameters

$$\ell_1 = \left\lceil \frac{m}{log_2(w)} \right\rceil, \qquad \ell_2 = \left\lfloor \frac{log_2(\ell_1(w-1))}{log_2(w)} \right\rfloor + 1, \qquad L = \ell_1 + \ell_2.$$

The public verification key is

$$Y = (y_1, \ldots, y_L) = (f_{sk_1}^{w-1}(x), \ldots, f_{sk_L}^{w-1}(x)),$$

where sk_i is the private signature key chosen uniformly at random and f^{w-1} as defined above.

Modified WOTS signature generation: This scheme signs messages of binary length m. The message bits are processed in base w representation. The message is $M = (m_1,\ldots,m_{\ell_1})$, $m_i \in \{0, ..., w-1\}$. The checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - m_i)$ in base w representation, and the length ℓ_2 are appended to M, resulting in $b = (b_1, \ldots, b_L)$. The signature is

$$\sigma = (\sigma_1, \ldots, \sigma_L) = (f_{sk_1}^{b_1}(x), \ldots, f_{sk_L}^{b_L}(x)).$$

Modified WOTS verification: To check the signature, the verifier computes the values $b = (b_1, \ldots, b_L)$ as in the signature generation, and then checks the equality

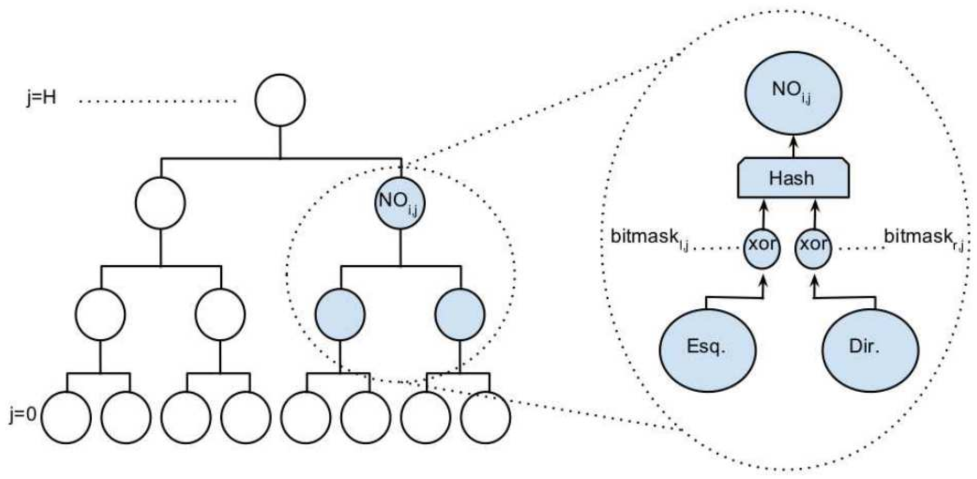
$$(f_{\sigma_1}^{w-1-b_1}(x), \ldots, f_{\sigma_L}^{w-1-b_L}(x)) = (y_1, \ldots, y_L).$$

XMSS public key generation: XMSS is a modification of the Merkle tree (BUCHMANN; DAHMEN; HÜLSING, 2011). A tree of height h has $h + 1$ levels. *XMSS* uses the hash function g_K and bitmasks (bitmaskTree) $(b_{l,j}||b_{r,j}) \in \{0, 1\}^{2n}$, chosen uniformly at random, where $b_{l,j}$ is the left *bitmask* and $b_{r,j}$ is the right *bitmask*. The nodes on level j, $0 \leq j \leq h$, are written $NODE_{i,j}$, $0 \leq i < 2^{h-j}$, and $0 < j \leq h$. The nodes are computed as

$$NODE_{i,j} = g_K((NODE_{2i,j-1} \oplus b_{l,j})||(NODE_{2i+1,j-1} \oplus b_{r,j})).$$

The bitmasks are the main difference to the other Merkle tree constructions, since they allow one to replace the collision resistant hash function family. Figure 20 illustrates how the tree nodes $NODE_{i,j}$ in the *XMSS* scheme are constructed at each level j, to generate the public key of the tree.

Figure 20: XMSS signature scheme



Source: (BARRETO et al., 2014)

To generate a leaf of the *XMSS* tree, an $\mathbb{L}$-tree is used. The one-time public verification keys $(y_1, \ldots, y_L)$ are the first L leaves of an $\mathbb{L}$-tree. If L is not a power of 2, then there are not sufficiently many leaves. A node that has no right sibling is lifted to a higher level of the $\mathbb{L}$-tree until it becomes the right sibling of another node. The hash function uses new bitmasks (bitmask$\mathbb{L}$tree). The bitmask$\mathbb{L}$tree are the same for each of those trees. The *XMSS* public key contains the bitmaskTree, bitmask$\mathbb{L}$tree and the root of the *XMSS* tree.

4.7 Security of the Hash-based Digital Signature Schemes

In this section we present the main results known about the security of hash-based digital signature schemes.

In (BERNSTEIN; BUCHMANN; DAHMEN, 2009, Chapter "Hash-based Digital Signature Schemes"), the Lamport-Diffie one-time signature scheme was proved to be existentially unforgeable under an adaptive chosen message attack (CMA-secure), assuming that the underlying one-way function is preimage resistant. In the same work and also in (CORONADO, 2005), the Merkle signature scheme was also proved has existentially unforgeable under the assumption that the hash function is collision-resistant and the underlying one-time signature scheme has existential unforgeability.

Regarding the security of *XMSS*, (BUCHMANN et al., 2011) proved the following result: if H_n is a second-preimage resistant hash function family and F_n a pseudorandom function family, then *XMSS* is existentially unforgeable under chosen message attacks. In addition, in the same paper, *XMSS* was shown to be forward secure under some modifications on the key generation process.

Hülsing *et. al.* proved that *W-OTS* is existentially unforgeable under adaptive chosen message attacks (HÜLSING, 2013a). The same work showed that the $XMSS^{MT}$ scheme is secure; more specifically, following result is proved: if H_n is a second-preimage resistant hash function family and F_n is a pseudorandom function family, then $XMSS^{MT}$ is a forward secure signature scheme.

4.8 Performance

In this section we present a summary of recent works on the implementation of variants of the Merkle signature scheme.

We use the following notation: time to generate keys (t_{key}), time to generate a signature (t_{sig}) and time to verify a signature (t_{ver}). Table 2 shows timings which were obtained in the following works:

- *CMSS* scheme (BUCHMANN et al., 2006) software implementation on a Pentium M 1.73 GHz, 1 GB of RAM running Microsoft Windows XP for 2^{40} signatures and $w = 3$;
- *GMSS* scheme (BUCHMANN et al., 2007) software implementation on a Pentium dual-core computer 1.8 GHz for 2^{40} signatures, $w_1 = 9$ and $w_2 = 3$ were 390 min, 10.7 ms and 10.7;

- *XMSS* scheme (BUCHMANN; DAHMEN; HÜLSING, 2011) software implementation on an Intel(R) Core (TM) *i*5 *M*540, 2.53GHz computer with Infineon technology;

- *CMSS* scheme (SHOUFAN; HUBER; MOLTER, 2011) hardware implementation on a novel architecture on an FPGA Platform (Virtex-5);

- *XMSS* scheme (OLIVEIRA; LÓPEZ, 2013) software implementation on an Intel Core *i*7 − 2670 QMCPU, 2.20 GHz with 6 GB of RAM.

Table 2: Performance results in the literature

Schemes	**hash**	**h**	**w**	$\mathbf{t}_{key}$	$\mathbf{t}_{sig}$	$\mathbf{t}_{ver}$
CMSS	SHA2	40	3	121 min	40.9 ms	3.7 ms
GMSS	SHA1	40	(9,3)	390 min	10.7 ms	10.7 ms
XMSS	SHA2	20	4	409 s	6.3 ms	0.5 ms
CMSS	SHA2	30	4	820 ms	43.4 ms	1.7 ms
XMSS	SHA2	20	4	553 s	2.7 ms	0.3 ms

Source: (BARRETO et al., 2014)

In Table 2 the size of all public keys is 32 bytes, except for the *XMSS* scheme, that also has to store the bitmasks. The private key and signature are smaller in the *XMSS* scheme, since in the other schemes information storage of more than one tree is required. The *XMSS* scheme presented the best timings for signing and verification on a software implementation, given that only one authentication path needs to be updated and checked for each signature. However, the *XMSS* is only recommended for applications requiring up to 2^{20} signature keys, since the generation of more keys is too time consuming. A Multi Tree *XMSS* ($XMSS^{MT}$) (A. HÜLSING; BUCHMANN, 2013) based on algorithms *CMSS* and *GMSS* is recommended for applications that require a large numbers of signatures.

4.9 Introducing the New Approach

Now we describe the contribution for reducing hash-based signature sizes. To this purpose we redefine the *W-OTS* and the Merkle Signature Scheme in a more convenient way to facilitate the description of our approach.

4.9.1 W-OTS

The Winternitz one-time signature (W-OTS) scheme views the message representative to be signed as a sequence of L w-bit words (or chunks), denoted by $m = (m_0, \dots, m_{L-1})$, where m_i stands for an integer value in the range $0 \dots 2^w - 1$. The signature component for any particular such word m_i will be an m_i-th iterated preimage of some (public) L-word hash value univocally associated to the i-th component of the message representative.

Formally, let $G : \{0,1\}^* \to \{0,1\}^{2n}$ be a collision-resistant hash function and $F : \{0,1\}^* \to \{0,1\}^n$ a preimage-resistant one-way function. Also let $F^k := F \circ F \circ \cdots \circ F$ be F iterated k times.

Message representative preparation:

Let $data \in \{0,1\}^*$ denote the original document to be signed. First compute $M = (m_0, \cdots, m_{\ell_1 - 1}) := G(data)$, where $\ell_1 = \lceil 2n/w \rceil$. Then, compute the checksum $CS := \sum_{i=0}^{\ell_1 - 1} (2^w - 1 - m_i)$ which is then split into w-bit words $(m_{\ell_1}, \cdots, m_{\ell_1 + \ell_2 - 1})$ and appended to M, resulting the message representative $m = M \| CS = (m_0, \dots, m_{L-1})$. The maximum checksum integer value is $(2^w - 1)\ell_1$ which fits in $\ell_2 = \lceil (lg((2^w - 1)\ell_1)/w \rceil$ w-bit words, therefore the total number of w-bit words is $L = \ell_1 + \ell_2$.

