Speeding up QUAD

Albrecht Petzoldt

Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
apetzoldt@cdc.informatik.tu-darmstadt.de

Abstract. QUAD is a provable secure stream cipher based on multivariate polynomials which was proposed in 2006 by Berbain, Gilbert and Patarin [1]. In this paper we show how to speed up QUAD over GF(256) by a factor of up to 5.8. We get this by using structured systems of polynomials, in particular partially circulant polynomials and polynomials generated by a linear recurring sequence (LRS), instead of random ones. By using this strategy, we can also reduce the system parameter of QUAD by about 99 %. We furthermore present experiments, which seem to show that using structured polynomials of this special choice does not influence the security of QUAD.

Keywords: Multivariate Cryptography, QUAD Stream Cipher, Partially Circulant Polynomials, Linear Recurring Sequences

1 Introduction

Stream ciphers are an important cryptographic primitive from the area of symmetric cryptography. The key idea of this construction is to generate a keystream of the same length as the message to be encrypted. The ciphertext is then obtained by bitwise XORing of message and keystream. In the last years, many new stream ciphers have been proposed, like HC-256 [8] and Salsa20 [2]. Another approach for generating the keystream is to use a block cipher like AES in the OFB mode.
A third direction is the design of provable secure stream ciphers based on hard mathematical problems. As examples for those schemes we mention here the stream cipher based on the Blum Blum Shub PRNG [4] and the code based stream cipher SYND [5].
Another example for such a scheme is the multivariate stream cipher QUAD, which was proposed in 2006 by Berbain, Gilbert and Patarin [1]. The security of QUAD can be reduced to the MQ Problem of solving systems of multivariate quadratic polynomials over a finite field $\mathbb{F}$. This problem is proven to be NP hard, even for the case of quadratic polynomials over the field GF(2). Furthermore, in contrast to number theoretic problems like integer factorization and discrete logarithms, it is believed to resist quantum computer attacks [3]. The main disadvantage of QUAD is the low speed of the keystream generation process which is about 1,000 times slower than that of non provable secure stream ciphers.
In this paper we show how the keystream generation of QUAD can be sped up by using systems of structured polynomials. In particular, we use polynomial systems whose coefficients are repeated by a cyclic shift and systems generated by linear recurring sequences (LRS) instead of random ones. By doing so, we can achieve a speed up of the key stream generation process by a factor of up to 5.8. We show by computer experiments that the security of QUAD is not weakened by this approach. The speed up of the keystream generation process is shown both by a theoretical analysis and by a C implementation of the schemes.
The remainder of this paper is organized as follows. In Section 2 we give a short introduction into multivariate cryptography and define what we mean by a "partially circulant" and by an "LRS" system of polynomials. Section 3 describes the QUAD stream cipher, whereas Section 4 deals with the evaluation of polynomial systems and shows how this step can be improved by using structured systems of polynomials. Finally, Section 5 presents the results of our computer experiments and Section 6 concludes this paper.

2 Multivariate Polynomials

Let $\mathcal{P}$ be a system of m quadratic polynomials in n variables over a finite field $\mathbb{F}$. $\mathcal{P}$ can be written in the form

$$\begin{array}{c} p_{11}^{(1)} \cdot x_1^2 + p_{12}^{(1)} \cdot x_1 x_2 + \ldots + p_{nn}^{(1)} \cdot x_n^2 + p_1^{(1)} \cdot x_1 + \ldots + p_n^{(1)} \cdot x_n + p_0^{(1)} \\ p_{11}^{(2)} \cdot x_1^2 + p_{12}^{(2)} \cdot x_1 x_2 + \ldots + p_{nn}^{(2)} \cdot x_n^2 + p_1^{(2)} \cdot x_1 + \ldots + p_n^{(2)} \cdot x_n + p_0^{(2)} \\ \vdots \\ p_{11}^{(m)} \cdot x_1^2 + p_{12}^{(m)} \cdot x_1 x_2 + \ldots + p_{nn}^{(m)} \cdot x_n^2 + p_1^{(m)} \cdot x_1 + \ldots + p_n^{(m)} \cdot x_n + p_0^{(m)}. \end{array} \tag{1}$$

The security of multivariate schemes is based on the

MQ-Problem: Given a system $\mathcal{P}$ of m multivariate polynomials $p^{(1)}, \ldots, p^{(m)}$, find a vector $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ such that $p^{(1)}(\bar{x}) = \ldots = p^{(m)}(\bar{x}) = 0$.

