Efficient Implementations of MQPKS on Constrained Devices

Peter Czypek, Stefan Heyse, and Enrico Thomae

Horst Görtz Institute for IT Security
Ruhr University Bochum
44780 Bochum, Germany
{peter.czypek,stefan.heyse,enrico.thomae}@rub.de

Abstract. Multivariate Quadratic Public Key Schemes (MQPKS) attracted the attention of researchers in the last decades for two reasons. First they are thought to resist attacks by quantum computers and second, most of the schemes were broken. The latter may be the reason why implementations are rare. This work investigates one of the most promising member of MQPKS and its variants, namely UOV, Rainbow and enTTS. UOV resisted all kinds of attacks for 13 years and can be considered one of the best examined MQPKS. We describe implementations of UOV, Rainbow and enTTS on an 8-bit microcontroller. To address the problem of large keys, we used several optimizations and also implemented the 0/1-UOV scheme introduced at CHES 2011. To achieve a practically usable security level on the selected device, all recent attacks are summarized and parameters for standard security levels are given. To allow judgement of scaling, the schemes are implemented for the most common security levels in embedded systems 2^{64}, 2^{80} and 2^{128} bits symmetric security. This allows for the first time a direct comparison of the four schemes because they are implemented for exactly the same security levels on the same platform and also by the same developer.

Keywords: Multivariate Quadratic Signatures, MQ, Unbalanced Oil and Vinegar, UOV, Rainbow, enTTS, AVR, Embedded Device.

1 Introduction

Since Peter Shor published efficient quantum algorithms [20] to solve the problem of factorization and discrete logarithm in 1995, there is a increasing demand in investigating possible alternatives. One such class of so-called post-quantum cryptosystems is based on multivariate quadratic ($\mathcal{MQ}$) polynomials. We know that solving systems of $\mathcal{MQ}$-polynomials is hard in the worst case, as the corresponding $\mathcal{MQ}$-problem is proven to be $\mathcal{NP}$-complete [11]. Unfortunately all schemes proposed so far also need the Isomorphism of Polynomials (IP) problem to hide the trapdoor. It is not known how hard this problem is and indeed most $\mathcal{MQ}$-schemes are broken this way. So for example, the balanced Oil and Vinegar scheme [15], Sflash [4] and much more [17,16,12,7,8,23]. To encapsulate, nearly all $\mathcal{MQ}$-encryption schemes and most of the $\mathcal{MQ}$-signature schemes are

E. Prouff and P. Schaumont (Eds.): CHES 2012, LNCS 7428, pp. 374–389, 2012.

broken up to this point. There are only very few exceptions like the signature schemes HFE$^-$, Unbalanced Oil and Vinegar (UOV) and its layer based variants Rainbow and enTTS. Well, breaking the first seems to be a matter of time as some ideas of the attack against Sflash from Asiacrypt 2011 [4] might also be applicable. On the other hand, UOV resisted all kinds of attacks for 13 years. It is thought to be the most promising member of the class of $\mathcal{MQ}$-schemes.

Previous Work and Contribution. Rainbow type hardware implementations got some attention during the last years. An 0.35µm ASIC, which signs in 0.012 ms, is reported in [2]. Further [21] presents an ASIC implementation, taking only 198 clock cycles for a sign operation. An ASIC implementation of enTTS(20,28) enabling sign in 0.044 seconds running at a slow clock of 100KHz, is reported in [25]. The authors also report a MSP430 implementation signing in 71 ms and verifying in 726 ms and a 8051-compatible µC implementation signing in 198ms. At CHES 2004, Yang et al. describe an implementation of TTS targetting 8051-compatible µCs [1]. Their implementation of TTS(20,28) signs in 144ms, 170ms, 60ms and for TTS(24,32) they achieve 191ms, 227 ms, 85 ms for an i8032AH, i8051AH and W77E59, respectively. We are not aware of any implementation of UOV or Rainbow targeting small microcontrollers.

This work describes implementations of the $\mathcal{MQ}$-signature schemes, UOV, Rainbow and enTTS, on an 8-bit microcontroller. Additionally, methods to reduce the key size are evaluated and a version of UOV published at CHES 2011 (0/1-UOV [19]) is introduced and also evaluated. To achieve a practically usable security level on the selected device, recent attacks are summarized and parameters for standard security levels are given. The actual implementations were all done by the same developer. This ensures, that we really compare different schemes and not just different skills of different developers.

Organization. Section 2 introduces $\mathcal{MQ}$-schemes in general and UOV, Rainbow and enTTS in special. Section 3 summaries recent attacks and derives parameter sets to achieve $2^{64}, 2^{80}$ and 2^{128} bit security. Afterwards, Section 4 describes our implementations before we present our results in Section 5. Finally, we conclude in Section 6 and point out some details for future improvements.

