Implementing Minimized Multivariate PKC on Low-Resource Embedded Systems

Bo-Yin Yang[1,⋆], Chen-Mou Cheng[2], Bor-Rong Chen[2], and Jiun-Ming Chen[3]

[1] Dept. of Math., Tamkang University, Tamsui, Taiwan
by@moscito.org
[2] Harvard University
{doug, brchen}@eecs.harvard.edu
[3] Chinese Data Security, Inc., and Nat'l Taiwan U., Taipei
jmchen@math.ntu.edu.tw

Abstract. Multivariate (or $\mathcal{MQ}$) public-key cryptosystems (PKC) are alternatives to traditional PKCs based on large algebraic structures (e.g., RSA and ECC); they usually execute much faster than traditional PKCs on the same hardware. However, one major challenge in implementing multivariates in embedded systems is that the key size can be prohibitively large for applications with stringent resource constraints such as low-cost smart cards, sensor networks (e.g., Berkeley motes), and radio-frequency identification (RFID). In this paper, we investigate strategies for shortening the key of a multivariate PKC. We apply these strategies to the Tame Transformation Signatures (TTS) as an example and quantify the improvement in key size and running speed, both theoretically and via implementation. We also investigate ways to save die space and energy consumption in hardware, reporting on our ASIC implementation of TTS on a TSMC 0.25μm process. Even without any key shortening, the current consumption of TTS is only 21 μA for computing a signature, using 22,000 gate equivalents and 16,000 100-kHz cycles (160 ms). With circulant-matrix key shortening, the numbers go down to 17,000 gates and 4,400 cycles (44 ms). We therefore conclude: besides representing a future-proofing investment against the emerging quantum computers, multivariates can be immediately useful in niches.

Keywords: Multivariate public-key cryptosystem, efficient implementation, digital signature schemes, embedded system, sensor networks, motes.

1 Introduction

A multivariate public-key cryptosystem (a multivariate PKC, an $\mathcal{MQ}$-scheme, or simply a multivariate) uses as the public key the $mn(n+3)/2$ coefficients of a quadratic polynomial map $V = (\ell_1, \ldots, \ell_m) : K^n \to K^m$ with no constant term, where $K = \mathrm{GF}(q)$ is a finite field, called the "base field". Computing $\mathbf{w} = V^{-1}(\mathbf{z}) = (w_1, \ldots, w_n) \in K^n$ is considered a hard problem, generically NP-hard [16].

⋆ Partially sponsored by Taiwan's Nat'l Science Council grant NSC-94-2115-M-032-010 and Taiwan Information Security Center (TWISC) at Nat'l Taiwan U. of Sci&Tech, Taipei, Taiwan.

J.A. Clark et al. (Eds.): SPC 2006, LNCS 3934, pp. 73–88, 2006.

In practice, the trapdoor needed for a PKC is almost always accomplished by using a public map composed of three maps as in $V : \mathbf{w} \in K^n \overset{\phi_1}{\mapsto} \mathbf{x} \overset{\phi_2}{\mapsto} \mathbf{y} \overset{\phi_3}{\mapsto} \mathbf{z} \in K^m$. The *central map* ϕ_2 is quadratic. $\phi_1 : \mathbf{w} \mapsto \mathsf{M}_1\mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathsf{M}_3\mathbf{y} + \mathbf{c}_3$ are affine, usually invertible. The private key is the $m(m+1) + n(n+1)$ coefficients in $(\mathsf{M}_1, \mathsf{M}_3, \mathbf{c}_1, \mathbf{c}_3)$ plus whatever necessary to compute ϕ_2^{-1}.

The security of a multivariate PKC thus depends on the infeasibility of decomposing maps and that of solving a large system of polynomial equations. The reader is referred to [41] for a comprehensive reference for multivariate PKCs, including the nomenclature and the state of the art.

1.1 Advantages That Accrue to Multivariates

When quantum computers (QCs) with thousands of qubits arrive, RSA and discrete-log schemes will be broken [40]. $\mathcal{MQ}$- and lattice-type schemes stand with halved log-complexity ([22], e.g., $2^{200} \rightarrow 2^{100}$). We also have quantum key exchange but not quantum signatures. So $\mathcal{MQ}$-schemes seem like future-proofing insurance.

Another reason is that the private map of RSA is intrinsically slow. As a result, transaction speed on commodity hardware suffers, hindering wider deployment. This slowness is magnified on embedded systems [1, 44], where lack of megahertz hurts multivariates less because these schemes spend most of their time in lookup tables.

1.2 Challenges for Multivariates and Our Contributions

Currently, multivariate schemes face some major challenges:

Novelty: This leads to distrust and a prolonged lack of interest. Consequently, there is little in the way of provable security results or optimizations as for RSA or ECC.

Key sizes: Multivariates have a large public key and sometimes a large private key as well. This necessitates costlier hardware just like the slow private map of RSA.

We will herein discuss aspects of low-resource implementations. We will illustrate with the signature scheme TTS (Sec. 2), *but the following apply to most $\mathcal{MQ}$-schemes if one wishes to cut down the private key:*

Key scheduling lets a smart card carry a small "mini key" (Sec. 3), based on which the "real key" and the private map is built on the fly.

Structure in the linear maps enables representation of M_1 and M_3 with fewer parameters. We can have a private key that is around 1/5 of its original length and use 40% less running time (Sec. 4).

Despite the fact that most multivariate schemes are already fairly fast, and that the public keys are much longer (which makes them the natural place to start if one wishes to cut down memory usage), we feel that our effort is justified in view of the following:

- Often a scheme needs to be as fast as possible, not just "fast enough".
- In many applications, we only need to store the private key on-card; for example, when the embedded devices are used as a cryptographic token.