The triple of algorithms that defines this scheme is:

- Gen: Choose L strings $s_i \xleftarrow{\$} \{0,1\}^n$ uniformly at random, and compute $v_i \leftarrow F^{2^w - 1}(s_i)$, for $i = 0, \dots, L-1$. The private key is the sequence $(s_0, \dots, s_{L-1})$, and the public key is $v = F(v_0 \parallel \cdots \parallel v_{L-1}) \in \{0,1\}^n$.

- Sig: To sign a message representative $m = (m_0, \ldots, m_{L-1})$, compute $S_i \leftarrow F^{2^w-1-m_i}(s_i)$, and let the signature be the sequence of resulting values $S = (S_0, \ldots, S_{L-1}) \in (\{0,1\}^n)^L$. Notice that S_i is an m_i-th iterated preimage of v_i for all $i = 0, \ldots, L-1$.

- Ver: To verify the signature $S = (S_0, \ldots, S_{L-1})$ of the message representative $m = (m_0, \ldots, m_{L-1})$, compute $t_i = F^{m_i}(S_i)$ for $i = 0, \ldots, L-1$ and check if $v = F(t_0 \parallel \cdots \parallel t_{L-1})$.

An obvious improvement to the scheme is to adopt a short secret string $s \in \{0,1\}^n$ as the private key, and then compute either $s_i \leftarrow F(s \parallel i)$ on demand to reducing memory usage, or else, given a longer hash $K : \{0,1\}^n \to (\{0,1\}^n)^L$ (say, a cryptographic sponge (BERTONI et al., 2007)), set $(s_0, \ldots, s_{L-1}) \leftarrow K(s)$. The first approach is more adequate for memory and speed limited devices since it can be built from a block cipher-based hash function. Usually many embedded devices provide fast SW and sometimes HW AES implementations, so it can be reused for the hash construction. Given the 4KiB SRAM memory and 7.37 MHz budget available in the ATmega128L microcontroller, our implementation focus on the first approach and adopts a software AES.

4.9.2 MSS

The Merkle tree-based signature scheme of height h (defined as the distance between the root and the leaves of the tree, so as to have 2^h leaves and $2^{h+1}-1$ total nodes) and a hash function F extends a one-time signature scheme to 2^h signable messages for each Merkle public key.

The technique consists of generating 2^h one-time key pairs $(s^{(j)}, v^{(j)})$, $0 \leqslant j < 2^h$, for a given one-time signature scheme, and then computing a tree of hash values

$q_1, \ldots, q_{2^{h+1}-1}$ so that $q_i = F(q_{2i} \parallel q_{2i+1})$ for $1 \leqslant i < 2^h$, and $q_{2^h+j} = F(v^{(j)})$ for $0 \leqslant j < 2^h$. The overall public key for the scheme is $Y = q_1$.

Given the one-time signature $S^{(j)}$ verifiable under the public key $v^{(j)}$, the Merkle technique assembles an *authentication path* consisting of the tree nodes whose values are not directly computable from $v^{(j)}$ alone, but are nevertheless needed to compute the values of the parent nodes leading from $v^{(j)}$ to the root Y.

Thus, a Merkle signature is a triple $\Sigma^{(j)} = (S^{(j)}, v^{(j)}, Q^{(j)})$ where $Q^{(j)}$ is the sequence of values $(q_{\lfloor j/2^u \rfloor \oplus 1} \mid u = 0, \ldots, h-1)$ along the authentication path. The Merkle signature length is $z = |S^{(j)}| + |v^{(j)}| + h|q_i|$.

Merkle's construction allows the root value to be computed from an as yet unused (and publicly unknown) $v^{(j)}$, and then compared to Y. This ensures that $v^{(j)}$ is itself authentic, whereby $S^{(j)}$ can be verified as well.

The Merkle-Winternitz scheme, which combines Winternitz one-time signatures with an overall Merkle tree scheme, yields one of the most efficient hash-based signatures. For a Merkle-Winternitz scheme, the signature size is $z = |S^{(j)}| + |v^{(j)}| + h|q_i| = Ln + n + hn = n(L + 1 + h)$ bits.

4.10 Proposed Scheme

Our scheme is related to that by Rohde *et al.* (ROHDE et al., 2008), which we call here REDBP scheme for short. A small conceptual difference is responsible for our reduced signature size and higher processing speeds attainable. Furthermore, we still avail ourselves of the remarks in (EISENBARTH; MAURICH; YE, 2013) to reduce the number of leaf generation to improve time performance.

Key generation requires heavy computational load, since all nodes up to the root must be calculated. Furthermore, many nodes are stored during this process for the setup of REDBP. Therefore, if key generation and signature calculation do not have to take place in the same device, a better memory usage can be achieved by storing some fixed nodes in ROM instead of RAM. Due to this fact, performing key generation on standard PCs is more suitable than on a microcontroller and then enhance the setup of the signing device.

Specifically, the REDBP scheme adopts a hash function $G : \{0,1\}^* \to \{0,1\}^{2n}$ to create digests of the form $M = (m_0, \cdots, m_{\ell_1 - 1}) = G(data), m_i \in \{0, \cdots, 2^w - 1\}$ and produces message representatives of the form $M||CS = (m_0, \cdots, m_{L-1})$ as described in Section 4.9.1, with a straightforward Merkle tree construction on top of Winternitz signatures. For security level roughly 2^n (whereby forging existentially a signature takes about 2^n computational steps), that scheme sets $\ell_1 w = 2n$. Thus the G hash size is twice that of the F hash size. Since each Winternitz signature has length $|S^{(j)}| = nL \approx n\ell_1$ (omitting the checksum size ℓ_2), as a result each Merkle-Winternitz signature has length $z = n(L + 1 + h) \approx n(\ell_1 + 1 + h) = n(2n/w + 1 + h)$.

Intuitively, this is necessary to prevent precomputed collision attacks. Indeed, since only the message *data* is fed into the hash function G, an attacker could mount a Yuval-style attack (YUVAL, 1979), preparing beforehand two sets of semantically equivalent messages, the first favorable to the signer and the second unfavorable, and looking for a collision between a favorable message *data* and an unfavorable one *data*′, finally presenting *data* to the signer and *data*′ to an arbitrating third party after a valid signature is obtained that holds for both messages.

In contrast, we adopt a randomized hash function

$$H : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$$

to create message digests of form $M \leftarrow H(Y, v, data)$.

The presence of Y in the hash is a reflex of the strong Fiat-Shamir heuristic. It would make sense to hash together *all* elements involved in generating a signature, i.e. not only Y, M and v, but the whole authentication path as well. However, Y

itself implicitly contains information on all possible authentication paths, so this does not seem to be major concern. Including the authentication path, however, would not create an efficiency bottleneck, having a modest impact at most.

The one-time verification key v associated to some signed message is only revealed together with the already computed signature for that message. It is therefore not known, and cannot be known let alone chosen by an adversary. The presence of v alone already precludes the possibility of mounting a Yuval-style attack: because v is not known beforehand, the attacker can no longer precompute a collision $H(Y, v, data) = H(Y, v, data')$. Furthermore, once v is revealed with the signature (and will never again be used in another signature because of the one-timeness of the construction), the adversary cannot meaningfully work towards a collision anymore since the message is already signed, being faced with the considerably harder task of finding a preimage $data'$ for it. Apart from this, our scheme inherits all other security properties from the REDBP construction.

As in the REDBP scheme, a signature in our proposal is a triple $\Sigma^{(j)} = (S^{(j)}, v^{(j)}, Q^{(j)})$ where $Q^{(j)}$ is the sequence of values $(q_{\lfloor j/2^u \rfloor \oplus 1} \mid u = 0, \ldots, h-1)$ along the authentication path. However, its length is now $z' = |S'^{(j)}| + |v^{(j)}| + h|q_i| = n(n/w + 1 + h)$ bits.

The signature size ratio between our proposal and the REDBP scheme is

$$z'/z = \frac{n/w + 1 + h}{2n/w + 1 + h}.$$

For practical parameters with $h \approx n/w$, the signatures in our scheme tend to take roughly $2/3$ the size of REDBP signatures, and proportionally even smaller than other proposals in the literature. Furthermore, the best reduction ratios are achieved when $n/w \gg h$. Thus, for a fixed n, the best gain is obtained by choosing the smallest possible w, i.e. $w = 2$. For instance, by adopting the values $(n, w, H) = (128, 2, 16)$ the reduction becomes $z'/z \approx 0.56$, almost halving the signature size.

Also, because both signing and verification involve a loop of length L, the overall number of operations is clearly shorter in our proposal than in the REDBP construction, being a simple consequence of the shorter hash that has to be signed in the Winternitz scheme. That is actually the most time-consuming part of the signature and verification procedures, since each individual w-bit word of the leaf-level hash incurs up to $2^w - 1$ hash computations (or about 2^{w-1} on average).

One particular tradeoff is possible in the scheme of (ROHDE et al., 2008) but not here. Namely, since v_i is not needed when the leaf-level hash is computed in that scheme, it can be postponed until the remainder of the signature is calculated, by completing the iterations needed to go from each piece of the signature to the corresponding input for v_i. In the present variant, v_i has to be known beforehand.

Our proposal can then be summarized as follows, considering only one hierarchical layer for simplicity:

- Gen: Choose $s \overset{\$}{\leftarrow} \{0,1\}^n$ uniformly at random, compute the $L \cdot 2^h$ strings $s_i^{(j)} \leftarrow F(s \parallel i \parallel j)$ and correspondingly the $L \cdot 2^h$ strings $v_i^{(j)} \leftarrow F^{2^w-1}(s_i^{(j)})$, compute $v^{(j)} \leftarrow F(v_0^{(j)} \parallel \cdots \parallel v_{L-1}^{(j)})$, compute the Merkle tree nodes $q_u = F(q_{2u} \parallel q_{2u+1})$ for $1 \leqslant u < 2^h$, and $q_{2^h+j} = F(v^{(j)})$ for $0 \leqslant i < L$, $0 \leqslant j < 2^h$. The private key is s, and the public key is $Y := q_1$, each consisting of L w-bit words[1]. The $s_i^{(j)}$ and $v^{(j)}$ keys as well as the authentication path can be recomputed on demand during a signing operation.