2 Multivariate Quadratic Public Key Cryptosystems

This section provides a brief introduction to UOV [14], 0/1 UOV [19], Rainbow [9] and enTTS [24]. The general idea of all these $\mathcal{MQ}$-signature schemes is to use a public multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with

$$\mathcal{P} = \begin{pmatrix} p^{(1)}(x_1, \ldots, x_n) \\ \vdots \\ p^{(m)}(x_1, \ldots, x_n) \end{pmatrix}$$

and

$$p^{(k)}(x_1, \ldots, x_n) := \sum_{1 \le i \le j \le n} \alpha_{ij}^{(k)} x_i x_j = x^{\intercal} \mathfrak{P}^{(k)} x,$$

where $\mathfrak{P}^{(k)}$ is the $(n \times n)$ matrix describing the quadratic form of $p^{(k)}$ and $x = (x_1, \ldots, x_n)^\intercal$. Note that we can neglect linear and constant terms as they never mix with quadratic terms and thus do not increase the security [5].

The trapdoor is given by a structured central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with

$$\mathcal{F} = \begin{pmatrix} f^{(1)}(u_1, \ldots, u_n) \\ \vdots \\ f^{(m)}(u_1, \ldots, u_n) \end{pmatrix}$$

and

$$f^{(k)}(u_1, \ldots, u_n) := \sum_{1 \le i \le j \le n} \gamma_{ij}^{(k)} u_i u_j = u^\intercal \mathfrak{F}^{(k)} u.$$

In order to hide this trapdoor we choose two secret linear transformations S, T and define $\mathcal{P} := T \circ \mathcal{F} \circ S$. See Figure 1 for an illustration.

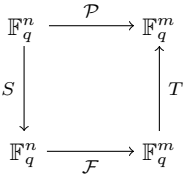

Fig. 1. $\mathcal{MQ}$-Scheme in general

Unbalanced Oil and Vinegar. For the UOV signature scheme the variables u_i, $i \in V := \{1, \ldots, v\}$ are called *vinegar variables* and the remaining variables u_i, $i \in O := \{v+1, \ldots, n\}$ are called *oil variables*. The central map $\mathcal{F}$ is given by

$$f^{(k)}(u_1, \ldots, u_n) := \sum_{i \in V, j \in V} \gamma_{ij}^{(k)} u_i u_j + \sum_{i \in V, j \in O} \gamma_{ij}^{(k)} u_i u_j.$$

The corresponding matrix $\mathfrak{F}^{(k)}$ is depicted in Figure 2.

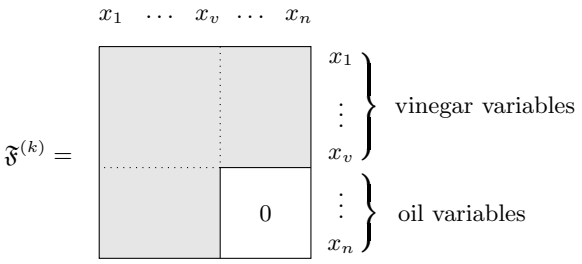

Fig. 2. Central map $\mathfrak{F}$ of UOV. White parts denote zero entries while gray parts denote arbitrary entries.

As we have m equations in $m + v$ variables, fixing v variables will yield a solution with high probability. Due to the structure of $\mathfrak{F}^{(k)}$, *i.e.* there are no quadratic terms of two oil variables, we can fix the vinegar variables at random to obtain a system of linear equations in the oil variables, which is easy to solve. This procedure is not possible for the public key, as the transformation S of variables fully mixes the variables (like oil and vinegar in a salad). Note that for UOV we can discard the transformation T of equations, as the trapdoor is invariant under this linear transformation.

Rainbow. Rainbow uses the same idea as UOV but in different layers. Current choices of parameters (q, v_1, o_1, o_2) use two layers, as it turned out to be the best choice in order to prevent MinRank attacks and preserve short signatures at the same time. We will use $q = 2^8$ throughout the paper. The central map $\mathcal{F}$ of Rainbow is divided into two layers $\mathfrak{F}^{(1)}, \ldots, \mathfrak{F}^{(o_1)}$ and $\mathfrak{F}^{(o_1+1)}, \ldots, \mathfrak{F}^{(o_1+o_2)}$ of form given in Figure 3.

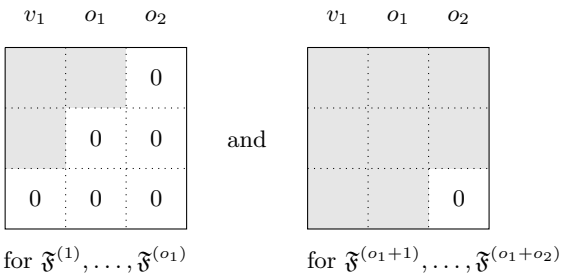

Fig. 3. Central map of Rainbow (q, v_1, o_1, o_2). White parts denote zero entries while gray parts denote arbitrary entries.