- Some applications can tolerate lower security but absolutely must fit within very tight resource limits. RSA simply won't fit on a low-end RFID. But a multivariate PKC might if its private key can be shortened. Schemes operating on small chunks are more likely to be implemented in such an environment [15].
- The central map of an $\mathcal{MQ}$-scheme often determines its properties, including weaknesses. *A chain can only be as strong as its weakest link.* So if a given central map leads to a cryptanalysis in, say, 2^{100}, then the 100+ parameters in the matrices are more likely to be an overkill rather than a security enhancement.

The rest of this paper is organized as follows. In Sec. 2, we go over the Tame Transformation Signatures (TTS), a workbench for key-shortening experimentation. Subsequently in Sec. 3, we discuss the aspects of key scheduling, and in Sec. 4, we describe the key-shortening strategy via restructuring the linear maps. In Sec. 5, we report the result of our application-specific integrated circuit (ASIC) implementation, mainly targeting at RFID applications. We conclude with the discussion in Sec. 6.

Some of the diversed results referenced are summarized in the appendix; the rest needs to be looked up from the original sources [8, 10, 11, 13, 27, 37, 41, 43, 44].

2 Tame Transformation Signatures

Before we move on, we need to emphasize that there is no reductionist proof of security today; the security of $\mathcal{MQ}$-schemes is so far thrust-and-parry. Since this is mainly about implementation, we leave out most details about security, although we check that known attacks are not functional against our schemes. The interested reader is referred to Appendix A, where we report the result of a few basic cryptographical experiments.

2.1 Classification of Multivariates, and enTTS (20,28)

In designing and implementing key-shortening techniques for multivariates, we note that there is a rationale to experimenting with variants of existing ones instead of inventing new ones. Extant schemes represent accumulated experience and history. In tweaking them appropriately, one can expect to inherit many of the good properties and reuse code, which introduces fewer opportunities for mistakes.

There are four basic ways [41] one can set up the trapdoor of a multivariate scheme for ϕ_2 to be inverted: Imai-Matsumoto's C^* [31] (and derivatives SFLASH [39] and PMI+ [10]), Hidden Field Equations [37], "Unbalanced Oil-and-Vinegar" [27], and "Stepwise Triangular System" [43]. Basic trapdoors must be modified for security — see [41] for a summary of modifications. All have $O(n^3)$-time (and space) public maps in the practical range, where n is the size of the digest or plaintext block.

C^* and HFE are "dual-field" multivariates, which operate over a larger field $K^k \equiv \mathrm{GF}(q^k)$. UOV and STS are "single-field" multivariates. The TTS [43] family of signature schemes use a UOV-STS combination. We will illustrate most of our techniques on a current variant in the family called the Enhanced TTS with $K = \mathrm{GF}(2^8)$, 20-byte hashes, and 28-byte signatures, also known as enTTS (20,28). This has the central map

$\phi_2 : \mathbf{x} = (x_0, x_1, \ldots, x_{27}) \mapsto \mathbf{y} = (y_8, y_9, \ldots, y_{27})$:

$$
\begin{aligned}
y_i &= x_i + \textstyle\sum_{j=1}^{7} p_{ij} x_j x_{8+(i+j \bmod 9)},\ i = 8 \cdots 16; \\
y_{17} &= x_{17} + p_{17,1} x_1 x_6 + p_{17,2} x_2 x_5 + p_{17,3} x_3 x_4 \\
&\quad + p_{17,4} x_9 x_{16} + p_{17,5} x_{10} x_{15} + p_{17,6} x_{11} x_{14} + p_{17,7} x_{12} x_{13}; \\
y_{18} &= x_{18} + p_{18,1} x_2 x_7 + p_{18,2} x_3 x_6 + p_{18,3} x_4 x_5 \\
&\quad + p_{18,4} x_{10} x_{17} + p_{18,5} x_{11} x_{16} + p_{18,6} x_{12} x_{15} + p_{18,7} x_{13} x_{14}; \\
y_i &= x_i + p_{i,0} x_{i-11} x_{i-9} + \textstyle\sum_{j=19}^{i-1} p_{i,j-18}\ x_{2(i-j)-(i \bmod 2)}\ x_j + p_{i,i-18} x_0 x_i \\
&\quad + \textstyle\sum_{j=i+1}^{27} p_{i,j-18}\ x_{i-j+19}\ x_j,\ i = 19 \cdots 27.
\end{aligned}
$$

Given message M, we compute a 160-bit secure hash digest vector $\mathbf{z} = H(M)$ from the message. We may now compute $\mathbf{y} = \mathsf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$, then $\mathbf{x} \in \phi_2^{-1}(\mathbf{y})$ as follows:

1. Assign random $x_1, \ldots, x_7$ and try to solve the first 9 equations for x_8 to x_{16}.
2. Solve serially for x_{17} and x_{18} using the next two equations (y_{17} and y_{18}).
3. Assign a random x_0 and try to solve the last 9 equations for x_{19} through x_{27}.

We find the signature $\mathbf{w} = \mathsf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$. Release $(M, \mathbf{w})$. At most 9 values of x_7 and x_0 make the two systems unsolvable, so we will find a solution.

enTTS (20,28) has 8680- and 1417-byte public and private keys. A smaller instance enTTS (16,22) has this central map, along with 4400- and 879-byte public and private keys [43]:

$$
\begin{aligned}
y_i &= x_i + \textstyle\sum_{j=1}^{6} p_{ij} x_j x_{6+(i+j+1 \bmod 7)},\ i = 6 \cdots 12; \\
y_{13} &= x_{13} + p_{13,1} x_1 x_4 + p_{13,2} x_2 x_3 \\
&\quad + p_{13,3} x_7 x_{12} + p_{13,4} x_8 x_{11} + p_{13,5} x_9 x_{10}; \\
y_{14} &= x_{14} + p_{14,1} x_2 x_5 + p_{14,2} x_3 x_4 \\
&\quad + p_{14,3} x_8 x_{13} + p_{14,4} x_9 x_{12} + p_{14,5} x_{10} x_{11}; \\
y_i &= x_i + p_{i,0} x_{i-7} x_{i-9} + \textstyle\sum_{j=15}^{i-1} p_{i,j-14}\ x_{2(i-j)-(i \bmod 2)}\ x_j + p_{i,i-14} x_0 x_i \\
&\quad + \textstyle\sum_{j=i+1}^{21} p_{i,j-14}\ x_{i-j+15}\ x_j,\ i = 15 \cdots 21.
\end{aligned}
$$

This was built to compare to RSA-768, while enTTS (20,28) was to RSA-1024.