The MQ-problem is proven to be NP hard even for the case of quadratic polynomials over the field GF(2).

Definition 1. *For a multivariate quadratic polynomial $p(x_1, \ldots, x_n)$ as shown in equation (1) we define the coefficient vector with respect to the graded lexicographic ordering of monomials by*

$$\Phi(p) = (p_{11}, p_{12}, \ldots, p_{nn}, p_1, \ldots, p_n, p_0) \in \mathbb{F}^D \tag{2}$$

with $D = \frac{(n+1)\cdot(n+2)}{2}$.

Definition 2. *For a system $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ of multivariate quadratic polynomials the Macauley matrix M_P is defined by $M_P = (\Phi(p^{(1)}), \ldots \Phi(p^{(m)}))^T$, i.e.*

$$M_P = \begin{pmatrix} p_{11}^{(1)} & p_{12}^{(1)} & \ldots & p_{nn}^{(1)} & p_1^{(1)} & \ldots & p_n^{(1)} & p_0^{(1)} \\ p_{11}^{(2)} & p_{12}^{(2)} & \ldots & p_{nn}^{(2)} & p_1^{(2)} & \ldots & p_n^{(2)} & p_0^{(2)} \\ \vdots & & & & & & & \vdots \\ p_{11}^{(m)} & p_{12}^{(m)} & \ldots & p_{nn}^{(m)} & p_1^{(m)} & \ldots & p_n^{(m)} & p_0^{(m)} \end{pmatrix}. \tag{3}$$

Definition 3. *We call a system of polynomials partially circulant, if for each $i \in \{2, \ldots, m\}$ the i-th row of the matrix M_P is given as the cyclic right shift of the $(i-1)$-th row.*

To generate a partially circulant system of polynomials, we randomly choose a vector $\mathbf{b} = (b_1, \ldots, b_D) \in \mathbb{F}^D$ and define the i-th row of the Macauley matrix M_P by

$$M_P[i] = \mathcal{R}^{i-1}(\mathbf{b}) \quad (i = 1, \ldots, m), \tag{4}$$

where $\mathcal{R}^i(\mathbf{b})$ denotes the cyclic right shift of the vector $\mathbf{b}$ by i positions. Therefore, the matrix M_P of a partially circulant system of polynomials $\mathcal{P}$ looks like

$$M_P = \begin{pmatrix} b_1 & b_2 & b_3 & \ldots & b_{D-1} & b_D \\ b_D & b_1 & b_2 & \ldots & b_{D-2} & b_{D-1} \\ b_{D-1} & b_D & b_1 & \ldots & b_{D-3} & b_{D-2} \\ \vdots & & & & & \vdots \\ b_{D-m+2} & b_{D-m+3} & b_{D-m+4} & \ldots & b_{D-m} & b_{D-m+1} \end{pmatrix}. \tag{5}$$

Definition 4. *Let L be a positive integer and $\gamma_1, \ldots, \gamma_L$ be given elements of a finite field $\mathbb{F}$. A linear recurring sequence (LRS) of length L is a sequence $\{s_1, s_2, \ldots\}$ of $\mathbb{F}$-elements satisfying the relation*

$$s_j = \gamma_1 \cdot s_{j-1} + \gamma_2 \cdot s_{j-2} + \cdots + \gamma_L \cdot s_{j-L} = \sum_{i=1}^{L} \gamma_i \cdot s_{j-i} \quad \forall j > L. \tag{6}$$

The values $s_1, \ldots, s_L$ are called the initial values of the LRS.

Definition 5. *The connection polynomial of an LRS is defined as*

$$C(x) = \gamma_L X^L + \gamma_{L-1} X^{L-1} + \cdots + \gamma_1 \cdot X + 1 = \sum_{i=1}^{L} \gamma_i X^i + 1.$$

A linear recurring sequence S of length L is uniquely determined by its initial state $s_1, s_2, \ldots, s_L$ and the connection polynomial $C(x)$. Therefore we write $S = \mathrm{LRS}(s_1, \ldots, s_L, C(x))$.