To use the trapdoor we first solve the small UOV system $\mathfrak{F}^{(1)}, \ldots, \mathfrak{F}^{(o_1)}$ by randomly fixing the v_1 vinegar variables. The solution $u_1, \ldots, u_{v_1+o_1}$ is now used as vinegar variables of the second layer. Solving the obtained linear system yields $u_{v_1+o_1+1}, \ldots, u_{v_1+o_1+o_2}$. A formal description of Rainbow is given by the following formula.

$$f^{(k)}(u_1, \ldots, u_n) := \sum_{i \in V_1, j \in V_1} \gamma_{ij}^{(k)} u_i u_j + \sum_{i \in V_1, j \in O_1} \gamma_{ij}^{(k)} u_i u_j \quad \text{for } k = 1, \ldots, o_1$$

$$f^{(k)}(u_1, \ldots, u_n) := \sum_{i \in V_1 \cup O_1, j \in V_1 \cup O_1} \gamma_{ij}^{(k)} u_i u_j + \sum_{i \in V_1 \cup O_1, j \in O_2} \gamma_{ij}^{(k)} u_i u_j \quad \text{for } k = o_1 + 1, \ldots, o_1 + o_2$$

0/1-Unbalanced Oil and Vinegar. At CHES 2011 Petzold *et al.* [19] showed that large parts of the public key are redundant in order to prevent key recovery

attacks. More precisely, S can be chosen of a special structure due to equivalent keys and thus large parts of the public and secret map are equal. Choosing this parts of $\mathcal{P}$ of a special structure, such that direct attacks on the public key do not become easier, they were able to reduce the key size and running time of the verification algorithm.

Enhanced TTS. Enhanced TTS was proposed by Yang and Chen in 2005 [24]. The general idea is the same as for Rainbow, but as TTS was designed for high speed implementation it uses as few monomials as possible. For the purpose of evaluating the security we generalize the scheme by adding more monomials. As soon as a monomial $x_i x_j$ with $x_i \in U$ and $x_j \in V$ occur in the original TTS polynomial, we just assume that all monomials $x_i x_j$ with $x_i \in U$ and $x_j \in V$ occur. This way we easily see that TTS is a very special case of the Rainbow signature scheme. There are two different scalable central maps given in [24], one is called *even* sequence and the other *odd* sequence. The following equations show the odd sequence. We restrict our implementation to this case.

$$f^{(i)} = u_i + \sum_{j=1}^{2\ell-3} \gamma_{ij} u_j u_{2\ell-2+(i+j+1 \bmod 2\ell-1)} \qquad \text{for } 2\ell-2 \le i \le 4\ell-4,$$

$$f^{(i)} = u_i + \sum_{j=1}^{\ell-2} \gamma_{ij} u_{i+j-(4\ell-3)} u_{i-j-2\ell} + \sum_{j=\ell-1}^{2\ell-3} \gamma_{ij} u_{i+j-3\ell+3} u_{i-j+\ell-2}$$
$$\text{for } i = 4\ell-3, 4\ell-2,$$

$$f^{(i)} = u_i + \gamma_{i0} u_{i-2\ell+1} u_{i-2\ell-1} + \sum_{j=4\ell-1}^{i-1} \gamma_{i,j-(4\ell-2)} u_{2(i-j)-(i \bmod 2)} u_j + \gamma_{i,i-4\ell+2} u_0 u_i$$
$$+ \sum_{j=i+1}^{6\ell-3} \gamma_{i,j-(4\ell-2)} u_{4\ell-1+i-j} u_j \qquad \text{for } 4\ell-1 \le i \le 6\ell-3.$$

If we generalize these equations to the Rainbow signature scheme, the central map is given by Figure 4.

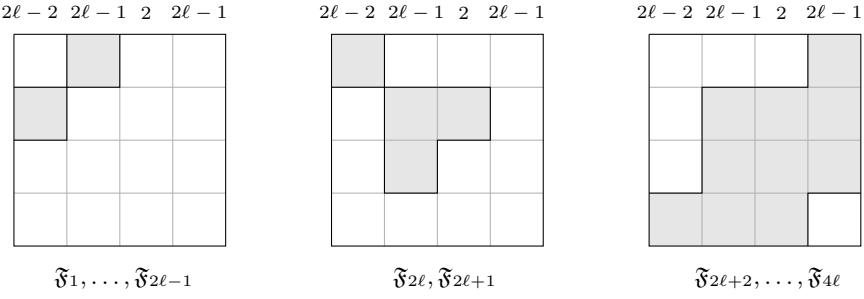

Fig. 4. Secret map $\mathcal{F}$ of odd sequence Enhanced TTS generalized

3 Security in a Nutshell

To provide a fair comparison between UOV, Rainbow and enTTS regarding memory consumption and running time, we first have to choose parameters of the same level of security. Therefore we briefly revisit the latest attacks and choices of parameters of all three schemes.