2.2 History and Security of enTTS and Other TTS Schemes

The STS-UOV family of $\mathcal{MQ}$-signatures contains in addition the Rainbow and TRMS schemes [2, 11]. The combination of recursive segmented evaluation and solving consecutive linear systems defend against the three well-known attacks based on linear algebra against STS [21] and UOV [27]. TRMS uses intermediate-field arithmetic, which also in effect solves a linear system. Both TRMS and enTTS are *sparse* variants, i.e., a TTS-type scheme, with fast signatures and key generation on both PCs and smart cards. The schemes in [44] slightly differ from enTTS due to the UOV-based attacks carried out by Ding and Yin [12], while high speed on a smart card is still retained.

The enTTS (20,28) scheme was designed to account for all known attacks [43], including linear algebra attacks mentioned above, algebraic attacks based on Faugère's

$\mathbb{F}_5$ and XL of Courtois *et al* [8, 13], improved search methods [4], and some methods tailored to specific schemes [19, 20, 36]. We repeated experiments checking that no vulnerabilities had been reintroduced in our short-key versions.

2.3 Performance on State-of-the-Art Sensor Nodes

To evaluate the performance of enTTS on state-of-the-art sensor nodes, we benchmark enTTS on one of the most popular wireless sensor network devices—the Tmote Sky mote. Tmote Sky is equipped with an 8 MHz Texas Instrument MSP430 microcontroller and a Chipcon CC2420 2.4 GHz radio that supports IEEE 802.15.4 wireless low-power medium access control standard. Running on top of it is the TinyOS sensor network operating system [23]. TinyOS is a modular event-driven operating system designed specifically for sensor network applications. It has a component-based programming model, supported by the nesC language [18]. Applications are written as modules that are composed together through a set of predefined interfaces, which describe the events to be handled and the implemented commands. The composition of modules is achieved in compile time by the nesC compiler, whose output is a single C source program, subsequently to be compiled by a regular C compiler into the MSP430 binary code. It is then downloaded onto the sensor devices to run. Such TinyOS-based wireless sensor systems are under active study and development by the sensor network community; they are also becoming a candidate for building pervasive computing infrastructures.

Security and privacy issues, such as authentication and data integrity, are important in some of these emerging applications, e.g., patients vital signals monitoring and dissemination in a hospital. Unfortunately, the more traditional PKCs have not been able to apply in these applications, due largely to the limited computational resources available on the sensor nodes. This is where $\mathcal{MQ}$ schemes like enTTS can find immediate applications.

A typical sensor node like the Tmote Sky mote has a small working RAM (10 KBytes in this case), a slightly larger read-only program memory (48 KBytes), and a relatively large flash memory (1 MBytes) for storing collected raw data and other auxiliary information for its operations. In our case, the private keys are small and can be fit into the working RAM. This helps achieving high signature performance, which is important because it is usually these computationally challenged sensors that are signing the raw data to protect data integrity. For verification, we need to store the public key in flash memory because they are too big to fit into RAM. We argue that this is tolerable because verification is mostly done in the relatively more powerful base stations, although it is needed occasionally by the motes, say, when they need to verify the authenticity of a message coming from a base station or another mote.

We benchmark both the verification and the signature performance of enTTS (20,28): The average signing time is 71 ms, while the average verification time is 726 ms[4], measured with the time provided by TinyOS at the granularity of 1/32,768 seconds, averaging over 1,000 runs. We note that the throughput of flash memory reads on our Tmote Sky mote is merely around 45 KBytes per second, so we spend a

[4] These numbers do not include the time in computing message digests.

significant portion of the verification time reading the 8,680-byte public key off the flash memory. We believe that the verification time can be significantly improved if the entire public key can be stored in a larger working RAM, which is expected to be in the future releases of the device. Even with such an impediment, the results are quite promising compared with previous results obtained using more traditional PKCs such as ECC [30].

3 From Previous Attempts at Key Shortening to Scheduling

Many attempts to use subfields for smaller keys in multivariates have been made. The original version of SFLASH used a subfield but was broken [19]. In general, using subfields to cut down public key size is not a very good idea. The reason is that, in many of the attacks against multivariates such as those in [21, 27, 38], guessing at variables is needed; thus the security is highly correlated to the size of the field in which the matrices elements of M_1 and M_3 are taken.

However, using elements of subfields for parameters in the central map *can* cut down private key size as will be in use in an implementation below. Further, subfield constructions for hardware (especially ASIC) are common. Especially notable are the layered designs of C. Paar [34, 35], which we borrowed into our designs here.

3.1 Key Scheduling and Its Practical Considerations

In general, key scheduling (generating sub-keys or round keys from an original key) will not shorten the running time. In fact, it usually makes everything slower. But it is considered worthwhile because short keys have advantages. In presenting SFLASH^{v2} [39], the authors mentioned that it is possible to run SFLASH with a key schedule. We try to affirm this idea with an actual run and to provide a control.

We have a few obvious candidates. AES submissions have well-tested key schedules with assembly-language routines available to call on the PC and the 8051, so we can use the Rijndael key set-up routine. Or we can also use a stream cipher like RC4 (with a variable key size), or a fast linear feed-back pseudo random number generator (PRNG) such as TT800 (a precursor to the mersenne twister, up to 100-byte keys, cf. [32]).

It seems natural to treat each routine as a PRNG or stream cipher. Two observations:

- We store sub-keys (random numbers) in a buffer and take inputs as needed.
- If we wish to output the public key on demand, we must be able to evaluate ϕ_i and ϕ_i^{-1} in any order. This is very difficult unless the state involved is very short, since we may have to keep six or seven sets of state for each stage.