- Sig: To sign the j-th message $data^{(j)}$, compute the message representative $m^{(j)} := (m_0^{(j)}, \ldots, m_{L-1}^{(j)}) \leftarrow H(Y, v^{(j)}, data^{(j)}) \| CS$, compute $s_i^{(j)} \leftarrow F(s \parallel i \parallel j)$ and $S_i^{(j)} \leftarrow F^{2^w-1-m_i}(s_i^{(j)})$ for $0 \leqslant i < L$, compute $S^{(j)} \leftarrow (S_0^{(j)}, \ldots, S_{L-1}^{(j)})$ and the authentication path $Q^{(j)} := (q_{\lfloor j/2^u \rfloor \oplus 1} \mid u = 0, \ldots, h-1)$, and finally let the signature be the triple $(S^{(j)}, v^{(j)}, Q^{(j)})$.

[1]The BDS algorithm, if adopted, would compute some ancillary information to expedite signing as well.

- Ver: To verify a signature $(S^{(j)}, v^{(j)}, Q^{(j)})$ for the j-th message $data^{(j)}$, compute the message representative $m^{(j)} := (m_0^{(j)}, \dots, m_{L-1}^{(j)})$ as before, compute $t_i^{(j)} = F^{m_i^{(j)}}(S_i^{(j)})$ for $0 \leqslant i < L$ and $t^{(j)} \leftarrow F(t_0^{(j)} \parallel \cdots \parallel t_{L-1}^{(j)})$. Then compute the nodes from the j-th leaf to the root via $q_{2^h+j} = F(t^{(j)})$ and $q_i \leftarrow F(q_{2i} \parallel q_{2i+1})$ for $1 \leqslant i < 2^h$, taking the missing nodes from the authentication path $Q^{(j)}$. Accept iff $q_1 = Y$ and $v^{(j)} = t^{(j)}$.

4.11 Efficiency Assessment

Table 3 compares the key and signature sizes of several hash-based signature proposals in the literature (specifically, XMSS (BUCHMANN; DAHMEN; HÜLSING, 2011), XMSS+ (HÜLSING, 2013b), SPR-MSS (DAHMEN et al., 2008), and REDBP (ROHDE et al., 2008)) with several parameterizations of our proposed construction. All sizes are expressed in bytes. For all schemes, n denotes the output hash size, h is the height of the Merkle tree and w is the word size in bits. For simplicity we do not consider more than one hierarchical layer. The somewhat unusual hash sizes do not constitute a problem for a modern sponge-based hash function with capacity at least twice the exponent of the desired security level.

To estimate the security level of Winternitz signatures based on preimage-resistant hash functions as needed in our proposal, we adopt the analysis from (BUCHMANN et al., 2011, Section 5), i.e. the security level for a hash of length $n = \ell_1 w$ bits is at least 2^k for $k = \ell_1 w - w - 1 - 2\lg(\ell_1 w)$. Although we follow this lower bound for the suggested parameters, it might be somewhat too conservative, and smaller signatures could be possible in practice by taking n to be the target security level itself (this seems to be the choice in (ROHDE et al., 2008), since a security level of 2^{128} is claimed rather than 2^{112} for $w = 2$ or 2^{110} for $w = 4$).

Overall, assuming $h = 16$ this amounts to between 1400 (for $w = 8$) and 1900

Table 3: Comparison of hash-based signatures. $|pk|$ and $|sig|$ are in bytes.

scheme	$\|pk\|$	$\|sig\|$	security
XMSS/SHA256($n = 256, h = 20, w = 2$)	1696	4899	2^{210}
XMSS/SHA256($n = 256, h = 20, w = 4$)	1696	2787	2^{196}
ours ($n = 216, h = 16, w = 2$)	27	3510	2^{198}
ours ($n = 216, h = 16, w = 4$)	27	1998	2^{196}
ours ($n = 216, h = 16, w = 8$)	27	1242	2^{192}
REDBP-MSS($n = 128, h = 16, w = 2$)	16	2350	2^{128}
REDBP-MSS($n = 128, h = 16, w = 4$)	16	1330	2^{128}
ours ($n = 144, h = 16, w = 2$)	18	1674	2^{127}
ours ($n = 144, h = 16, w = 4$)	18	1008	2^{125}
ours ($n = 152, h = 16, w = 8$)	19	722	2^{129}
SPR-MSS($n = 128, h = 20$)	928	4416	2^{98}
XMSS+($n = 128, h = 16, w = 2$)	544	3476	2^{96}
XMSS+($n = 128, h = 16, w = 4$)	512	1892	2^{93}
XMSS+($n = 128, h = 16, w = 5$)	480	1588	2^{92}
ours ($n = 120, h = 16, w = 2$)	15	1215	2^{103}
ours ($n = 120, h = 16, w = 4$)	15	750	2^{101}
ours ($n = 120, h = 16, w = 8$)	15	510	2^{97}
XMSS/AES128($n = 128, h = 20, w = 2$)	912	2451	2^{82}
ours ($n = 104, h = 16, w = 2$)	13	949	2^{88}
ours ($n = 104, h = 16, w = 4$)	13	598	2^{86}

Source: Author

(for $w = 4$) bytes at the 2^{128} security level, which is quite acceptable for processors with 4–8 KiB RAM. Those storage requirements reflect a trade-off between storage and processing speed (smaller w leads to much faster signing but takes a somewhat larger space). Even at the higher 2^{192} level the storage stays at less than 3 KiB so it is possible to increase security on many typical IoT platforms.

4.12 Security

The security properties of our scheme inherits most of those from REDBP. Regardless of the required collision-resistance property by the initial hashing of the data to be signed, REDBP points out that forging a MSS signature in practice requires the attacker to compute preimages and second-preimages (ROHDE et al., 2008; NAOR; SHENHAV; WOOL, 2005). Therefore, the security of MSS actually relies on these two properties of the underlying hash function. This enables the adoption of a n-bit output length for F matching the 2^n security level. This already occurs in the literature (ROHDE et al., 2008; BUSOLD, 2012; HÜLSING; BUSOLD; BUCHMANN, 2013).

The main difference between our scheme and REDBP is that our construction randomizes the input of the hash function G (redefined as H in our scheme), thus avoiding Yuval-style collision attacks, which allows the reduction of the output length from $2n$ to n bits. The randomized hashing approach is not new and forgoes the requirement of collision-resistant hash in a hash-then-sign paradigm by adding a nonce to the message before signing it (HALEVI; KRAWCZYK, 2006).

Keeping secret the one-time signature verification key v_i until the computation of the i-th signature is indeed an additional security requirement. Nevertheless it already happens in practice, since the Winternitz keys are (re)generated on-the-fly at signing time. Another observation is that for the signature of the i-th message (when i indexes a left leaf of the tree), its right sibling leaf $F(v_{i+1})$ is in fact revealed in the authentication path. This does not constitute a problem since recovering v_{i+1} from $F(v_{i+1})$ involves computing a preimage on F. Hence the practical impact of keeping v_i secret is minimum.

As other signature schemes that resort to randomized hashing, our scheme also allows a malicious signer to mount a collision attack with $2^{n/2}$ steps, nonetheless this represents no violation of existential unforgeability security notion. As discussed by Halevi *et. al.* in CRYPTO 2006 (HALEVI; KRAWCZYK, 2006), no one but the signer is able to forge a signature, thus he/she cannot disavow a signature by presenting a collision. In this vein, some digital signatures, such as the Schnorr signature scheme (SCHNORR, 1991), do not secure against a malicious signer. In a real world scenario, the signer is likely to be held liable for any message carrying his valid signature, even if the signer is able to show a collision between two messages.

4.13 Implementation and Results

For the IoT candidate, we selected the AVR low-power microcontroller ATmega128L from Atmel which is typically embedded in MICAz sensor nodes, widely employed in Wireless Sensor Networks. ATmega128L has an 8-bit CPU and operates at 7.37 MHz clock speed (MEMSIC, 2012). It provides 4 KiB of SRAM and 128KiB ROM (flash) storage. We show that our solution is practical even on this kind of platform which is more constrained than others evaluated in the literature for the same context of hash-based signatures (ROHDE et al., 2008; EISENBARTH et al., 2013; BUSOLD, 2012).

We adopt an iterative Merkle-Damgård (MD) hash function to instantiate F and H. The advantage of such constructions is the presence of fast block cipher implementations at many constrained platforms. The underlying compression function for MD is Matyas-Meyer-Oseas (MMO) with a single-block input of n bits within Winternitz signature and double-block input of $2n$ bits for the parent computation in the Merkle tree (HÜLSING; BUSOLD; BUCHMANN, 2013). Since F inputs are fixed length, the additional MD padding block for strengthening can be omitted as pointed out in (HÜLSING; BUSOLD; BUCHMANN, 2013). In the case of the hash function H which accepts arbitrary length inputs, the additional padding block is used.

There is an efficiency advantage of Matyas-Meyer-Oseas (MMO) over Davies-Meyer (DM) compression function in the context of single-block inputs. In MMO the block-cipher key input is fed with a fixed IV. This IV would be expanded each time the block cipher performs the key schedule. So, in this case, it is possible to deploy the IV already expanded in compilation time, thus avoiding key schedule procedure (ROHDE et al., 2008). We obtained a reduction in time of about 20% for single-block hash computation using the mentioned approach. In DM mode this optimization can not be performed as the block-cipher key input is fed with the non-fixed message block.

The block cipher used is 128-bit AES. We selected an Assembly (ASM) implementation for AVR devices called RIJNDAELFAST[2]. It is a pure ASM and we had to craft it to work along with C code, i.e. passing keys and plaintexts parameters from C to ASM routines and get the resulting ciphertexts back to C.

The Merkle Tree Traversal algorithm also called TreeHash was adopted to generate the Merkle signatures (BUCHMANN; DAHMEN; SCHNEIDER, 2008). TreeHash algorithm calculates efficiently the authentication path for each leaf on demand. TreeHash is recommended when there is a limited amount of storage available. The idea behind TreeHash algorithm is 1) to save some strategic nodes of the Merkle Tree during the key generation procedure resulting in less leaf computations during the signature generation resulting in considerable time speedups and 2) computing on demand, at each signature procedure, nodes that will be needed for the next authentication paths. To speedup computations, a parameter K is responsible to store strategic nodes (called Retain nodes) during the key generation. Additionally, this algorithm stores an exponential number of Retain nodes during the key generation (i.e., $2^K - K - 1$ nodes), thus K must be selected carefully.