Definition 6. *We call a multivariate quadratic polynomial p generated by an LRS S of length L, if its coefficient vector $\Phi(p)$ (see Definition 1) consists of the first D elements of S.*

Definition 7. *We call a system of polynomials an LRS-system of length L, if each of its components is a polynomial generated by an LRS of length L.*

Throughout this paper we look at LRS-systems of length 1. So whenever we speak of an LRS-system of polynomials, we refer to an LRS-system of length 1. To generate such a system, we choose a vector $\gamma \in \mathbb{F}^m$ and define the i-th row of the Macauley matrix M_P by

$$M_P[i] = (1, \gamma_i, \gamma_i^2, \ldots, \gamma_i^{D-2}, \gamma_i^{D-1}) \quad (i = 1, \ldots, m). \tag{7}$$

Therefore, the Macauley matrix of an LRS-system $\mathcal{P}$ looks like

$$M_P = \begin{pmatrix} 1 & \gamma_1 & \gamma_1^2 & \ldots & \gamma_1^{D-2} & \gamma_1^{D-1} \\ 1 & \gamma_2 & \gamma_2^2 & \ldots & \gamma_2^{D-2} & \gamma_2^{D-1} \\ \vdots & & & & & \vdots \\ 1 & \gamma_m & \gamma_m^2 & \ldots & \gamma_m^{D-2} & \gamma_m^{D-1} \end{pmatrix}. \tag{8}$$

To ensure that the single polynomials of the system are linearly independent, the elements of the vector γ must be pairwise distinct.

Table 1 shows, that Gröbner Basis algorithms can not distinguish between random, partially circulant and LRS-systems of polynomials and behave very similar for all three types. For our experiments we used MAGMA version 2.13-10 and solved determined systems over GF(256) using the MAGMA command `Variety`.

		$n = 9$	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 14$
random system	time (s)	5.5	40.9	300.2	2,391	19,054	169,317
	degree	11	12	13	14	15	16
	matrix size	2034 × 2188	4146 × 4529	8904 × 9400	18662 × 19834	38493 × 40158	79738 × 83640
part. circulant system	time (s)	5.4	40.6	300.0	2,390	19,046	168,846
	degree	11	12	13	14	15	16
	matrix size	2034 × 2188	4146 × 4529	8904 × 9400	18662 × 19834	38493 × 40158	79738 × 83640
LRS system	time (s)	5.4	40.4	299.9	2,386	18,964	169,152
	degree	11	12	13	14	15	16
	matrix size	2034 × 2188	4146 × 4529	8904 × 9400	18662 × 19834	38493 × 40158	79738 × 83640

Table 1. Experiments with determined systems

3 The QUAD stream cipher

QUAD is a provable secure multivariate stream cipher which was introduced in 2006 by Berbain, Gilbert and Patarin [1]. The security of QUAD is based on the MQ-Problem of solving nonlinear polynomial systems over a finite field.

Like all stream ciphers QUAD encrypts a message by producing a keystream ks which has the same length as the message. The ciphertext c of the message m is then created by simply bitwise XORing of message and keystream, i.e.

$$c_i = m_i \oplus ks_i \quad \forall i = 0, \ldots, \mathrm{Len}(m) - 1.$$

A ciphertext c is decrypted in the same way, namely

$$m_i = c_i \oplus ks_i \quad \forall i = 0, \ldots, \mathrm{Len}(c) - 1.$$

The keystream of QUAD is generated as follows. Let $\mathbb{F}$ be a finite field. One chooses 4 multivariate quadratic systems $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{S}_0$ and $\mathcal{S}_1 : \mathbb{F}^n \to \mathbb{F}^n$ [1]. These four systems are viewed as system parameters and are fixed for a large number of users. Before encrypting a message, a user chooses a key $k \in \mathbb{F}^n$ and an initial vector $\mathrm{IV} \in \{0,1\}^{80}$. The keystream of QUAD is then generated by following Algorithms 1 and 2. Figure 1 shows a graphical description of the keystream generation process. In Algorithm 2 we set $L = \lceil \frac{\mathrm{Len}(m)}{\lg(q) \cdot n} \rceil$, where q is the cardinality of the underlying field.