A trick from [44] seems applicable: Factor both M_1 and M_3 as $L_i D_i U_i$ where L and U are lower- and upper-triangular, respectively, with 1's on the diagonal, and D is invertible and diagonal. Order the bytes L_{21}, $L_{31}, \ldots, L_{n1}$, $L_{32}, \ldots, L_{n,n-1}$ (column order), then D^{-1} (diagonal only), then $U_{n-1,n}$, $U_{n-2,n}, \ldots, U_{1,n}$, $U_{n-2,n-1}, \ldots, U_{1,2}$ (U, in reversed column order). We can thus compute a product by M_1^{-1} and M_3^{-1}. Store $\mathbf{c}_1$ ahead of the M_1, and compute and tuck $\mathbf{c}_3$ away somewhere.

3.2 Testing Results with Key Scheduling

We justify, after a fashion, the claim that key scheduling [1, 39] is feasible for multivariates. We are able to fit in the keys and the program of an SFLASH scheme in 8kB ROM for a 8051-compatible micro-controller (this test was run on simulator of a development kit only) and a TTS scheme in 4kB. We can not claim our 8051 programs as well optimized, but here are the test results for TTS:

The "private key" lengths listed are really the seed maintained by the key scheduling routine. We are unable to test key generation on card with TT800 and RC4 because we somehow ran out of resources each time. Using the key setup routine for AES, the key generation time is long but barely doable. The conclusion is that *perhaps we don't need on-card key generation for such low-cost concerns*, even though there appears to be no security concerns in so doing, which can be seen from some basic testing on the public keys generated by the key-scheduled methods (Appendix A) showing "no difference."

Table 1. Key-scheduled signatures schemes on a stock 3.57MHz Intel 8032AH card

Scheme	Signature	Pr. Key	Pub.Key	setup ROM	setup	sign	sign ROM
TTS (20,28)	224 bits	1.4 kB	8.6 kB	4.2 kB	62 sec	198 ms	1.4 kB
Rijndael	224 bits	120 B	8.6 kB	6.4 kB	703 sec	454 ms	3.6 kB
RC4	224 bits	32 B	8.6 kB	N/A	N/A	334 ms	2.2 kB
TT800	224 bits	32 B	8.6 kB	N/A	N/A	310 ms	2.5 kB

4 Restructure for Faster Signing and Tiny Private Keys

Decomposing a matrix is not new. There are $[\mathbf{n}]_q! = (q^n - 1)\ (q^n - q)\ \cdots (q^n - q^{n-1})$ invertible $n \times n$ matrices over $\mathrm{GF}(q)$. But for generating an invertible matrix, it is common to use LU decomposition, which produces only $(q-1)^n q^{n(n-1)}$ of the non-singular matrices. To decrease storage needs, we must decompose differently here. Trying not to repeat history, we examine two such earlier attempts in Sec. 4.1.

Here we must mention the two recent other papers dealing with equivalency and normal forms of multivariates, [25] and [42]. As mentioned by [41], sparse variants like TTS are not affected much by such equivalency concerns.

4.1 Earlier Attempts

Two ideas heretofore seen in print tried replacing multiplication by a non-singular matrix by a sequence of $O(n)$ linear steps between the components of a vector.

$\mathbf{D}(\mathbf{J} - \mathbf{P})\mathbf{D}$: Both M_1 and M_3 are in form $L(J - P_\sigma)R$, where L and R are invertible and diagonal, J a matrix of all 1's, and P_σ a permutation matrix. Store L and R plus the σ's in private key.

Elementary Row Operations: [24] proposed to use both M_1 and M_3 in the form of $P_{\sigma^{-1}\rho^{-1}}\left(\prod_{i=i}^{n-1}(1+\beta_i E_{n-i+1,n-i})\right) P_\rho \left(\prod_{j=1}^{n-1}(1+\alpha_j E_{j,j+1})\right) P_\sigma D$, where n is the dimension, P_σ a permutation matrix with $\sigma = [\sigma_1, \sigma_2, \ldots]$ (list notation), D is invertible diagonal, and E_{ij} is all 0's except a 1 in the (i,j) position.

But both have security concerns. Let $\mathbf{x} = \mathsf{M}\mathbf{w}$ and $\mathsf{M} = (m_{k\ell})_{1\le k,\ell\le n}$, then

$$x_i x_j = \sum_{k=1}^{n} m_{ik}m_{jk}\ w_k^2 + \sum_{1\le k<\ell\le n} (m_{ik}m_{j\ell} + m_{i\ell}m_{jk})\ w_k w_\ell.$$

If either most of the m_{ij} are zero, or there is a tendency for $m_{ik} : m_{i\ell} = m_{jk} : m_{j\ell}$, then cross-term coefficients will tend to vanish. If M is constructed with elementary row operations as above, many entries will be zero. When both m_{ik} and m_{jk} are non-zero, the odds are good that a multiple of row i has been just added to row j or vice versa.

The matrix $L(J-P)R$ has far fewer zeroes, but most entries of $J - P_\sigma$ are 1's, hence $m_{k\ell}$ is usually $L_k R_\ell$. When neither σ_1 nor σ_2 is 1 or 2 (very likely), we have

$$x_1 = L_1(R_1 w_1 + R_2 w_2 + \cdots), \quad x_2 = L_2(R_1 w_1 + R_2 w_2 + \cdots),$$

and $w_1 w_2$ coefficient in $x_1 x_2$ is zero. More precisely, let $\mathsf{M}_1 = L(J-P_\sigma)R$, $\mathbf{x} = \mathsf{M}_1\mathbf{w}$. The (i,j) entry of M_1 is 0 if $j = \sigma_i$ and $L_i R_j$ otherwise. Thus only $2n-2$ of the $n(n-1)/2$ cross terms are non-zero in the cross term $x_i x_j$.

$$x_i x_j = L_i L_j \left(\sum_{k=1}^{n} R_k^2 w_k^2 + (R_{\sigma_i} w_{\sigma_i} + R_{\sigma_j} w_{\sigma_j}) \sum_{k=1}^{n} R_k w_k + R_{\sigma_i} w_{\sigma_i} R_{\sigma_j} w_{\sigma_j} \right) \quad (1)$$

It is verifiable that polynomials for $\mathbf{y}$ *in* $\mathbf{w}$ *are sparse* for cryptosystems with relatively few cross terms which leads to easier Gröbner bases computations [14].