Since we are using a low clock device (one quarter of the clock used in other works (BUSOLD, 2012)), relevant speedups are needed to not compromise usability. The adoption of larger values of the TreeHash parameter K reduces the signature computation time. However, that approach brings a memory penalty which has to be carefully addressed. In the case of a Merkle Tree with height $h = 10$ using the largest possible K for that height, i.e $K = h - 2 = 8$, the TreeHash state (present in RAM) stores $2^8 - 8 - 1 = 247$ Retain nodes. Since each node has 19 bytes [3], the overall storage for Retain is $247 * 19 = 4693$ bytes becoming impractical for the 4 KiB RAM budget.

We circumvented the storage problem by noticing that the Retain nodes in the

[2]Available from: http://point-at-infinity.org/avraes/

[3]In our implementation each node stores 16 bytes for the value, 1 byte for the height and 2 bytes for the index within the height.

TreeHash state are not updated in any signature, i.e. they are always fixed until the very last Merkle signature. Thus, Retain nodes can be viewed as a precomputed table being calculated at key generation. This table does not need to be stored in the TreeHash state as suggested originally, but could be stored apart in ROM (flash) memory with the program code. After applying this procedure, it became possible to deploy higher Merkle trees which can be translated as signing many more messages.

Our experiments show that the maximum possible K for Retain storage in our device is $K = 11$. So, trees up to $h = 13$ can achieve the best choice of the precomputed Retain. Thus, for $h > 13$ processing times begin to degrade. For instance, for $h = 14$ one can not use $K = 12$, and we jump to the next possibility which is $K = 10$ (remember the TreeHash restriction that $h - K$ must be even). This led to a $\approx 86\%$ increase in computation time. The same penalty happens to $h = 15$ which has to use $K = 11$ instead of $K = 13$, and twice of this penalty is paid for $(h = 16, K = 10)$.

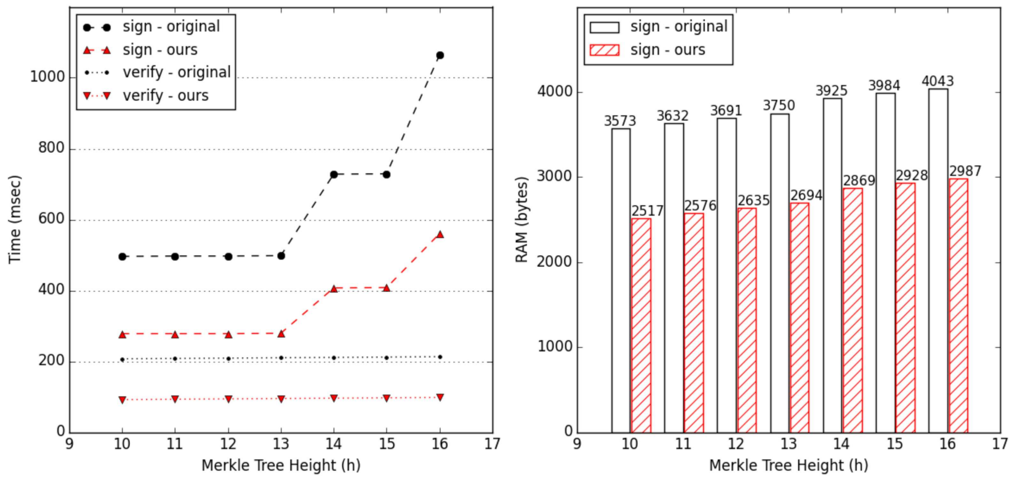
Figure 21 shows our time and memory results. In order to provide a fair evaluation of our proposed improvement we have implemented the Merkle with W-OTS in two settings. We call the first one the *original Merkle with W-OTS* which uses a doubled size hash output $|G| = 2n = 256$, and *our variant Merkle with W-OTS* where $|H| = n = 128$. Also, the results for $h = 16$ on the original scheme are extrapolations, since it runs out of memory for that height.

4.13.1 Detailed Benchmarks

Table 4 gives us a comparison of our variant MSS scheme with the previous one (which uses a larger hash output G, i.e. $|G| = 2n = 256$ as in REDBP). It is important to notice that our implementation prioritizes speed for reducing energy consumption. For achieving that goal we selected suitable parameters K (TreeHash) and w (Winternitz) to speedup computations.

Of course, if one targets smaller code size or RAM usage it is possible to explore

Figure 21: Timings and static RAM comparison of original vs. our variant Merkle with W-OTS schemes on the ATmega128L (@7.37MHz, 4KiB SRAM, 128KiB ROM). The W-OTS parameters are $n = 128$ and $w = 2$.



Source: Author

the memory vs. time tradeoff and drastically reduce these footprints by adopting for example a smaller parameter K (for reducing code size) or increasing parameter w for reducing signature sizes, and therefore RAM consumption.

4.13.2 Energy Measurement

When IoT devices have limited lifetime, which is the case of wireless sensors powered by battery, energy consumption becomes an important metric to take into account by the algorithms. Therefore, we also investigate energy consumption of our implementation running on a MICAz sensor. In order to obtain an accurate measurement of the energy consumption, we perform direct measurements on the target platform.

A Agilent E3631A (AGILENT, 2010) power supply is configured to provide a constant voltage to power the target device. The Agilent 34401A (AGILENT, 2012) digital multimeter (DMM) is used to measure the current flow as the different hardware subsystems become active/inactive during the tasks' execution. Figure 22 shows a block diagram describing our measurement setup: a GPIB (General Purpose Interface Bus -

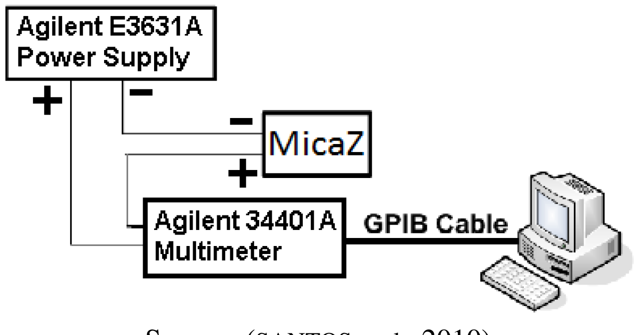
Table 4: Our implementation of original and our variant Merkle with W-OTS scheme on the ATmega128L. The W-OTS parameters are $n = 128$ and word size is $w = 2$. The compression function is Matyas-Meyer-Oseas (MMO).

Parameters	Memory in bytes					Time in ms	
(h,K)	$\lvert sk\rvert$	$\lvert pk\rvert$	$\lvert sig\rvert$	SRAM	ROM	t_{sign}	t_{verify}
Original MSS with W-OTS							
$(h = 10, K = 8)$	16	16	2304	3573	14904	496	207
$(h = 11, K = 9)$	16	16	2320	3632	20024	497	208
$(h = 12, K = 10)$	16	16	2336	3691	29752	497	209
$(h = 13, K = 11)$	16	16	2352	3750	49208	498	210
$(h = 14, K = 10)$	16	16	2368	3925	31908	728	211
$(h = 15, K = 11)$	16	16	2384	3984	49208	729	212
$(h = 16, K = 10)$	———— out-of-RAM-memory ————						
This work							
$(h = 10, K = 8)$	16	16	1264	2517	15198	278	92
$(h = 11, K = 9)$	16	16	1280	2576	23186	278	93
$(h = 12, K = 10)$	16	16	1296	2635	32914	278	94
$(h = 13, K = 11)$	16	16	1312	2694	52114	279	95
$(h = 14, K = 10)$	16	16	1328	2869	32914	407	96
$(h = 15, K = 11)$	16	16	1344	2928	52370	408	97
$(h = 16, K = 10)$	16	16	1360	3103	32914	560	98

Source: Author

IEEE 488 standard) cable is used to connect the Agilent 34401A DMM to a computer running the software LabView (NATIONAL INSTRUMENTS, 2012), which collects and records the measurement samples. The reading rate of DMM is configured to 60 Hz.

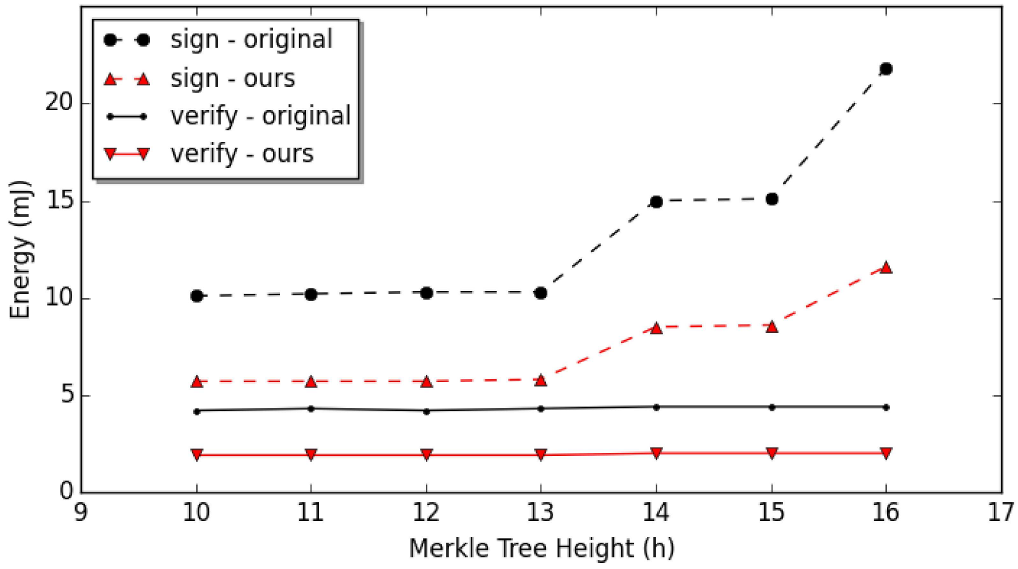
Figure 22: Measurements setup



Source: (SANTOS et al., 2010)

We ran each algorithm and measured the current drained. We obtained the charge via time integration of the current and, since voltage is constant, we obtain the energy consumption. We also measured the system idle current drained (i.e., no particular tasks being executed), obtaining the threshold that is deduced from the measurements. Therefore, the system's overall energy consumption is given by the energy consumed by each task added to the energy consumed when the system is idle. This methodology is described in (SANTOS et al., 2010). Figure 23 shows the energy consumption results.

Figure 23: Energy results of original vs. our variant Merkle with W-OTS. The W-OTS parameters are $n = 128$ and $w = 2$.