3.1 Security and Parameters of UOV and 0/1-UOV

Direct Attack. To forge a single signature an attacker would have to solve a system of o quadratic equations in v variables over $\mathbb{F}_q$. The usual way of finding one solution is first guessing v variables at random. This preserves one solution with high probability. The best way of solving the remaining $\mathcal{MQ}$-system of o equations and variables is to guess a few further variables and then apply some Gröbner Basis algorithm like F_4 (see Hybrid Approach of Bettale *et al.* [3]). Recently Thomae *et al.* showed that we can do better than guessing v variables at random [22]. Calculating these v variables through linear systems of equations allows to solve a system of $o - \left\lfloor \frac{v}{o} \right\rfloor$ quadratic equations and variables afterwards. To determine the complexity of solving a $\mathcal{MQ}$-system using a Groebner basis algorithm like F_4 we refer to [3]. In a nutshell, we first have to calculate the degree of regularity d_{reg}. For semi-regular sequences, which generic systems are assumed to be, the degree of regularity is the index of the first non-positive coefficient in the Hilbert series $S_{m,n}$ with

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

where d_i is the degree of the i-th equation. Then the complexity of solving a zero-dimensional (semi-regular) system using F_4 [3, Prop. 2.2] is

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

with $2 \leq \alpha \leq 3$ the linear algebra constant. We used $\alpha = 2$ throughout the paper.

Key Recovery Attacks. There are two key recovery attacks known so far. The first is a purely algebraic attack called *Reconciliation* attack [6]. In order to obtain the secret key S we have to solve $\binom{k+1}{2}o$ quadratic equations in kv variables for an optimal parameter $k \in \mathbb{N}$. The second attack is a variant of the *Kipnis-Shamir* attack on the balanced Oil and Vinegar scheme [15]. The overall complexity of this attack is $\mathcal{O}(q^{v-o-1}o^4)$. Note that $v = 2o$ is very conservative in order to prevent this attack and thus v can be chosen much smaller for o large enough. As $k \geq 2$ even the Reconciliation attack will not badly benefit of choosing v smaller and direct attacks even suffer of such a choice.

Table 1. Minimal 0/1-UOV parameters achieving certain levels of security. Thereby g is the optimal number of variables to guess in the hybrid approach and k is the optimal parameter selectable for the Reconciliation attack.

security	parameter (o, v)	direct attack	Reconciliation	Kipnis-Shamir
2^{64}	$(21, 28)$	2^{67} $(g = 1)$	2^{131} $(k = 2)$	2^{66}
2^{80}	$(28, 37)$	2^{85} $(g = 1)$	2^{166} $(k = 2)$	2^{83}
2^{128}	$(44, 59)$	2^{130} $(g = 1)$	2^{256} $(k = 2)$	2^{134}

3.2 Security and Parameters of Rainbow

All attacks against UOV also apply to Rainbow. Additionally the security of Rainbow relies on the MinRank-problem. Thus we also have to take MinRank and HighRank attacks, as well as the Rainbow Band Separation attack into account. See Petzold *et al.* [18] for an overview of the attacks and the parameters to choose.

Table 2. Minimal Rainbow parameters achieving certain levels of security. Thereby g is the optimal number of variables to guess for the hybrid approach.

security	(v_1, o_1, o_2)	direct attack	Band	MinRank	HighRank	Kipnis	Reconciliation
2^{64}	$(15, 10, 10)$	2^{67} $(g = 1)$	2^{70}	2^{141}	2^{93}	2^{125}	2^{242} $(k = 6)$
2^{80}	$(18, 13, 14)$	2^{85} $(g = 1)$	2^{81}	2^{167}	2^{126}	2^{143}	2^{254} $(k = 5)$
2^{128}	$(36, 21, 22)$	2^{131} $(g = 2)$	2^{131}	2^{313}	2^{192}	2^{290}	2^{523} $(k = 7)$

3.3 Security and Parameters of Enhanced TTS

All attacks against Rainbow also apply to enTTS. The only attack that seriously benefit from the changes made between Rainbow and enTTS is the Reconciliation attack with large k. But as the complexities of this attacks are out of reach anyway this do not affect the security. Actually the complexity is higher than the ones of all the other attacks, so we omit it. More important is the slight benefit of the Band Separation attack. For the odd sequence enTTS we derive $m + n - 1$ quadratic equations in $n - 2$ instead of n variables.

4 Implementation on AVR Microprocessors

The goal of these implementations is a fair comparison between some of the most promising $\mathcal{MQ}$-based post quantum public key schemes. All schemes were analysed in the previous section and sets of parameters with equivalent security were defined under considerations of most recent attacks. A problem when comparing such schemes is that every implementation has its own philosophy of what

Table 3. Minimal odd sequence enTTS parameters achieving certain levels of security. Thereby g is the optimal number of variables to guess for the hybrid approach.

security	(ℓ, m, n)	direct attack	Band	MinRank	HighRank	Kipnis-Shamir
2^{64}	$(7, 28, 40)$	2^{89} $(g = 1)$	2^{68}	2^{126}	2^{117}	2^{127}
2^{80}	$(9, 36, 52)$	2^{110} $(g = 2)$	2^{85}	2^{159}	2^{151}	2^{160}
2^{128}	$(15, 60, 88)$	2^{176} $(g = 3)$	2^{131}	2^{258}	2^{249}	2^{259}

is most worthy of optimization. Therefore we aim for a comparison with equal conditions for all schemes such as the same platform and implementation by the same person, also with nearly the same possible optimizations. Additionally practical figures are given in a real world scenario for signature verification and generation time. All the schemes were implemented with runtime optimization in mind.