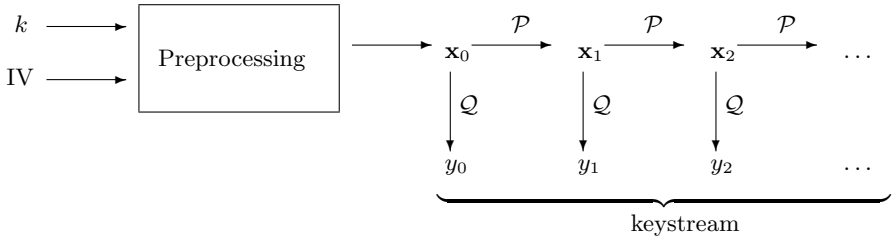

Fig. 1. keystream generation

In the rest of this paper, we look at the QUAD scheme defined over the field with 256 elements. Therefore we have

$$L = \lceil \frac{\mathrm{Len}(m)}{8 \cdot n} \rceil.$$

4 Evaluation of Polynomials

The most expensive part during the keystream generation of QUAD is the evaluation of polynomial systems. We look at two different ways to perform this step and show how we can improve its efficiency significantly by using structured systems of polynomials.

[1] While, for the functioning of QUAD the three systems, $\mathcal{S}_0, \mathcal{S}_1$ and $\mathcal{P}$ are required to be determined, we do not need this property for the map $\mathcal{Q}$. In particular, the system $\mathcal{Q}$ could be overdetermined ($m > n$), which makes the key generation process more efficient. But, since overdetermined systems are easier to solve, we choose for reasons of security $m = n$.

Algorithm 1 Preprocessing

Input: key $k \in \mathbb{F}^n$, IV $\in \{0,1\}^{80}$
Output: initial state IS $\in \mathbb{F}^n$

IS $\leftarrow k$
for $i = 0$ to 79 **do**
  if IV[i]=1 **then**
    IS $\leftarrow \mathcal{S}_1$(IS)
  else
    IS $\leftarrow \mathcal{S}_0$(IS)
  end if
end for
for $i = 0$ to 79 **do**
  IS $\leftarrow \mathcal{P}$(IS)
end for
return IS

Algorithm 2 keystream generation

Input: initial state IS $\in \mathbb{F}^n$
Output: keystream $ks \in \mathbb{F}^{m \cdot L}$

ks $\leftarrow []$
for $i = 0$ to $L - 1$ **do**
  $ks \leftarrow ks \| \mathcal{Q}$(IS)
  IS $\leftarrow \mathcal{P}$(IS)
end for
return ks

In the first way (later referred to as the standard approach) a polynomial system $\mathcal{P}$ is stored in the form of its Macauley matrix (see Definition 2).
When evaluating the system $\mathcal{P}$ at a point $\mathbf{x} = (x_1, \ldots, x_n)$, we first compute an $\frac{(n+1)\cdot(n+2)}{2}$ vector mon containing the values of all monomials of degree ≤ 2, i.e.

$$\text{mon} = (x_1^2, x_1x_2, \ldots, x_n^2, x_1, \ldots, x_n, 1). \tag{9}$$

Then we have

$$\mathcal{P}(\mathbf{x}) = \begin{pmatrix} M_P[1] \cdot \text{mon}^T \\ M_P[2] \cdot \text{mon}^T \\ \vdots \\ M_P[m] \cdot \text{mon}^T \end{pmatrix}, \tag{10}$$

where $M_P[i]$ denotes the i-th row of the Macauley matrix M_P.

To evaluate an $m \times n$ system $\mathcal{P}$ by this approach, one needs

- $\frac{n\cdot(n+1)}{2}$ field multiplications to compute the vector mon of equation (9) and
- $m \cdot \left(\frac{n\cdot(n+1)}{2} + n\right)$ field multiplications to compute the scalar products of equation (10).