Source: Author

4.13.3 Performance Comparison

It is worth to remind that besides the fact that it was possible to obtain shorter signatures, $\approx 1.9\times$ faster signing procedures and $\approx 2.6\times$ faster verification procedures (see Table 5) than the original Merkle with W-OTS signature scheme were achieved using Davies-Meyer compression function. This improvement also leads to comparable reduction factors on energy consumption for those operations in the microcontroller ATMega128L embedded in the MICAz wireless sensor node. We provide additional comparisons with popular signatures based on state-of-the-art elliptic curves, i.e. safe curves (BERNSTEIN; LANGE, 2015). Notice also that for the original Merkle with W-OTS, benchmarks for a Merkle-Tree height $h = 16$ could not be obtained due to a large code size involved, meaning that $h = 15$ is the limit for the previous version. On the other hand this was possible in our new construction.

Table 5: Our implementation of original vs. our variant Merkle with W-OTS signature for the ATmega128L. The W-OTS parameter is $w = 2$. The underlying hash-function is Davies-Meyer (DM) compression function.

Parameters	$\|sig\|$ in bytes	Time in ms		Energy in mJ | Reduction (×)	
(h,K)	$\|sig\|$	t_{sign}	t_{verify}	E_{sign}	E_{verify}
Original Merkle with W-OTS					
$(h = 10, K = 8)$	2304	662	281	13.6 | N.A.	6.0 | N.A.
$(h = 11, K = 9)$	2320	663	281	13.7 | N.A.	6.0 | N.A.
$(h = 12, K = 10)$	2336	663	281	13.9 | N.A.	6.0 | N.A.
$(h = 13, K = 11)$	2352	663	281	14.2 | N.A.	6.1 | N.A.
$(h = 14, K = 10)$	2368	1009	281	19.4 | N.A.	6.1 | N.A.
$(h = 15, K = 11)$	2384	1009	281	21.2 | N.A.	6.1 | N.A.
This work					
$(h = 10, K = 8)$	1264	342	109	7.0 | 1.94×	2.1 | 2.86×
$(h = 11, K = 9)$	1280	342	109	7.1 | 1.93×	2.5 | 2.86×
$(h = 12, K = 10)$	1296	342	109	7.2 | 1.93×	2.4 | 2.50×
$(h = 13, K = 11)$	1312	342	109	7.2 | 1.97×	2.3 | 2.65×
$(h = 14, K = 10)$	1328	520	109	11.0 | 1.97×	2.3 | 2.65×
$(h = 15, K = 11)$	1344	520	109	11.0 | 1.93×	2.2 | 2.77×

Source: Author

Given the possible security issues with DM mode as argued in Section 4.13, we also provide an implementation using the MMO compression function, which in turn is even more efficient than DM applying the trick described also in Section 4.13. Table 6 illustrate the results with MMO.

4.13.4 Comparison with ECC-based Digital Signatures

As widely analyzed in the literature, hash-based signatures are known to be faster than conventional RSA and ECC-based signatures.

We present comparisons of our hash-based signatures with ECC which is usually known to be faster than RSA. Firstly we started our benchmarks with the RELIC (ARANHA; GOUVÊA, 2015) library. After some work we realized that safe curves were not deeply optimized for our target platform in that library. For exam-

Table 6: Our implementation of original vs. our variant Merkle with W-OTS signature for the ATmega128L. The W-OTS parameter is $w = 2$. The underlying compression function is MMO.

Parameters	$\|sig\|$ in bytes	Time in ms		Energy in mJ | Reduction (×)	
(h,K)	$\|sig\|$	t_{sign}	t_{verify}	E_{sign}	E_{verify}
Original Merkle with W-OTS					
$(h = 10, K = 8)$	2304	496	207	10.1 | N.A.	4.2 | N.A.
$(h = 11, K = 9)$	2320	497	208	10.2 | N.A.	4.3 | N.A.
$(h = 12, K = 10)$	2336	497	209	10.3 | N.A.	4.2 | N.A.
$(h = 13, K = 11)$	2352	498	210	10.3 | N.A.	4.3 | N.A.
$(h = 14, K = 10)$	2368	728	211	15.0 | N.A.	4.4 | N.A.
$(h = 15, K = 11)$	2384	729	212	15.1 | N.A.	4.4 | N.A.
$(h = 16, K = 10)$	———— out-of-RAM-memory ————				
This work					
$(h = 10, K = 8)$	1264	278	92	5.7 | 1.77×	1.9 | 2.21×
$(h = 11, K = 9)$	1280	278	93	5.7 | 1.79×	1.9 | 2.26×
$(h = 12, K = 10)$	1296	278	94	5.7 | 1.81×	1.9 | 2.21×
$(h = 13, K = 11)$	1312	279	95	5.8 | 1.78×	1.9 | 2.26×
$(h = 14, K = 10)$	1328	407	96	8.5 | 1.76×	2.0 | 2.20×
$(h = 15, K = 11)$	1344	408	97	8.6 | 1.74×	2.0 | 2.20×
$(h = 16, K = 10)$	1360	560	98	11.6 | NA	2.0 | NA

Source: Author

ple, for the ECDSA over Curve22103 which provides a similar security level to our scheme, signing is 18× slower and verifying is even more drastically 54× slower. This is due to the fact that RELIC does not provide Assembly code for those curves and this comparison is unfair given that our hash-based implementation is a mixed C and Assembly code.

Afterward, we have found the AVRNaCl library[4] (HUTTER; SCHWABE, 2013) which provides tailored Assembly code for the particular curve25519 and that was finally chosen to be included in our benchmarks. AVRNaCl is particularly designed for the 8-bit AVR ATMega family of microcontrollers, which includes the target ATmega128L. AVRNaCl provides assembly procedures in its fastest module in order to accelerate the most critical ECC functions.

AVRNaCl provides two different optimization approaches: *low-area* (memory op-

[4] http://munacl.cryptojedi.org/atmega.shtml

timized) and *high-speed* (processing time optimized). We have chosen the *high-speed* to compare with since run time directly impacts the energy consumption in our target platform. The AVRNaCl *high-speed* module is called **AVRNaCl fast**. In order to benchmark *avrnacl_fast* we deployed it along with the TinyOS operating system which is naturally energy-aware because it is event-oriented, i.e. roughly speaking it stays in idle state (low consumption) until a new event is triggered. TinyOS was also used in our hash-based signature benchmarks for fairness.

We have evaluated two functions from the **AVRNaCl fast** module:

- *crypto_sign_ed25519*: which is the Ed25519 signing procedure,
- *crypto_sign_open*: which is the Ed25519 verification procedure.

We refer the reader to Hutter's work (HUTTER; SCHWABE, 2013) for implementational and bitslicing details used by the AVRNaCl library.

Before performing the benchmarks we calculated the expected theoretical run time for each algorithm. This was possible because Hutter *et. al.* (HUTTER; SCHWABE, 2013) provided the cycle count for both *crypto_sign_ed25519* and *crypto_sign_open*, that is, $23,211,611$ and $32,937,940$ cycles, respectively. Also, according to the MICAz node specification (MEMSIC, 2012, Table 1-2) its embedded ATmega128L microcontroller operates at 7.37 MHz when a $3V$ voltage is applied. Also, ATmega128L delivers 1 MIPS per MHz[5]. Therefore, we are able to calculate the expected timings as being the cycle count divided by the clock frequency. Thus, the theoretical values for signature and verification are $3.15s$ and $4.47s$.

In Table 7 we provide the practical timings collected from our benchmarks over the real platform. One can see that they are $\approx 3.08s$ and $\approx 4.39s$, respectively. The small difference between theoretical and practical values of 2.3% and 1.8% for signing and verifying could be attributed to a possible clock frequency deviation due to a small

[5]http://www.atmel.com/devices/atmega128.aspx

variation in the voltage provided by the source. In fact, we noticed small variations in the voltage (2.99V to 3.01V) detected by the multimeter.

Table 7 highlights in red the improvement factors of run time and energy attained when our solution is compared with conventional ECC signatures. The results show that our solution using DM can achieve a ≈ 6× to 9× smaller energy consumption for signing and a ≈ 40× to 44× factor for verification in the ATmega128L microcontroller.

Table 7: Comparison between our variant Merkle with W-OTS signature and the ECC-based AVRNaCl library for the ATmega128L microcontroller. The underlying hash is the DM compression function.

Parameters	$\|signature\|$ (bytes)	Time (ms) t_{sign}	Time (ms) t_{verify}	Energy (mJ) E_{sign}	Energy (mJ) E_{verify}
		Elliptic Curve Signature			
Curve25519	64	3078 | N.A.	4391 | N.A.	64.2 | N.A.	93.5 | N.A.
		Our variant Merkle with W-OTS			
$h = 10$	1264	342 | 9.0×	109 | 40.3×	7.0 | 9.2×	2.1 | 44.5×
$h = 13$	1312	342 | 9.0×	109 | 40.3×	7.2 | 8.9×	2.3 | 40.7×
$h = 15$	1344	520 | 5.9×	109 | 40.3×	11.0 | 5.8×	2.2 | 42.5×

Source: Author

We also provided a comparison using the MMO compression function in Table 8. The gain in energy is even better than using DM, i.e. reaching 11× for signature generation and 49× for signature verification. Similar speedup factors are achieved in processing times for these operations.

Table 8: Comparison between our variant Merkle with W-OTS signature and the ECC-based AVRNaCl library for the ATmega128L microcontroller. The underlying hash is the MMO compression function.

Parameters	$\|signature\|$ (bytes)	Time (ms) t_{sign}	Time (ms) t_{verify}	Energy (mJ) E_{sign}	Energy (mJ) E_{verify}
		Elliptic Curve Signature			
Curve25519	64	3078 | N.A.	4391 | N.A.	64.2 | N.A.	93.5 | N.A.
		Our variant Merkle with W-OTS			
$h = 10$	1264	278 | 11.1×	92 | 47.7×	5.7 | 11.3×	1.9 | 49.2×
$h = 13$	1312	279 | 11.0×	95 | 46.2×	5.8 | 11.1×	1.9 | 49.2×
$h = 15$	1344	408 | 7.5×	97 | 45.3×	8.5 | 7.6×	2.0 | 46.8×

Source: Author

4.14 Considerations about Hash-based Digital Signatures

We have described a hash-based signature that yields shorter signatures and smaller generation and verification times than the previous state of the art. Our proposal depends on a preimage-resistant (rather than collision-resistant) hash function due to the adoption of randomized hashing using a special nonce.