4.2 Designing for Non-sparseness

An obvious improvement on the previous section is to use other forms of zero-one matrices that leads to fewer zeros coefficients in the quadratic polynomials.

Proposition 1. *Suppose* $\mathsf{M} = LSR$*, where* L *and* R *are diagonal and* $S = (s_{ij})$ *is a 0-1 matrix, and if* $s_{ij} = 1$ *with probability* p*, then the probability for the coefficient of* $w_k w_\ell$ *in the expansion of* $x_i x_j$ *(denoted* $[w_k w_\ell](x_i x_j)$*) to be non-zero is* $\approx 2p^2(1-p^2)$.

Proof. Let entries of L and R be (resp.) L_i and R_j, then $x_i = L_i\left(\sum_{j=1}^{n} R_i s_{ij} w_i\right) + c_i$. So when $k \neq \ell$, $[w_k w_\ell](x_i x_j) = L_i L_j R_k R_l (s_{ik}s_{j\ell} + s_{i\ell}s_{jk})$. This will vanish unless exactly one of $s_{ik}s_{j\ell}$ and $s_{i\ell}s_{jk}$ is 1, or rather if and only if either $s_{ik} = s_{j\ell} = 1$ (at least one of $s_{i\ell}$ and s_{jk} is zero); or $s_{i\ell} = s_{jk} = 1$ (at least one of s_{ik} and $s_{j\ell}$ is zero).

We can make a coefficient non-zero at most half the time when $p \approx 1/\sqrt{2}$. For example, when $n = 28$ we want nineteen entries in each row and column of S to be 1 and the other nine to be 0. Elementary row operations and $D(J-P)D$ decompositions were

considered partly because we want the length of the private key and the signing time to be $O(n)$, yet (as above) the number of zeros as well as the number of ones need to be $\propto n^2$. What we want is: *S should be in a form that makes $\mathbf{x} \mapsto \mathbf{w} = \mathsf{M}^{-1}\mathbf{x}$ doable in $O(n)$, even though M itself contains $\propto n^2$ of both zeros and ones.* This is where circulants come in.

4.3 Circulant Matrices

A square matrix $\mathsf{M} = (m_{ij})_{1\le i,j\le n}$ is *circulant* if $i - k \equiv j - \ell \pmod n$ implies $m_{ij} = m_{k\ell}$. Circulant matrices have been studied for a long time (e.g., the monograph [7]). The S-Box of AES effectively uses a multiplication by the circulants

$$S_8 = \begin{bmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{bmatrix} \quad S_8^{-1} = \begin{bmatrix} 0&1&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&0&0&1&0&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&1\\ 1&0&1&0&0&1&0&0 \end{bmatrix}.$$

Borrowing a page from the same book, we let $\mathsf{M}_1 = LP_\sigma SP_\rho R$, where $\sigma, \rho \in \mathcal{S}_m$ are permutations. We want an S that would let us multiply a vector by S^{-1} quickly. One obvious candidate is $S_{28} = (s_{ij})_{1\le i,j\le 28}$, where $s_{ij} = 1$ if $j-i$ is congruent to $0 \cdots 18$ and $= 0$ otherwise. Multiplying by S_{28} is easy, but multiplying by S_{28}^{-1} is even easier, because

$$S_{28}^{-1} = (a_{i,j})_{1\le i,j\le 28},\ a_{ij} = 1 \text{ if } i - j \equiv 0, 9, 18 \pmod{28}, \text{ else } a_{ij} = 0.$$

We would like to find an S_m for any m with similarly nice properties: $S_m^{-1}\mathbf{x}$ easily evaluated in $O(m)$, and S_m in some nice form that has the proportion of 1's as close to 0.707 as possible. Luckily, the same thing works for any $m = 3k + 1$. In fact we have

Proposition 2 (Trilinear Circulant Matrices). *In a field of characteristic 2:*

1. *Let $S_{3k+1} = (s_{ij})$, $s_{ij} = 1$ iff $j - i \equiv 0 \cdots 2k \pmod{3k+1}$ and $= 0$ otherwise, then $S_{3k+1}^{-1} = (a_{ij})$ is a 0-1 circulant with $a_{ij} = 1$ iff $i - j \equiv 0, k, 2k$;*
2. *Let $S_{3k+2} = (s_{ij})$, $s_{ij} = 1$ iff $j - i \equiv 0 \cdots 2k \pmod{3k+2}$ and $= 0$ otherwise, then $S_{3k+2}^{-1} = (a_{ij})$ is a 0-1 circulant with $a_{ij} = 1$ iff $i - j \equiv -1, k, 2k+1$.*

The proofs can be found in [45], the more complete version of this paper submitted as a research report to TWISC.

We see that our formulas are still missing the $3k$ cases. Circulants in the $DPCPD$ form seem to fit TTS (28,20) well: neither dimension is divisible by 3, and the proportion of zeros in each cross-term is $41/81$, very close to one-half. It works for SFLASH^{v2} (central map of $\mathrm{GF}(2^7)^{37} \to \mathrm{GF}(2^7)^{26}$) too, but SFLASH^{v2} takes most of its time in the central map so the time savings are scarce.