Therefore, for the whole evaluation process one needs

$$\frac{n}{2} \cdot ((m+1) \cdot (n+1) + 2 \cdot m) \tag{11}$$

field multiplications.
When evaluating the system $\mathcal{P}$ according to the alternative approach, we store the coefficients of $\mathcal{P}$ in m upper triangular $(n+1) \times (n+1)$ matrices $MP^{(1)}, \ldots, MP^{(m)}$:

$$MP^{(1)} = \begin{pmatrix} p_{11}^{(1)} & p_{12}^{(1)} & \cdots & p_{1n}^{(1)} & p_1^{(1)} \\ 0 & p_{22}^{(1)} & \cdots & p_{2n}^{(1)} & p_2^{(1)} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & 0 & p_{nn}^{(1)} & p_n^{(1)} \\ 0 & \cdots & \cdots & 0 & p_0^{(1)} \end{pmatrix}, \ldots, \; MP^{(m)} = \begin{pmatrix} p_{11}^{(m)} & p_{12}^{(m)} & \cdots & p_{1n}^{(m)} & p_1^{(m)} \\ 0 & p_{22}^{(m)} & \cdots & p_{2n}^{(m)} & p_2^{(m)} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & 0 & p_{nn}^{(m)} & p_n^{(m)} \\ 0 & \cdots & \cdots & 0 & p_0^{(m)} \end{pmatrix}. \tag{12}$$

Furthermore we define for a vector $\mathbf{x} = (x_1, \ldots, x_n)$ the extended vector $\hat{\mathbf{x}} = (x_1, \ldots, x_n, 1)$.
With this notation, we get

$$\mathcal{P}(\mathbf{x}) = \begin{pmatrix} \hat{\mathbf{x}} \cdot MP^{(1)} \cdot \hat{\mathbf{x}}^T \\ \hat{\mathbf{x}} \cdot MP^{(2)} \cdot \hat{\mathbf{x}}^T \\ \vdots \\ \hat{\mathbf{x}} \cdot MP^{(m)} \cdot \hat{\mathbf{x}}^T \end{pmatrix}. \tag{13}$$

To evaluate a single polynomial by this approach, one needs

- $\frac{(n+1)\cdot(n+2)}{2} - 1$ field multiplications to compute the matrix vector product $\text{temp} = \hat{\mathbf{x}} \cdot MP^{(k)}$ and
- $n+1$ field multiplications to compute the scalar product $\text{temp} \cdot \hat{\mathbf{x}}$.

Therefore, to evaluate the whole system $\mathcal{P}$ by the alternative approach, we need

$$m \cdot \left(\frac{(n+1)\cdot(n+4)}{2} - 1\right) \tag{14}$$

field multiplications.

As Table 2 shows, evaluating a random polynomial with the standard approach is more efficient than doing it with the alternative approach. But, when we look at structured systems of polynomials, we can evaluate equation (13) much faster. In the next two subsections we show how to do this for partially circulant and LRS systems of polynomials.

4.1 Partially circulant polynomials

For a partially circulant system of polynomials the matrices $MP^{(i)}$ look as shown in Figure 2. We have

$$MP^{(i)}_{j,k} = MP^{(i-1)}_{j,k-1} \; \forall i = 2, \ldots, m, \; k = 2, \ldots, n, \; j = 1, \ldots, k-1. \tag{15}$$

Therefore we get

$$(x_1, \ldots, x_{k-1}) \cdot \begin{pmatrix} MP^{(i)}_{1,k} \\ MP^{(i)}_{2,k} \\ \vdots \\ MP^{(i)}_{k-1,k} \end{pmatrix} = (x_1, \ldots, x_{k-1}) \cdot \begin{pmatrix} MP^{(i-1)}_{1,k-1} \\ MP^{(i-1)}_{2,k-1} \\ \vdots \\ MP^{(i-1)}_{k-1,k-1} \end{pmatrix} \; \forall i = 2, \ldots, m, \; k = 2, \ldots, n. \tag{16}$$

The boxes in Figure 2 illustrate this relation. The black boxes show the vector $(MP^{(i-1)}_{1,k-1}, \ldots, MP^{(i-1)}_{k-1,k-1})^T$ on the right hand side of the equation, whereas the blue boxes show the vector $(MP^{(i)}_{1,k}, \ldots, MP^{(i)}_{k-1,k})^T$ on the left hand side. Note that the blue boxes in the matrix $MP^{(i)}$ correspond exactly to the black ones in the matrix $MP^{(i-1)}$ $(i = 2, \ldots, m)$.
Algorithm 3 uses this fact to evaluate the system $\mathcal{P}$ in a more efficient way.