Our experimental results showed that for a tree of height $h = 10$, a signature can be generated in less than 0.3 s and verification performed in less than 0.1 s on a device running at a low clock rate (7.37 MHz). The static RAM usage is about 2.5 KiB, code size is 15 KiB and signature size is 1.26 KiB for this case. Additionally, a constant-time property can be attained when a constant-time underlying block-cipher is adopted.

We also showed how to explore a memory vs. time tradeoffs from the TreeHash algorithm and how to handle memory constraints when one wants to adopt hash-based digital signatures on very constrained platforms.

Ultimately, our conceptual and practical improvements of MSS with W-OTS were shown to have great potential to become a suitable signature solution for constrained platforms typical of the IoT.

5 CONCLUSIONS AND OPEN PROBLEMS

In this thesis we have described new approaches for the main bottlenecks of two promising candidates for digital signatures in a quantum era, i.e. multivariate quadratic signatures and hash-based signatures.

For multivariate quadratic signatures a new form of parametrization that could reduce UOV and Rainbow key sizes was described. A negative result (polynomial time attack) was found for the case of adopting matrix rings that are too much compact such as circulant matrices. In addition, the centrosymmetric matrix ring, a less compact matrix ring, was analyzed. Besides presenting an apparently exponential behavior of the reconciliation attack for a single-layer construction, the multi-layer version was shown to be weaker for the same attack. This behavior does not matches the expected one, as lacks solid evidence for the attained security. A worse perspective for the target scenario is that the resulting key sizes are still too large for deployment on very constrained devices such as the ones described in the Chapter 4 providing 4KiB of RAM memory (which is smaller than the key size storage required by the Centrosymmetric-UOV for example).

In contrast, there were given positive results concerning hash-based signatures, presented in Chapter 4 which were shown to be much more promising than the ones for multivariate quadratic signatures. Our approach allowed at the same time reducing signature sizes, processing times and energy consumption on a very constrained embedded device (ATmega128L). We believe that our method can be combined not only with the original Merkle with Winternitz signature scheme, but with any other Merkle variant in the literature. This may potentially lead to even better results and enabling for signing a larger amount of messages than the amount provided in our work (2^{16} at most). We leave this evaluation as a future work to be analyzed. Another interesting future work would be to evaluate the energy consumption related to communication of Wireless Sensor Nodes, since we have analyzed only the energy consumption for local processing. We also plan to make our source code available on GitHub in the future.

5.1 Publications

In the book chapters (BARRETO et al., 2013) and (BARRETO et al., 2014) the author reviewed aspects of post-quantum cryptography giving a panorama of the area.

The content of Chapter 3 was presented in 2 posters, "Efficient Code-Based and Multivariate Quadratic Cryptosystems" in Latincrypt 2012 and "Assinatura Digital Pós-Quântica Prática para Dispositivos Limitados" in the "Workshop de Pós-Graduação da Área de Concentração Engenharia de Computação (WPG-EC)" in 2012 which earned the best poster award among all PhD students of computer engineering.

The related paper of the contribution in Chapter 4 is entitled "Shorter hash-based signatures", in collaboration with Paulo Barreto and Cassius Puodzius is accepted to the "JSS 35th Anniversary special issue" in the Journal of Systems and Software.

During his PhD, the author had two other collaborations regarding ECC, subject of his MSc work. (PEREIRA et al., 2013) is an application of author's MSc work, which is a design of an ECC-based framework for secure SMS transmission. It was published in JSS in 2013. The other one is a more theoretical approach (BARRETO et al., 2015) which regards the subgroup security in pairing-based cryptography bulit over ordinary pairing-friendly elliptic curves. The author presented the work in Latincrypt 2015. These 2 papers are summarized respectively in the Appendix A and Appendix B.

REFERENCES

A. HÜLSING, L. R.; BUCHMANN, J. Optimal parameters for $XMSS^{MT}$. In: *CD-ARES 2013*. Regensburg, Germany: Springer, 2013. (Lecture Notes in Computer Science, v. 8128), p. 194–208.

AGILENT. *E363xA Series Programmable DC Power Supplies – Data Sheet*. 2010. Disponível em: <http://cp.literature.agilent.com/litweb/pdf/5968-9726EN.pdf>. Acesso em: 31/08/2015.

AGILENT. *Agilent 34401A Multimeter Uncompromising Performance for Benchtop and System Testing – Data Sheet*. Santa Clara - CA: Agilent Technologies, Inc., 2012. Disponível em: <http://cp.literature.agilent.com/litweb/pdf/5968-0162EN.pdf>. Acesso em: 31/08/2015.

ARAMPATZIS, T.; LYGEROS, J.; MANESIS, S. A survey of applications of wireless sensors and wireless sensor networks. In: IEEE. *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*. Limassol, Cyprus, 2005. p. 719–724.

ARANHA, D. F.; GOUVÊA, C. P. L. *RELIC is an Efficient LIbrary for Cryptography*. 2015. Disponível em: <https://github.com/relic-toolkit/relic>. Acesso em: 31/08/2015.

BARRETO, P. S. L. M.; BIASI, F. P.; DAHAB, R.; LÓPEZ-HERNÁNDEZ, J. C.; PEREIRA, G. C. C. F.; RICARDINI, J. E. Introdução à criptografia pós-quântica. *Minicursos: SBSEG*, p. 46–100, 2013.

BARRETO, P. S. L. M.; BIASI, F. P.; DAHAB, R.; LÓPEZ-HERNÁNDEZ, J. C.; MORAIS, E. M. de; OLIVEIRA, A. D. S. de; PEREIRA, G. C. C. F.; RICARDINI, J. E. A Panorama of Post-quantum Cryptography. In: *Open Problems in Mathematics and Computational Science*. Switzerland: Springer, 2014. p. 387–439.

BARRETO, P. S. L. M.; COSTELLO, C.; MISOCZKI, R.; NAEHRIG, M.; PEREIRA, G. C.; ZANON, G. Subgroup security in pairing-based cryptography. In: LAUTER, K.; RODRÍGUEZ-HENRÍQUEZ, F. (Ed.). *Progress in Cryptology – LATINCRYPT 2015*. [S.l.]: Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9230). p. 245–265. ISBN 978-3-319-22173-1.

BELLARE, M.; GOLDWASSER, S. The complexity of decision versus search. *SIAM Journal on Computing*, v. 23, p. 97–119, 1994.

BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E. *Post-Quantum Cryptography*. Heidelberg, Deutschland: Springer Science & Business Media, 2009.

BERNSTEIN, D. J.; LANGE, T. Batch NFS. In: *Selected Areas in Cryptography–SAC 2014*. Montreal, Quebec: Springer, 2014. p. 38–58.

BERNSTEIN, D. J.; LANGE, T. *SafeCurves: choosing safe curves for elliptic-curve cryptography*. 2015. Disponível em: <http://safecurves.cr.yp.to>. Acesso em: 31/08/2015.

BERTONI, G.; DAEMEN, J.; PEETERS, M.; ASSCHE, G. V. *Sponge functions*. 2007.

BETTALE, L.; FAUGÈRE, J.-C.; PERRET, L. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, v. 3, n. 3, p. 177–197, 2009.

BETTALE, L.; FAUGÈRE, J.-C.; PERRET, L. Solving polynomial systems over finite fields: Improved analysis of the hybrid approach. In: ACM. *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. Grenoble, France, 2012. p. 67–74.

BOEGE, W.; GEBAUER, R.; KREDEL, H. Some examples for solving systems of algebraic equations by calculating Groebner bases. *Journal of Symbolic Computation*, Elsevier, v. 2, n. 1, p. 83–98, 1986.

BRAEKEN, A.; WOLF, C.; PRENEEL, B. A study of the security of Unbalanced Oil and Vinegar signature schemes. In: *Topics in Cryptology–CT-RSA 2005*. San Francisco, USA: Springer, 2005. p. 29–43.

BRENT, R. P. Recent progress and prospects for integer factorisation algorithms. In: *Computing and Combinatorics*. Sydney, Australia: Springer, 2000. p. 3–22.

BUCHMANN, J.; CORONADO, C.; DAHMEN, E.; DÖRING, M.; KLINTSEVICH, E. CMSS – An Improved Merkle Signature Scheme. In: *Progress in Cryptology – INDOCRYPT 2006*. Kolkata, India: Springer, 2006. (Lecture Notes in Computer Science, v. 4329), p. 349–363.

BUCHMANN, J.; DAHMEN, E.; ERETH, S.; HÜLSING, A.; RÜCKERT, M. On the security of the Winternitz one-time signature scheme. In: *Progress in Cryptology – Africacrypt 2011*. Dakar, Senegal: Springer, 2011. (Lecture Notes in Computer Science, v. 6737), p. 363–378.

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 – PQCrypto 2011*. Taipei, Taiwan: Springer, 2011. (Lecture Notes in Computer Science, v. 7071), p. 117–129.

BUCHMANN, J.; DAHMEN, E.; KLINTSEVICH, E.; OKEYA, K.; VUILLAUME, C. Merkle signatures with virtually unlimited signature capacity. In: *Applied Cryptography and Network Security – ACNS 2007*. Zhuhai, China: Springer, 2007. (Lecture Notes in Computer Science, v. 4521), p. 31–45.

BUCHMANN, J.; DAHMEN, E.; SCHNEIDER, M. Merkle tree traversal revisited. In: BUCHMANN, J.; DING, J. (Ed.). *Post-Quantum Cryptography – PQCrypto 2008*. Cincinnati, USA: Springer, 2008. (Lecture Notes in Computer Science, v. 5299), p. 63–78.

BUSOLD, C. *Hash-based Signatures on Smart Cards*. Dissertação (Mestrado) — Technische Universität Darmstadt, 2012.

COFFEY, N. *RSA key lengths*. 2012. Disponível em: <http://www.javamex.com/tutorials/cryptography/rsa_key_length.shtml>. Acesso em: 31/08/2015.

CORONADO, C. On the security and the efficiency of the Merkle signature scheme. *IACR Cryptology ePrint Archive*, v. 2005, p. 192, 2005.

COUVEIGNES, J. M. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, Citeseer, v. 2006, p. 291, 2006.

CZYPEK, P.; HEYSE, S.; THOMAE, E. Efficient implementations of MQPKS on constrained devices. In: *Cryptographic Hardware and Embedded Systems–CHES 2012*. Leuven, Belgium: Springer, 2012. p. 374–389.

DAHMEN, E.; OKEYA, K.; TAKAGI, T.; VUILLAUME, C. Digital signatures out of second-preimage resistant hash functions. In: BUCHMANN, J.; DING, J. (Ed.). *Post-Quantum Cryptography – PQCrypto 2008*. Cincinnati, USA: Springer, 2008. (Lecture Notes in Computer Science, v. 5299), p. 109–123.