4.4 The Quirks of Circulants: Filling in the Blanks

We can summarize our problems encountered and observation made in trying to find a nice regular form for S_{3k} in the following proposition :

Proposition 3. *Let the $n \times n$ circulant $S_{n,k}$ have first row $[\underbrace{1,\ 1,\ldots,1}_{k},\ \underbrace{0,\ldots,\ 0}_{n-k}]$, then*

1. *$S_{n,k}$ is singular if (and only if) $d = \gcd(n,k) > 1$;*
2. *For large enough n with $\gcd(n,q) = 1$, some $S_{n,k}^{-1}$ will have q lines of 1's.*
3. *If n is even, $\min_{k>1}$(# of 1's in a row of $S_{n,k}^{-1}$) is the smallest odd prime $p \not| n$.*

Here d and q are defined as odd (since we are playing in $\mathrm{GF}(2)$).

Proposition 3 implies that it is more difficult to find nice matrices for $n = 3k$ (or $n = 6k$, assuming only even dimensions). If we do not want to avoid multiples of 3, we need a circulant $S_{3k} \in \{0,1\}^{3k\times 3k}$ with about two-thirds of its entries being 1, such that S_{3k}^{-1} has only three 1's in a row.

Proposition 4 (Trilinear Circulants of Size $3k$). *We can find circulants $\hat{C}_{k,j}$ so that $\hat{C}_{k,j}^{-1}$ exists in $\mathrm{GF}(2)$ and has $(2k-1)$ 1's in every column. One way is to form $\hat{C}_{k,j}$ from this row:*

$$110^j10^{3k-3-j} = [1,\ 1,\ \overbrace{0,\ldots,\ 0}^{j},\ 1,\ \overbrace{0,\ldots,\ 0}^{3k-3-j}],$$

We can choose S_{3k} satisfying the requirements in Sec. 4.3 among such $\hat{C}_{k,j}^{-1}$.

Again we need to leave the proofs to the more complete Research Report [45].

4.5 Performance Evaluation

Armed with the above, for any TTS central map, we can build a TTS variant with both M_1 and M_3 in $D_LP_LCP_RD_R$ form and call it TTS LPSQR. We use only elements of the subfield $\mathrm{GF}(16)$ for the coefficients in the central map. Also, The permutations are stored packed: for dimensions under 32, we only use 5 bits to an entry, so the private key totals 289 bytes (170/2 in ϕ_2 + 20 in $\mathbf{c}_3$ + 28 in $\mathbf{c}_1$ + $(20+28)\times(1+\frac{5}{8})\times 2$

Table 2. Performance of various signature schemes on a 500MHz Pentium III

Scheme	Signature	Pub. Key	Priv. Key	KeyGen	Signing	Verifying
RSA-PSS	1024 bits	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA	326 bits	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
SFLASH	259 bits	15.4 kB	2.4 kB	1.2 sec	1.6 ms	0.28 ms
TTS (20,28)	224 bits	8.6 kB	1.4 kB	15 ms	51 μs	0.11 ms
w.LPSQR	224 bits	8.6 kB	289 B	16 ms	28 μs	0.11 ms
TTS (24,32)	256 bits	13.4 kB	1.8 kB	25 ms	67 μs	0.18 ms
w.LPSQR	256 bits	13.4 kB	455 B	27 ms	41 μs	0.18 ms

in M_1, M_3). We compare the speed of TTS, TTS LPSQR and alternatives[5] in Table 2. The tests compile with only `gcc -O3` (ver. 3.3) using plain ANSI C and no inlined assembly. For TTS, there is an expected speedup.

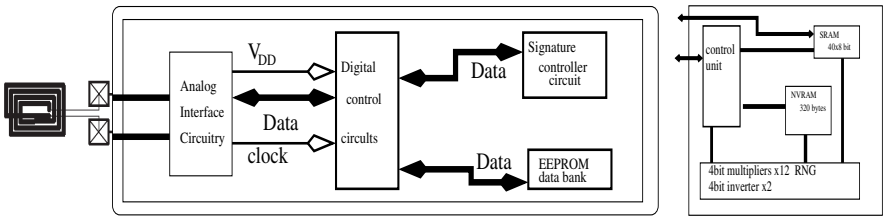
5 ASIC Implementation

The following represents our attempt to implement the enTTS scheme straight into an ASIC for verification purposes. This is to avoid any new vulnerabilities peculiar to structured matrices. To our best knowledge, this work appears to be the first ASIC implementation of a multivariate signature scheme in the literature, although there have been ASIC implementations of multivariate encryption schemes, e.g., NtruEncrypt [17].

In application scenarios like RFID-based security systems, typically we require a tag to authenticate itself to a reader, e.g., using the tag as a cryptographic token. In this case, only the signature functionality is needed on the tag. To meet the stringent resource constraint posed by devices like RFID tags, we only implement the signature functionality on the tags, leaving verification to the more resource-abundant (memory in particular) readers.

We first describe a hardware design that implements the plain enTTS (20,28) signature scheme without key shortening and report its synthesis result on the TSMC 0.25 μm process. The architecture of our implementation is depicted in Fig. 1 (b). It is divided into 4 portions:

1. SRAM cells: 81 bytes (8 for loop control and temporary storage, 28 for storing the signature, and 45 for solving a 9×9 linear system using Lanczos' method), taking about 9,900 gate equivalents.
2. NVRAM block: 1579 bytes (private key and look-up tables), taking about 6,400 gate equivalents.
3. Arithmetic Unit: about 800 gate equivalents, containing a set of 12 GF(16) multipliers and 2 GF(16) inverters, as well as an 8-bit integer unit for loop control.
4. Control Unit: about 3,900 gate equivalents, containing a microcoded state engine (200), microcode firmware (1,100), and an address generating unit (2,600).



(a) (b)

Fig. 1. Functional block diagrams for the ASIC implementation: (a) the entire RFID tag (b) inside the signature controller circuit

[5] ECC, RSA, and QUARTZ data from NESSIE website [33]. For earlier data of TTS, see [43].