Algorithm 3 Evaluation of cyclic polynomials

Input: internal state $\mathrm{IS} \in \mathbb{F}^n$, partially circulant system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$
Output: result $\mathrm{res} = \mathcal{P}(\mathrm{IS})$

1: $\hat{\mathbf{x}} \leftarrow (\mathrm{IS}_1, \ldots, \mathrm{IS}_n, 1)$
2: **for** $i = 1$ to $n + 1$ **do**
3: $\quad \mathrm{temp}_i \leftarrow \sum_{j=1}^{i} MP^{(1)}_{ji} \cdot \hat{\mathbf{x}}_j$
4: **end for**
5: $\mathrm{res}_1 \leftarrow \sum_{j=1}^{n+1} \mathrm{temp}_j \cdot \hat{\mathbf{x}}_j$
6: **for** $l = 2$ to m **do**
7: $\quad$ **for** $i = n + 1$ to 2 by -1 **do**
8: $\quad\quad \mathrm{temp}_i \leftarrow \mathrm{temp}_{i-1} + MP^{(l)}_{ii} \cdot \hat{\mathbf{x}}_i$
9: $\quad$ **end for**
10: $\quad \mathrm{temp}_1 \leftarrow MP^{(l)}_{11} \cdot \hat{\mathbf{x}}_1$
11: $\quad \mathrm{res}_l \leftarrow \sum_{j=1}^{n+1} \mathrm{temp}_j \cdot \hat{\mathbf{x}}_j$
12: **end for**
13: **return** res

Algorithm 3 works as follows:
The first polynomial is evaluated just as a random polynomial with the alternative approach. In the loop (line 2 to 4) we compute the vector $\mathrm{temp} = \hat{\mathbf{x}} \cdot MP^{(1)}$. In line 5 the algorithm then computes the product $\mathrm{temp} \cdot \hat{\mathbf{x}}^T$. In the big loop (line 6 to 12) we evaluate the remaining polynomials $p^{(2)}, \ldots, p^{(m)}$. Again we start by computing the matrix vector product $\mathrm{temp} = \hat{\mathbf{x}} \cdot MP^{(l)}$ (line 7 to 10). During this step we can reuse the values of temp_i $(i = 1, \ldots, n)$ computed for the previous polynomial. In fact, we can compute the product $\hat{\mathbf{x}} \cdot MP^{(l)}$ $(l = 2, \ldots, m)$ by using only n multiplications (instead of $\frac{(n+1)\cdot(n+2)}{2}$). In line 11 we finally compute the product $\mathrm{temp} \cdot \hat{\mathbf{x}}^T$.

To evaluate the whole system Algorithm 3 needs only

$$\frac{(n+1) \cdot (n+4)}{2} + (m-1) \cdot (2 \cdot n + 1) \tag{17}$$

field multiplications. For a QUAD system with $m = n = 26$ Algorithm 3 reduces the number of field multiplications by a factor of 5.9 (compared to the evaluation of a random polynomial with the standard approach).

$$MP^{(1)} = \begin{pmatrix} b_1 & b_2 & b_3 & \dots & b_{n-1} & b_n & b_{n+1} \\ 0 & b_{n+2} & b_{n+3} & \dots & b_{2n-1} & b_{2n} & b_{2n+1} \\ 0 & 0 & b_{2n+2} & \dots & b_{3n-2} & b_{3n-1} & b_{3n} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-5} & b_{D-4} & b_{D-3} \\ 0 & & \dots & & 0 & b_{D-2} & b_{D-1} \\ 0 & & \dots & & \dots & 0 & b_D \end{pmatrix}$$