4.1 Target Platform and Tools

An ATxMega128a1 on an xplain board was used as target device. This micro processor has a clock frequency of 32 MHz, 128KB flash program memory and 8KB SRAM. The code was written in C and optimized for embedded use. As compiler avr-gcc in version 4.5.1 and at some places assembler gcc-as 2.20.1 was used.

Polynomial Representation / Key Storage. When implementing MQPKS on microprocessors it is important to construct an efficient way of storing and reading the keys out of memory. All polynomials of an $\mathcal{MQ}$-scheme are represented by their coefficients. It is important to decide how this coefficients are processed during runtime. The coefficients of UOV and Rainbow can be easily mapped to some readout loops. This is not that easy with enTTS as only a minimal count of coefficients are used and this few coefficients are spread over three layers and six different cyclic structures. As random access on the flash memory produces a lot of addressing overhead while calculating the address each time a serial approach was chosen. All coefficients are stored in memory in the same exact order in which they are read out. There are no gaps or zeros in memory which is also memory efficient. This memory architecture allows us to read out the keys directly and simply increment the address to reach the next coefficient. The AVR instruction set allows a memory readout with a post increment in one clock cycles from SRAM or two clock cycles from Flash memory. Therefore no additional address calculation is needed. The number of coefficients to store and thus the memory consumptions in bytes is $o\left(ov + \frac{v(v+1)}{2}\right)$ for UOV, $o_1\left(o_1 v + \frac{v(v+1)}{2}\right) + o_2\left(o_2(v + o_1) + \frac{(v+o_1)(v+o_1+1)}{2}\right)$ for Rainbow and $8l^2 - 6l - 3$ for enTTS. The resulting memory requirements for specific security parameters are given in Table 5.

4.2 Arithmetic and Field

As the used microprocessor is based on an 8 bit architecture, working in $\mathbb{F}_{2^8}$ is optimal. Multiplication is done by a table look up, each element is brought to its exponential representation, processed and then transformed back to the normal polynomial representation. Every transformation from the exponential to the basis representation costs one memory access, therefore in all implementations the exponential representation is kept as long as no $\mathbb{F}_{2^8}$ addition takes place, which is a bitwise exclusive OR operation of two coefficients in the basis representation. As the coefficients of the keys are first read in by a multiplication, all keys are already stored in the exponential form. Random numbers are generated by the rand() gcc pseudo random number generator. This function is seeded with a value derived from uninitialized SRAM blocks which are arbitrary on every start up.

Inverting the Layers. All schemes require the inversion of multivariate systems of equations. As only linear systems of equation can be solved efficiently, we have to fix variables until the system gets linear and then perform a simple Gaussian elimination using LU decomposition. Here the exponential representation is also used where possible. For example the lower matrix and all variables were saved in exponential form. In enTTS the middle layer consists only of polynomials depending on already known variables. Therefore these polynomials can be inverted directly.

4.3 Key Size and Signature Runtime Reduction

The main problem of $\mathcal{MQ}$-schemes are large keys, as storage space is limited on embedded devices. Large private keys come also together with long signature time, due to the processing of more data. As the signature for a fixed message is not unique, there is a lot of redundancy that can be used to reduce the secret key S (cf. theory of equivalent keys). We used such minimal keys for UOV as well as for Rainbow. Note that there are no equivalent keys known for enTTS and thus the whole matrix S has to be stored. The special form of S has two additional side effects in addition to less space. First, also the signature time is reduced. The multiplication with the identity matrix corresponds to a copy of the signature so that only the multiplication with the remaining coefficients has to be done. For UOV this saves us $\frac{(v-1)\cdot v}{2} + \frac{(o-1)\cdot o}{2}$ equations and for Rainbow $\frac{(v-1)\cdot v}{2} + \frac{(o_1-1)\cdot o_1}{2} + \frac{(o_2-1)\cdot o_2}{2}$. The second observation is that due to the identity matrix in the vinegar $\times$ vinegar part, large parts of $\mathcal{P}$ and $\mathcal{F}$ are equal. They do not increase security an can be seen as a system parameter (cf. [19]). As required by the authors of [19] for 0/1-UOV, also a different monomial ordering was chosen according to a minimal Turán graph. This reordering prevent easier attacks on the public key. The same procedure is probably possible for Rainbow. But as no publication exists which investigated this case, it was not implemented. For enTTS this is not possible as the Tame equations in the middle layer cause to blur the variable structure and no equivalent keys are known.

4.4 Verify Runtime Reduction

In the case of 0/1-UOV, choosing the coefficient from $\mathbb{F}_2$ has another advantage besides of less memory consumption. The verification and signature generation time can be reduced. As we know that the majority of coefficients are from $\mathbb{F}_2$, we can check for a one or a zero, which leads to a copy instruction in the case of one or a skip instruction in case of zero. Only otherwise we have to perform a costly multiplication in $\mathbb{F}_{2^8}$. The effect is in our implementation not marginally visible, because the used table look up method is fast compared to a schoolbook multiplication method.