DE FEO, L.; JAO, D.; PLÛT, J. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, v. 8, n. 3, p. 209–247, 2014.

DEMILLO, R. A.; DOBKIN, D. P.; JONES, A. K.; LIPTON, R. J. *Foundations of secure computation*. Georgia, USA: Academic Press, 1978.

DIERKS, T.; RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF, August 2008. RFC 5246 (Proposed Standard). (Request for Comments, 5246). Updated by RFCs 5746, 5878, 6176. Disponível em: <http://www.ietf.org/rfc/rfc5246.txt>. Acesso em: 31/08/2015.

DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. *Information Theory, IEEE Transactions on*, IEEE, v. 22, n. 6, p. 644–654, 1976.

DING, J.; SCHMIDT, D. Rainbow, a new multivariable polynomial signature scheme. In: SPRINGER. *Applied Cryptography and Network Security*. New York, USA, 2005. p. 164–175.

DODS, C.; SMART, N.; STAM, M. Hash based digital signature schemes. In: *Cryptography and Coding*. Cirencester, UK: Springer, 2005. (Lecture Notes in Computer Science, v. 3796), p. 96–115.

EISENBARTH, T.; MAURICH, I. von; PAAR, C.; YE, X. A performance boost for hash-based signatures. In: *Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*. Berlin, Germany: Springer, 2013. (Lecture Notes in Computer Science, v. 8260), p. 166–182. http://dx.doi.org/10.1007/978-3-642-42001-6_13.

EISENBARTH, T.; MAURICH, I. von; YE, X. Faster hash-based signatures with bounded leakage. In: *Selected Areas in Cryptography – SAC 2013 – 20th International Conference, Burnaby, BC, Canada, August 14–16, 2013, Revised Selected Papers*. Burnaby, Canada: Springer, 2013. (Lecture Notes in Computer Science, v. 8282), p. 223–243. http://dx.doi.org/10.1007/978-3-662-43414-7_12.

FREY, G.; RÜCK, H.-G. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, v. 62, n. 206, p. 865–874, 1994.

GALBRAITH, S. *Joux kills pairings in characteristic 2*. 2013. Disponível em: <https://ellipticnews.wordpress.com/2013/05/22/joux-kills-pairings-in-characteristic-2/>. Acesso em: 31/08/2015.

GALLAGHER, P.; KERRY, C. *FIPS pub 186-4: Digital signature standard, DSS*. Gaithersburg, USA: NIST, 2013.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman and Company, 1979. ISBN 0-7167-1044-7, 0-7167-1045-5.

GOLDREICH, O.; GOLDWASSER, S.; MICALI, S. How to construct random functions. *Journal of the ACM (JACM)*, ACM, v. 33, n. 4, p. 792–807, 1986.

GOLDWASSER, S.; MICALI, S.; RIVEST, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, SIAM, v. 17, n. 2, p. 281–308, 1988.

GRIEU, F. *How big an RSA key is considered secure today?* 2015. Disponível em: <http://crypto.stackexchange.com/questions/1978/how-big-an-rsa-key-is-considered-secure-today>. Acesso em: 31/08/2015.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: ACM. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. Philadelphia, USA, 1996. p. 212–219.

GROVER, L. K. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, APS, v. 79, n. 2, p. 325, 1997.

GULAMHUSEIN, M. Simple matrix-theory proof of the discrete dyadic convolution theorem. *Electronics Letters*, IET, v. 9, n. 10, p. 238–239, 1973.

GÜNEYSU, T.; LYUBASHEVSKY, V.; PÖPPELMANN, T. Practical lattice-based cryptography: A signature scheme for embedded systems. In: *Cryptographic Hardware and Embedded Systems–CHES 2012*. Leuven, Belgium: Springer, 2012, (Lecture Notes in Computer Science). p. 530–547.

HALEVI, S.; KRAWCZYK, H. Strengthening digital signatures via randomized hashing. In: *Advances in Cryptology-CRYPTO 2006*. Santa Barbara, USA: Springer, 2006. p. 41–59.

HANKERSON, D.; MENEZES, A. J.; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. New York, USA: Springer Science & Business Media, 2006.

HEVIA, A.; MICCIANCIO, D. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In: *Advances in Cryptology – ASIACRYPT 2002*. Queenstown, New Zealand: Springer, 2002. p. 379–396.

HIRSCH, J. *Model S not a car but a sophisticated computer on wheels*. 2015. Disponível em: <http://www.latimes.com/business/autos/la-fi-hy-musk-computer-on-wheels-20150319-story.html>. Acesso em: 31/08/2015.

HOFFSTEIN, J.; PIPHER, J.; SILVERMAN, J. H. NTRU: A new high speed public key cryptosystem. In: *Algorithmic Number Theory Symposium – ANTS III*. Portland, USA: Springer, 1998. (Lecture Notes in Computer Science, v. 1423), p. 267–288.

HÜLSING, A. *Practical forward secure signatures using minimal security assumptions*. Tese (Doutorado) — TU Darmstadt, 2013.

HÜLSING, A. W-OTS+ – shorter signatures for hash-based signature schemes. In: *Progress in Cryptology – AFRICACRYPT 2013*. Cairo, Egypt: Springer, 2013. (Lecture Notes in Computer Science, v. 7918), p. 173–188.

HÜLSING, A.; BUSOLD, C.; BUCHMANN, J. Forward secure signatures on smart cards. In: SPRINGER. *Selected Areas in Cryptography*. Burnaby, Canada, 2013. p. 66–80.

HUTTER, M.; SCHWABE, P. NaCl on 8-Bit AVR Microcontrollers. In: SPRINGER. *AFRICACRYPT*. Cairo, Egypt, 2013. p. 156–172.

JAO, D.; SOUKHAREV, V. Isogeny-based quantum-resistant undeniable signatures. In: *Post-Quantum Cryptography*. Waterloo, Canada: Springer, 2014. p. 160–179.

JOUX, A. Faster index calculus for the medium prime case. application to 1175-bit and 1425-bit finite fields. *IACR Cryptology ePrint Archive*, v. 2012, p. 720, 2012.

JOUX, A. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. *IACR Cryptology ePrint Archive*, v. 2013, p. 95, 2013.

JOUX, A. *Discrete Logarithms in* $GF(2^{6168})$ *[=* $GF((2^{257})^{24})$*]*. 2013. Disponível em: <https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;49bb494e.1305>. Acesso em: 31/08/2015.

JOUX, A. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In: *Advances in Cryptology–EUROCRYPT 2013*. Athens, Greece: Springer, 2013. p. 177–193.

KIPNIS, A.; PATARIN, J.; GOUBIN, L. Unbalanced oil and vinegar signature schemes. In: SPRINGER. *Advances in Cryptology – EUROCRYPT 1999*. Prague, Czech Republic, 1999. p. 206–222.

KIPNIS, A.; SHAMIR, A. Cryptanalysis of the oil and vinegar signature scheme. In: SPRINGER. *Advances in Cryptology – CRYPTO 1998*. Santa Barbara, USA, 1998. p. 257–266.

KLEINJUNG, T.; AOKI, K.; FRANKE, J.; LENSTRA, A. K.; THOMÉ, E.; BOS, J. W.; GAUDRY, P.; KRUPPA, A.; MONTGOMERY, P. L.; OSVIK, D. A. et al. Factorization of a 768-bit RSA modulus. In: *Advances in Cryptology–CRYPTO 2010*. Santa Barbara, USA: Springer, 2010. p. 333–350.

LAMPORT, L. *Constructing digital signatures from a one-way function*. Palo Alto, USA, 1979.

LENSTRA, A. K.; LENSTRA JR, H. W.; MANASSE, M. S.; POLLARD, J. M. *The number field sieve*. Berlin, Germany: Springer, 1993.

LYUBASHEVSKY, V. Lattice signatures without trapdoors. In: *Advances in Cryptology–EUROCRYPT 2012*. Cambridge, UK: Springer, 2012. p. 738–755.

MARGI, C. B.; DE OLIVEIRA, B. T.; DE SOUSA, G. T.; SIMPLICIO JR, M.; BARRETO, P. S.; CARVALHO, T. C.; NÄSLUND, M.; GOLD, R. et al. Impact of operating systems on wireless sensor networks (security) applications and testbeds. In: IEEE. *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*. Zurich, Switzerland, 2010. p. 1–6.

MARGI, C. B.; SIMPLICIO JR, M.; BARRETO, P. S.; CARVALHO, T. Segurança em redes de sensores sem fio. *Minicursos: SBSEG*, p. 149–194, 2009.

MEMSIC. *6020-0065-05 Rev A, MICAz Datasheet*. 2012. Disponível em: <http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html>. Acesso em: 31/08/2015.

MERKLE, R. C. *Secrecy, authentication, and public key systems*. Tese (Doutorado) — Stanford University, 1979.

MERKLE, R. C. A digital signature based on a conventional encryption function. In: SPRINGER. *Advances in Cryptology – CRYPTO 1987*. Santa Barbara, USA, 1988. p. 369–378.

MISOCZKI, R.; TILLICH, J.-P.; SENDRIER, N.; BARRETO, P. S. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: IEEE. *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. Istanbul, Turkey, 2013. p. 2069–2073.

NAOR, D.; SHENHAV, A.; WOOL, A. One-time signatures revisited: Have they become practical? *IACR Cryptology ePrint Archive*, Citeseer, v. 2005, p. 442, 2005.

NAOR, M.; YUNG, M. Universal one-way hash functions and their cryptographic applications. In: ACM. *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. Seattle, USA, 1989. p. 33–43.

NATIONAL INSTRUMENTS. *LabVIEW*. Austin, TX: National Instruments Corporation, 2012. Disponível em: <http://www.ni.com/labview/>. Acesso em: 31/08/2015.

NIST. *Federal Information Processing Standard FIPS 186-3 – Digital Signature Standard (DSS) – 6. The Elliptic Curve Digital Signature Algorithm (ECDSA)*. Gaithersburg, USA: National Institute of Standards and Technology (NIST), 2012. Disponível em: <http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf>. Acesso em: 31/08/2015.

OLIVEIRA, A. K. D. S.; LÓPEZ, J. Implementação em software do esquema de assinatura digital de Merkle e suas variantes. In: *Brazilian Symposium on Information and Computer Systems Security – SBSeg 2013*. Manaus, Brazil: SBC, 2013.