The total synthesized circuit contains about 21,000 gate equivalents. Signing a message takes about 60,000 cycles, which translates to 0.6 seconds on a 100kHz clock. Most of the running time is spent in Lanczos' method. To save storage space, we do not store the matrix entries when inverting ϕ_2, so we have to generate the entries on the fly. A back-of-envelope calculation reveals that it takes about 1,000 cycles to perform a matrix-vector multiplication or to compute a bilinear form, and each iteration in the Lanczos method involves 5 such operations, adding up to a bit less than 6,000 cycles per iteration. Typically it takes 9 iterations to solve a 9×9 system, and we need to solve two such systems when inverting ϕ_2. As a result, we spend almost 90% of the time in Lanczos' method. The measured time consumed in Lanczos' method is close to 0.5 seconds per signing, fairly close to what we have estimated.

We note that it is possible to further speed up the operation by either using more SRAM or introducing parallelism. We report our attempt along the first direction. By adding 9 bytes of SRAM (totaling 54 bytes, 45 for storing a symmetric 9×9 matrix and 9 for a vector), we are able to use symmetric Gaussian elimination in place of Lanczos' method. The idea is to generate $M^T M$, solve $M^T M x = M^T w$, and then verify if $Mx = w$. This way we are able to cut the running time to about 0.16 seconds (a direct extrapolation would give 0.22, but there are some improvements we can do), less than one third the time that Lanczos' method takes. If situation permits, it is possible to halve the running time to about 0.077 seconds by adding 36 more bytes and doing a full-fledged Gaussian elimination.

At a first glance, such a strategy may not look so attractive in a resource-constrained environment such as RFID tags, particularly because SRAM cells are very expensive in terms of gate counts and power consumption. However, it will make sense from energy consumption's point of view, since cutting running time can reduce the total energy consumption and thus may achieve a higher energy efficiency. Experiments indeed validate our hypothesis. To our surprise, possibly due to the reduced switching activities in SRAM (e.g., due to less memory accesses) when doing symmetric Gaussian elimination, the power consumption of which is actually lower than that of doing Lanczos, even though the former uses more SRAM cells.

We summarize our synthesis results on the TSMC 0.25 μm process and a 100kHz clock in Table 3. Finally, with circulant (LPSQR) form of enTTS (20,28), we can go down to as low as 0.044 seconds per signing and 17,000 gates using Gaussian elimination, or 0.13 seconds and 14,000 gates using symmetrized Gaussian.

Table 3. Simulation and synthesis result on the TSMC 0.25μm process

	Lanczos		*Sym. GE*		*GE*	
Component	Gates	μA	Gates	μA	Gates	μA
SRAM	9,902	7.5	10,782	6.9	15,057	12.4
NVRAM	6,401	2.7	6,399	3.4	6,399	2.2
Arithmetic	768	4.3	768	3.5	768	4.2
Control Unit	4,207	6.2	4,009	7.5	3,689	6.6
Total	21,278	20.7	21,958	21.3	25,913	25.4

6 Concluding Remarks

NESSIE [33] did not recommend SFLASH for general use but said that it was very efficient on low-cost smart cards, where the size of the public key is not a constraint.

We think that the techniques we proposed here help with the issue of overly large keys. *This discussion applies also to other multivariate PKC schemes.* We think that multivariate schemes will be better contenders against the established schemes if they are a lot better customized and optimized.

The usual heir apparent if RSA/ECC is toppled would be NTRU. In fact NTRUEncrypt has many good qualities. We point out that diversity is a good thing, and NTRUSign is not yet as widely implemented. So we think that there is still value to our implementations here.

We Believe That the Most Important Issue Is Still Provable Security for Multivariates. Aside from that, there are at least the following directions to explore: optimization towards smaller keys and better performance in multivariate PKCs and their security estimates. Once key-shortening techniques, e.g., in the LPSQR form or other forms, are considered safe, they can be implemented for a lot of savings in space and time, stimulating more interest towards multivariate schemes. There should still be a lot to work for in this multivariate arena.

References

1. M. Akkar, N. Courtois, R. Duteuil, and L. Goubin, *A Fast and Secure Implementation of SFLASH*, PKC'03, LNCS 2567, pp. 267–278.
2. C.-Y. Chou, Y.-H. Hu, F.-P. Lai, L.-C. Wang, and B.-Y. Yang, *Tractable Rational Map Signature*, PKC'05, LNCS 3386, pp. 244–257.
3. N. Courtois, *Generic Attacks and the Security of Quartz*, PKC'03, LNCS 2567, pp. 351–364.
4. N. Courtois, L. Goubin, W. Meier, and J. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, PKC'02, LNCS 2274, pp. 211–227.
5. N. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, EUROCRYPT 2000, LNCS 1807, pp. 392–407.
6. J. Daemen and V. Rijmen, *The Design of Rijndael, AES - the Advanced Encryption Standard.* Springer 2002.
7. P. Davis, *Circulant matrices*, John Wiley & Sons, New York-Chichester-Brisbane, 1979.
8. C. Diem, *The XL-algorithm and a conjecture from commutative algebra*, ASIACRYPT'04, LNCS 3329, pp. 338–353.
9. J. Ding, *A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation*, PKC'04, LNCS 2947, pp. 305–318.
10. J. Ding, J. Gower *et al*, *Innoculating Multivariate Schemes against Differential Attacks*, `http://eprint.iacr.org/2005/255/`.
11. J. Ding and D. Schmidt, *Rainbow, a new Digitial Multivariate Signature Scheme*, ACNS'05, LNCS 3531, pp. 164-177.
12. J. Ding and Z. Yin, *Cryptanalysis of TTS and tame-like multivariable signature schemes*, presentation, IWAP'04.
13. J.-C. Faugère, *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)*, Proceedings of ISSAC'02, pp. 75-83, ACM Press 2002.