4.5 RAM Requirements

$\mathcal{MQ}$-schemes do not need a lot of RAM, in contrast to the persistent flash memory requirements. In Table 4 the requirements are listed. Besides RAM needed for persistent, counting or temporary variables, only the Gaussian elimination algorithm needs a noticeable amount of RAM. As the inversion is computed in place, only one quadratic systems at time has to be stored in RAM. In case of multiple layers the maximal requirements are defined by the largest system of equations to be solved.

Table 4. Minimal Ram Requirements for LES Solving in Bytes

security	2^{64}	2^{80}	2^{128}	general
UOV	441	784	1936	m^2
Rainbow	400	729	1849	$(o_1 + o_2)^2$
enTTS	169	289	841	$(2l - 1)^2$

4.6 Key Generation

The keys for all schemes are generated on a standard PC using a C program. Basically $T \circ \mathcal{F} \circ S = P$ has to be computed. Using the quadratic form, the composition can be written as in (1). An overview of the key generation process of 0/1 UOV with small parameters can be found in the appendix.

$$\mathfrak{P}^{(i)} = \sum_{j=1}^{m} t_{ij} S^{\mathsf{T}} \mathfrak{F}^{(j)} S \tag{1}$$

Another way to generate an UOV key is described in [19]. It can be done by transforming the matrix S into a matrix A_{uov} and write all coefficients of $f^{(i)}$ ordered lexicographically to the rows of Q. Then the following equation holds: $A_{uov} \cdot Q = S^T \mathfrak{F}^{(i)} S$. With this relation inverting A_{uov} is possible and therefore a inverse approach, choosing first $\mathcal{P}$ and then applying A_{uov} to get $\mathcal{F}$. For the runtime optimization the reordering of monomials can take place in A_{uov} instead of reorder the monomials in $\mathcal{P}$ and $\mathcal{F}$.

5 Results

Table 5 shows our achieved results. They are easy to compare because schemes are grouped by security level. For all schemes key size, runtime and code size are given. Where applicable the system parameter size is also included. The public and secret key sizes can be easily calculated. One element responds to one byte and no other overhead needs to be saved so the keys consists only of the coefficients of the public or secret maps and the linear transformations. In the case of 0/1-UOV a large part is fixed and declared as a system parameter, but it must be anyway saved or be easy to generate in a real world scenario, therefore thus size is also listed.

Clock cycles were count internally with two concatenated 16 bit counters which are enabled to count on every clock cycle. As the count of verify operations scales with $(\frac{n\cdot(n+1)}{2} \cdot m)$ the measured times do not surprise. As enTTS uses the largest numbers of n and m it has the lowest verify performance and the largest public keys. Rainbow is the fastest as the parameters can be chosen relatively low. The big advantage of enTTS is the small private key. Large parts of the central map are zero and have not to be saved. In terms of theoretical public key size 0/1-UOV performs the best. If the possibility to generate the system parameter on the device would exist, it would ensure the smallest public key. The gain of verification and signature time in comparison to the standard UOV is only minimal as the multiplication by table look up has no significant runtime difference in comparison to a multiplication with 0 or 1 as the 0 case is a special case and is checked anyway every time in a normal multiplication in $\mathbb{F}_{2^8}$. When measuring scalability for secret/public key size at the step from 2^{64} to 2^{128}, UOV has a increase factor of 9/9, 0/1-UOV of 9/9, Rainbow 10/11 and enTTS of 4/10. UOV scales the best in public key size, enTTS the best in private key size. Regarding the signature size, UOV has the highest expansion factor, with a message to signature ratio of approximately 2.3, followed by Rainbow with 1.7 and enTTS with 1.4.

As a comparison of an µC with an ASIC or PC implementation is meaningless, the only MQ implementation we can compare with is the one from [25]. The authors implemented enTTS$(5, 20, 28)$ on a MSP430 running at 8 MHz. Signing requires 17.75 ms and verifying 181.5 ms, when scaled up to our clock frequency. Although, the MSP430 is a 16 bit CPU, our implementation is a factor of 3.7 faster in signing and 5.1 times faster in verifying.

Also when comparing our work with implementations of the classical signature schemes RSA and ECDSA, all four schemes perform well. E.g. for 2^{80} bit security [13] reports 203ms for a ECC sign operation, where our implementations are two to ten times faster. For the verifying operation our work is up to three times faster. Due to the short exponent in RSA-verify, [13] verifies in the same order of magnitude. But the RSA-sign operation is at least a factor of 25 slower than our work. Table 6 summarizes other implementations on comparable 8 bit platforms.