OLIVEIRA, B. T. de; MARGI, C. B.; GABRIEL, L. B. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. In: IEEE. *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*. Cartagena, Colombia, 2014. p. 1–6.

PATARIN, J. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: SPRINGER. *Advances in Cryptology – EUROCRYPT 1996*. Zaragoza, Spain, 1996. p. 33–48.

PATARIN, J. The Oil and Vinegar signature scheme. In: *Dagstuhl Workshop on Cryptography*. Dagstuhl, Germany: [s.n.], 1997. v. 80.

PATARIN, J.; GOUBIN, L. Trapdoor one-way permutations and multivariate polynomials. In: *ICICS'97*. Beijing, China: Springer, 1997. (Lecture Notes in Computer Science, v. 1334), p. 356–368.

PATARIN, J.; GOUBIN, L.; COURTOIS, N. Improved algorithms for isomorphisms of polynomials. In: *Advances in Cryptology – EUROCRYPT 1998*. Espoo, Finland: Springer, 1998. p. 184–200.

PEREIRA, G. C.; SANTOS, M. A.; DE OLIVEIRA, B. T.; SIMPLICIO JR, M. A.; BARRETO, P. S.; MARGI, C. B.; RUGGIERO, W. V. SMSCrypto: A lightweight cryptographic framework for secure SMS transmission. *Journal of Systems and Software*, Elsevier, v. 86, n. 3, p. 698–706, 2013.

PETZOLDT, A.; BUCHMANN, J. A multivariate signature scheme with an almost cyclic public key. *IACR Cryptology ePrint Archive*, v. 2009, p. 440, 2009.

PETZOLDT, A.; BULYGIN, S.; BUCHMANN, J. Cyclicrainbow–a multivariate signature scheme with a partially cyclic public key. In: *Progress in Cryptology-INDOCRYPT 2010*. Hyderabad, India: Springer, 2010. p. 33–48.

PETZOLDT, A.; BULYGIN, S.; BUCHMANN, J. Selecting parameters for the Rainbow signature scheme. In: *Post-Quantum Cryptography*. Darmstadt, Germany: Springer, 2010. p. 218–240.

PETZOLDT, A.; BULYGIN, S.; BUCHMANN, J. Linear recurring sequences for the UOV key generation. In: *Public Key Cryptography–PKC 2011*. Taormina, Italy: Springer, 2011. p. 335–350.

PETZOLDT, A.; THOMAE, E.; BULYGIN, S.; WOLF, C. Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems. In: *Cryptographic Hardware and Embedded Systems–CHES 2011*. Nara, Japan: Springer, 2011. p. 475–490.

PUCCINELLI, D.; HAENGGI, M. Wireless Sensor Networks: applications and challenges of ubiquitous sensing. *Circuits and Systems Magazine, IEEE*, IEEE, v. 5, n. 3, p. 19–31, 2005.

ROHDE, S.; EISENBARTH, T.; DAHMEN, E.; BUCHMANN, J.; PAAR, C. Fast hash-based signatures on constrained devices. In: GRIMAUD, G.; STANDAERT, F.-X. (Ed.). *International Conference on Smart Card Research and Advanced Applications – CARDIS 2008*. Ehgam, UK: Springer, 2008. (Lecture Notes in Computer Science, v. 5189), p. 104–117.

ROMPEL, J. One-way functions are necessary and sufficient for secure signatures. In: ACM. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. Philadelphia, USA, 1990. p. 387–394.

SANTOS, M. A.; MARGI, C. B.; SIMPLÍCIO JR, M.; PEREIRA, G. C.; OLIVEIRA, B. T. de. Implementation of data survival in unattended Wireless Sensor Networks using cryptography. In: IEEE. *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. Denver, USA, 2010. p. 961–967.

SATOH, T. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Math. Univ. Sancti Pauli*, v. 47, n. 1, p. 81–92, 1998.

SAWERS, P. *Android users have an average of 95 apps installed on their phones, according to Yahoo Aviate data*. Disponível em: <http://thenextweb.com/apps/2014/08/26/android-users-average-95-apps-installed-phones-according\-yahoo-aviate-data>. Acesso em: 31/08/2015.

SCHNORR, C.-P. Efficient signature generation by smart cards. *Journal of cryptology*, Springer, v. 4, n. 3, p. 161–174, 1991.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, v. 26, p. 1484–1509, 1997.

SHOUFAN, A.; HUBER, N.; MOLTER, H. A novel cryptoprocessor architecture for chained Merkle signature scheme. In: *Microprocessors and Microsystems*. Netherlands: Elsevier, 2011. p. 34–47.

STOLBUNOV, A. *Cryptographic schemes based on isogenies*. Tese (Doutorado) — Norwegian University of Science and Technology (NTNU), 2012.

SUN, X.; TIAN, H.; WANG, Y. Toward quantum-resistant strong designated verifier signature. *International Journal of Grid and Utility Computing 4*, Inderscience Publishers Ltd, v. 5, n. 2, p. 80–86, 2014.

SZYDLO, M. Merkle tree traversal in log space and time. In: *Advances in Cryptology – Eurocrypt 2004*. Interlaken, Switzerland: Springer, 2004. (Lecture Notes in Computer Science, v. 3027), p. 541–554.

THOMAE, E. A Generalization of the Rainbow Band Separation Attack and its Applications to Multivariate Schemes. *IACR Cryptology ePrint Archive*, v. 2012, p. 223, 2012.

THOMAE, E. Quo vadis quaternion? cryptanalysis of Rainbow over non-commutative rings. In: *Security and Cryptography for Networks*. Amalfi, Italy: Springer, 2012. p. 361–373.

THOMAE, E.; WOLF, C. Solving underdetermined systems of multivariate quadratic equations revisited. In: *Practice and Theory in Public Key Cryptography – PKC 2012*. Darmstadt, Germany: Springer, 2012. (Lecture Notes in Computer Science, v. 7293), p. 156–171.

UNE, M.; KANDA, M. Year 2010 issues on cryptographic algorithms. *Monetary and Economic Studies*, Institute for Monetary and Economic Studies, Bank of Japan, v. 25, n. 1, p. 129–164, 2007.

Wikipedia. *Merkle tree — Wikipedia, The Free Encyclopedia*. 2015. Disponível em: <https://en.wikipedia.org/wiki/Merkle_tree>. Acesso em: 31/08/2015.

WOLF, C.; PRENEEL, B. Taxonomy of public key schemes based on the problem of multivariate quadratic equations. *IACR Cryptology ePrint Archive*, Citeseer, v. 2005, p. 77, 2005.

YASUDA, T.; SAKURAI, K.; TAKAGI, T. Reducing the key size of Rainbow using non-commutative rings. In: *Topics in Cryptology – CT-RSA 2012*. San Francisco, USA: Springer, 2012. p. 68–83.

YUVAL, G. How to swindle Rabin. *Cryptologia*, v. 3, p. 187–190, 1979.

APPENDIX A – SMSCRYPTO: A LIGHTWEIGHT CRYPTOGRAPHIC FRAMEWORK FOR SECURE SMS TRANSMISSION

This paper was published in Journal of Systems and Software in 2013. It can be shortly described as follows.

Despite the continuous growth in the number of smartphones around the globe, Short Message Service (SMS) still remains as one of the most popular, cheap and accessible ways of exchanging text messages using mobile phones. Nevertheless, the lack of security in SMS prevents its wide usage in sensitive contexts such as banking and health-related applications. Aiming to tackle this issue, this paper presents SMSCrypto, a framework for securing SMS-based communications in mobile phones. SMSCrypto encloses a tailored selection of lightweight cryptographic algorithms and protocols, providing encryption, authentication and signature services. The proposed framework is implemented both in Java (target at JVM-enabled platforms) and in C (for constrained SIM Card processors) languages, thus being suitable for a wide range of scenarios. In addition, the signature model adopted does not require an on-line infrastructure and the inherent overhead found in the Public Key Infrastructure (PKI) model, facilitating the development of secure SMS-based applications. The proposed framework is evaluated on a real phone and on SIM Card-comparable microcontroller.

The contribution in the paper is the design of SMSCrypto, a security framework especially tailored for protecting SMS-based applications. Unlike many proposals found in the literature, SMSCrypto not only includes lightweight algorithms based on Elliptic Curve Cryptography (ECC) in order to overcome the need of auxiliary storage/processing devices, but also provides protocols in the certificateless Baek, Safavi-Naini and Susilo (BSS) model that avoids the need of a full-fledged PKI. The efficiency of the solution is evaluated through benchmarks on real platforms.

APPENDIX B - SUBGROUP SECURITY IN PAIRING-BASED CRYPTOGRAPHY

This paper was accepted to Latincrypt 2015 and can be shortly described as follows.

Pairings are typically implemented using ordinary pairing-friendly elliptic curves. The two input groups of the pairing function are groups of elliptic curve points, while the target group lies in the multiplicative group of a large finite field. At moderate levels of security, at least two of the three pairing groups are necessarily proper subgroups of a much larger composite-order group, which makes pairing implementations potentially susceptible to small-subgroup attacks.

To minimize the chances of such attacks, or the required effort to thwart them, we put forward a property for ordinary pairing-friendly curves called *subgroup security*. We point out that existing curves in the literature and in publicly available pairing libraries fail to achieve this notion, and propose a list of replacement curves that do offer subgroup security. These curves were chosen to drop into existing libraries with minimal code change, and to sustain state-of-the-art performance numbers. In fact, there are scenarios in which the replacement curves could facilitate faster implementations of protocols because they can remove the need for expensive group exponentiations that test subgroup membership.

In this paper new instances of pairing-friendly elliptic curves that aim to provide stronger resistance against small-subgroup attacks are proposed. A small-subgroup attack can be mounted on a discrete-logarithm-based cryptographic scheme that uses a prime-order group which is contained in a larger group of order divisible by small prime factors. By forcing a protocol participant to carry out an exponentiation of a non-prime-order group element with a secret exponent, an attacker could obtain information about that secret exponent. This is possible if the protocol implementation does not check that the group element being exponentiated belongs to the correct subgroup and thus has large prime order. In the worst case, the user's secret key could be fully revealed although the discrete logarithm problem (DLP) in the large prime-order subgroup is computationally infeasible. We start by illustrating the possibility of such attacks in the context of (pairing-based) *digital signature schemes*, many of which are based on the celebrated short signature scheme of Boneh, Lynn and Shacham (BLS).