14. J.-C. Faugère, invited talk at AES4 conference, and private communication.
15. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, *Strong Authentication for RFID Systems Using the AES Algorithm*, CHES 2004, LNCS 3156, pp. 357–370.
16. M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, Freeman and Co., 1979, p. 251.
17. G. Gaubatz, J.-P. Kaps, and B. Sunar, *Public Key Cryptography in Sensor Networks—Revisited*, 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), LNCS 3313, Heidelberg, Germany, August, 2004.
18. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, *The nesC Language: A Holistic Approach to Networked Embedded Systems*, ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI), San Diego, CA, USA, June, 2003.
19. H. Gilbert and M. Minier, *Cryptanalysis of SFLASH*, EUROCRYPT 2002, LNCS 2332, pp. 288–298.
20. W. Geiselmann, R. Steinwandt, and T. Beth, *Attacking the Affine Parts of SFLASH*, 8th International IMA Conference on Cryptography and Coding, LNCS 2260, pp. 355–359.
21. L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem*, ASIACRYPT 2000, LNCS 1976, pp. 44–57.
22. L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th Annual ACM Symposium on the Theory of Computing, (May '96) pp. 212–220.
23. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, *System Architecture Directions for Networked Sensors*, Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (November 2000), pp. 93–104.
24. Y. Hu, L. Wang, J. Chen, F. Lai, and C. Chou, *A Performance Report and Security Analysis of a fast TTM implementation*, 2003 IEEE Int'l Symp. on Information Theory, Yokohama, Japan, June 2003.
25. Y. Hu, L. Wang, F. Lai, and C. Chou, *Similar Keys of Multivariate Quadratic Public Key Cryptosystems*, CANS'05, LNCS 3810, pp. 211-222.
26. A. Joux, S. Kunz-Jacques, F. Muller, P.-M. Ricordel, *Cryptanalysis of the Tractable Rational Map Cryptosystem*, PKC'05, LNCS 3386, pp. 258–274.
27. A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, CRYPTO'99, LNCS 1592, pp. 206–222.
28. R. Lidl and H. Niederreiter, *Finite Fields*. Addison-Wesley, 1984.
29. S. Ljungkvist, in the 8051 code library `http://www.8052.com/codelib.phtm`
30. D. Malan, M. Welsh, and M. Smith, *A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography*, First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, CA, USA, October, 2004.
31. T. Matsumoto and H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, LNCS 330, pp. 419–453.
32. M. Matsumoto and T. Nishimura, *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Trans. on Modeling and Computer Sim., 8 (1998), pp. 3-30.
33. The NESSIE project homepage: `http://www.cryptonessie.org`.
34. C. Paar, *Some Remarks on Efficient Inversion in Finite Fields*, 1995 IEEE International Symposium on Information Theory, Whistler, B.C. Canada, September 1995, available from the author's website.
35. C. Paar, *A New Architechture for a Parallel Finite Field Multiplier with Low Complexity Based on Composition Fields*, Brief Contributions section of *IEEE Transactions on Computers*, vol. 45(1996), No. 7, pp. 856–861.

36. J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, CRYPTO'95, LNCS 963, pp. 248–261.
37. J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, EUROCRYPT'96, LNCS 1070, pp. 33–48.
38. J. Patarin, L. Goubin, and N. Courtois, *C^*_{-+} and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai*, ASIACRYPT'98, LNCS 1514, pp. 35–49.
39. J. Patarin, N. Courtois, and L. Goubin, *FLASH, a Fast Multivariate Signature Algorithm*, CT-RSA'01, LNCS 2020, pp. 298–307. Updated version available at `http://www.cryptonessie.org`
40. P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, Proc. 35nd Annual Symposium on Foundations of Computer Science (S. Goldwasser, ed.), IEEE Computer Society Press (1994), 124-134.
41. C. Wolf and B. Preneel, *Taxonomy of Public-Key Schemes based on the Problem of Multivariate Quadratic Equations*, `http://eprint.iacr.org/2005/077`.
42. C. Wolf and B. Preneel, *Equivalent Keys in HFE, C^*, and variations,* In Mycrypt'05, LNCS 3715, pp. 33-49, 2005.
43. B.-Y. Yang and J.-M. Chen, *Rank Attacks and Defence in Tame-Like Multivariate PKC's*, ACISP 2005, LNCS 3574, p. 518–531. Older version at E-Print Archive 2004/061.
44. B.-Y. Yang, Y.-H. Chen, and J.-M. Chen, *TTS: High-Speed Signatures on a Low-Cost Smart Card*, CHES'04, LNCS 3156, pp. 371–385.
45. B.-Y. Yang, C.-M. Cheng, B.-R. Chen, and J.-M. Chen, Technical Research Report Number 11, 2005, Taiwan Information Security Center (TWISC).

A Cryptographical Experiments

Hazards (cf. Appendix A) facing multivariate digital signature schemes include algebraic attacks, mainly XL-Gröbner-Bases [5, 8, 13], searching [4], rank attacks [21, 43], and other tailored attacks. In this section, we report our effort to make sure that our shortened keys stand up against all known attacks.

We randomly generated enTTS (with both miniature and normal instances; see [43]) and SFLASH/C^{*-} public keys (for control purposes) using both generic and LPSQR matrices, and put them through equation-solving mechanisms using Gröbner Bases, FXL simulators and rank attacks. Also small tests were run with Magma (including $\mathbf{F_4}$, the precursor to $\mathbf{F_5}$). We conducted repeated runs of tests until our test machines (with 3 GB of RAM) ran out of memory. Each test was run at least 100 times except for the maximum-sized one listed, repeated only 70 times.

Table 4. List of tests run to look for differences between generic and proposed shortened keys

Test	Test Size	enTTS	C^{*-}
FXL	Up to 14 remaining variables	no	no
MAGMA $\mathbf{F_4}$	Up to 8 eqs & vars	no	no
High Rank	Up to 16×16 matrices	no	no
Low Rank	Up to 16×16 matrices	no	no
Oil+Vinegar	Up to 12×12 matrices	no	no

The conclusion is that the shortened keys behave identically under Gröbner and XL to generic keys and do not have rank vulnerabilities. We hope to try again later with more memory and better optimized programs, perhaps by implementing Faugère's $\mathbf{F_5}$ [13] for ourselves. We must note that there is no guarantee for the security of our schemes solely based on the result of these tests, but cryptologists are still working on getting some measure of reductionist security for multivariates.