Table 5. Results

	Scheme	n	m	Key Size [Byte] private	public	System Parameter private	public	Clockcyles x 1000 sign	verify	Time[ms]@32MHz sign	verify	Code Size [Byte] sign	verify
	enTTS(5, 20, 28)	28	20	1351	8120	*	*	153	1,126	4.79	35.22	12890	827
	enTTS(5, 20, 28)[25]	28	20	1417	8680	*	*	568[1]	5,808[1]	17.75[2]	181.5[2]	-	-
2^{64}	uov(21, 28)	49	21	21462	25725	*	*	1,615	1,690	50.49	52.83	2188	466
	0/1 uov(21, 28)	49	21	12936	4851	8526	20874	1,577	1,395	49.29	43.60	2258	578
	rainbow(15, 10, 10)	35	20	9250	12600	*	*	848	1,010	26.51	31.58	4162	466
	enTTS(7, 28, 40)	40	28	2731	22960	*	*	332	2,558	10.37	79.95	24898	827
2^{80}	uov(28, 37)	65	28	49728	60060	*	*	3,637	3,911	113.66	122.23	2188	466
	0/1 uov(28, 37)	65	28	30044	11368	19684	48692	3,526	3,211	110.20	100.37	2258	578
	rainbow(18, 13, 14)	45	27	19682	27945	*	*	1,740	2,214	54.38	69.19	4162	466
	enTTS(9, 36, 52)	52	36	4591	49608	*	*	609	6,658	19.03	208.07	41232	827
2^{128}	uov(44, 59)	103	44	194700	235664	*	*	13,314	14,134	416.07	441.70	2188	466
	0/1 uov(44, 59)	103	44	116820	43560	77880	192104	12,782	13,569	399.43	424.04	2258	578
	rainbow(36, 21, 22)	79	43	97675	135880	*	*	8,227	9,216	257.11	288.01	4162	466
	enTTS(15, 60, 88)	88	60	13051	234960	*	*	2,142	3,0789	66.94	962.17	116698	827

[*] Not applicable
[1] Derived from values in original work
[2] Scaled to the same clock frequency

Table 6. Overview of other implemenatations on comparable platforms

Method	Time[ms]@32MHz	
	sign	verify
enTTS$(5, 20, 28)$[25]	17.75[1]	181.5[1]
ECC-P160 (SECG) [13]	203[1]	203[1]
ECC-P192 (SECG) [13]	310[1]	310[1]
ECC-P224 (SECG) [13]	548[1]	548[1]
RSA-1024 [13]	2,748[1]	108[1]
RSA-2048 [13]	20,815[1]	485[1]
NTRU-251-127-31 sign [10]	143[1]	-

[1] For a fair comparison with our implementation running at 32MHz, timings at lower frequencies were scaled accordingly.

6 Conclusion

In this work we present the first µC implementations of the three most common MQPKS since nearly 10 years. Additionally, we implemented for the first time 0/1-UOV on a constrained device. All recent attacks were summarized and we proposed current security parameters for 2^{64}, 2^{80} and 2^{128} bit symmetric security. Additionally, we showed that choosing $v = 2o$ for UOV is outdated. When comparing with existing MQ implementations, ours are a factor of three and five times faster in signing and verifying, respectively. We hope our implementations will inspire follow up work, to improve acceptance of MQPKS in constrained environments.

6.1 Further Improvements

There is still space for improvements and the upper limit is not reached yet. A few ideas were not implemented in this work. Saving the system parameters is not optimal. Here a replacement by a pseudo random number generator or an other generator function would reduce the public key drastically, even if verification time would be increased. In our implementation all elements of $\mathbb{F}_2$ are saved as a byte value. It would be possible to achieve smaller keys when saving 8 elements in one byte, combined with a verification function which utilizes assembler instructions maybe even a faster verification could be possible. An overall time vs. code size trade-off is still a topic to investigate. $\mathcal{MQ}$-schemes are very well scalable in regard to this trade-off.

References

1. Yang, B.-Y., Chen, J.-M., Chen, Y.-H.: TTS: High-Speed Signatures on a Low-Cost Smart Card. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 371–385. Springer, Heidelberg (2004)
2. Balasubramanian, S., Carter, H., Bogdanov, A., Rupp, A., Ding, J.: Fast Multivariate Signature Generation in Hardware: The Case of Rainbow. In: International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2008, pp. 25–30 (July 2008)
3. Bettale, L., Faugère, J.-C., Perret, L.: Hybrid Approach for Solving Multivariate Systems over Finite Fields. Journal of Mathematical Cryptology 3(3), 177–197 (2009)
4. Bouillaguet, C., Fouque, P.-A., Macario-Rat, G.: Practical key-recovery for all possible parameters of SFLASH. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 667–685. Springer, Heidelberg (2011)
5. Braeken, A., Wolf, C., Preneel, B.: A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 29–43. Springer, Heidelberg (2005), http://eprint.iacr.org/2004/222/
6. Buchmann, J., Ding, J. (eds.): PQCrypto 2008. LNCS, vol. 5299. Springer, Heidelberg (2008)
7. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
8. Courtois, N.T., Daum, M., Felke, P.: On the Security of HFE, HFEv- and Quartz. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 337–350. Springer, Heidelberg (2002)
9. Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
10. Driessen, B., Poschmann, A., Paar, C.: Comparison of Innovative Signature Algorithms for WSNs. In: Proceedings of ACM WiSec 2008. ACM (2008)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability — A Guide to the Theory of NP-Completeness. W.H. Freeman and Company (1979) ISBN 0-7167-1044-7 or 0-7167-1045-5
12. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM Cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
13. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs, pp. 119–132 (2004)
14. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
15. Kipnis, A., Shamir, A.: Cryptanalysis of the Oil and Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
16. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
17. Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)

18. Petzoldt, A., Bulygin, S., Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 218–240. Springer, Heidelberg (2010)
19. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small Public Keys and Fast Verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Systems. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 475–490. Springer, Heidelberg (2011)
20. Shor, P.W.: Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. on Computing, 1484–1509 (1997)
21. Tang, S., Yi, H., Ding, J., Chen, H., Chen, G.: High-Speed Hardware Implementation of Rainbow Signature on FPGAs. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 228–243. Springer, Heidelberg (2011)
22. Thomae, E., Wolf, C.: Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 156–171. Springer, Heidelberg (2012)
23. Wolf, C., Braeken, A., Preneel, B.: Efficient Cryptanalysis of RSE(2)PKC and RSSE(2)PKC (2004)
24. Yang, B.-Y., Chen, J.-M.: Building Secure Tame-like Multivariate Public-Key Cryptosystems: The new TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)
25. Yang, B.-Y., Cheng, C.-M., Chen, B.-R., Chen, J.-M.: Implementing Minimized Multivariate PKC on Low-Resource Embedded Systems. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 73–88. Springer, Heidelberg (2006)

A Toy Example of 0/1 UOV Key Generation

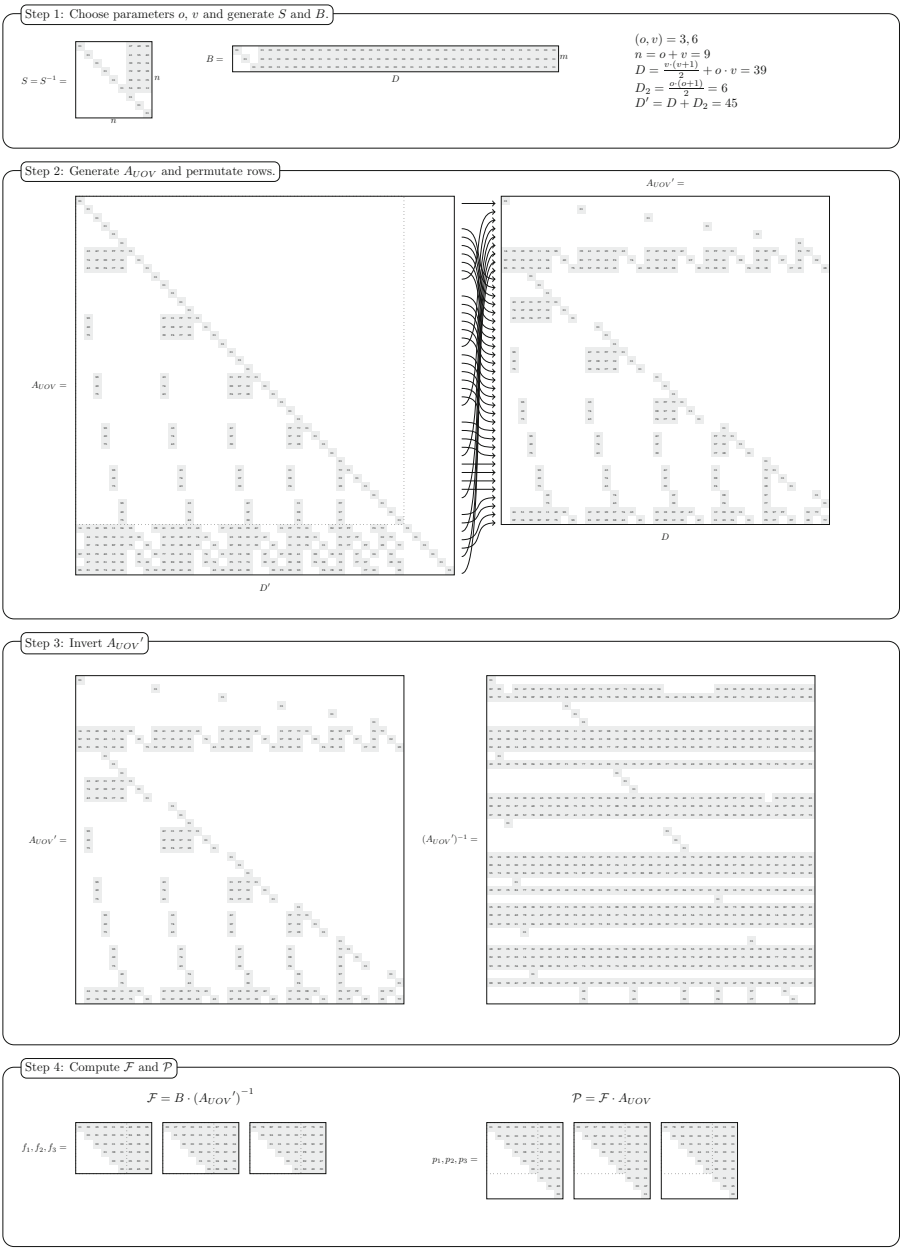

Fig. 5. 0/1 UOV Key Generation. For details see [19].