$$MP^{(2)} = \begin{pmatrix} b_D & b_1 & b_2 & \dots & b_{n-2} & b_{n-1} & b_n \\ 0 & b_{n+1} & b_{n+2} & \dots & b_{2n-2} & b_{2n-1} & b_{2n} \\ 0 & 0 & b_{2n+1} & \dots & b_{3n-3} & b_{3n-2} & b_{3n-1} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-6} & b_{D-5} & b_{D-4} \\ 0 & & \dots & & 0 & b_{D-3} & b_{D-2} \\ 0 & & \dots & & \dots & 0 & b_{D-1} \end{pmatrix}$$

$$\vdots$$

$$MP^{(n-1)} = \begin{pmatrix} b_{D-n+3} & b_{D-n+4} & b_{D-n+5} & \dots & b_1 & b_2 & b_3 \\ 0 & b_4 & b_5 & \dots & b_{n+1} & b_{n+2} & b_{n+3} \\ 0 & 0 & b_{n+4} & \dots & b_{2n-2} & b_{2n-1} & b_{2n} \\ \vdots & & & \ddots & \vdots & & \vdots \\ 0 & & & \dots & b_{D-n-3} & b_{D-n-2} & b_{D-n-1} \\ 0 & & & \dots & 0 & b_{D-n} & b_{D-n+1} \\ 0 & & & \dots & \dots & 0 & b_{D-n+2} \end{pmatrix}$$

$$MP^{(n)} = \begin{pmatrix} b_{D-n+2} & b_{D-n+3} & b_{D-n+4} & \dots & b_D & b_1 & b_2 \\ 0 & b_3 & b_4 & \dots & b_n & b_{n+1} & b_{n+2} \\ 0 & 0 & b_{n+3} & \dots & b_{2n-3} & b_{2n-2} & b_{2n-1} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-n-4} & b_{D-n-3} & b_{D-n-2} \\ 0 & & \dots & & 0 & b_{D-n-1} & b_{D-n} \\ 0 & & \dots & & \dots & 0 & b_{D-n+1} \end{pmatrix}$$

Fig. 2. Matrices $MP^{(i)}$ for cyclicQUAD

4.2 LRS-system

$$MP^{(i)} = \begin{pmatrix} 1 & \gamma_i & \gamma_i^2 & \cdots & \cdots & \gamma_i^{n-2} & \gamma_i^{n-1} & \gamma_i^{n} \\ 0 & \gamma_i^{n+1} & \gamma_i^{n+2} & \cdots & \cdots & \gamma_i^{2n-2} & \gamma_i^{2n-1} & \gamma_i^{2n} \\ 0 & 0 & \gamma_i^{2n+1} & \cdots & \cdots & \gamma_i^{3n-3} & \gamma_i^{3n-2} & \gamma_i^{3n-1} \\ \vdots & & \ddots & & & \vdots & \vdots & \vdots \\ \vdots & & & & & \vdots & \vdots & \vdots \\ \vdots & & & & \ddots & \gamma_i^{D-6} & \gamma_i^{D-5} & \gamma_i^{D-4} \\ 0 & \cdots & & \cdots & & 0 & \gamma_i^{D-3} & \gamma_i^{D-2} \\ 0 & \cdots & & \cdots & & & 0 & \gamma_i^{D-1} \end{pmatrix} \qquad (i = 1, \ldots, n)$$

Fig. 3. Matrices $MP^{(i)}$ for LRSQUAD

For an LRS-system of length 1 the matrices $MP^{(i)}$ look as shown in Figure 3. We have

$$MP^{(i)}_{j,k} = \gamma_i \cdot MP^{(i)}_{j,k-1} \;\forall\; i = 1, \ldots, m,\;\; k = 2, \ldots, n,\;\; j = 1, \ldots, k-1. \tag{18}$$

Therefore we get

$$(x_1, \ldots, x_{k-1}) \cdot \begin{pmatrix} MP^{(i)}_{1,k} \\ MP^{(i)}_{2,k} \\ \vdots \\ MP^{(i)}_{k-1,k} \end{pmatrix} = \gamma_i \cdot (x_1, \ldots, x_{k-1}) \cdot \begin{pmatrix} MP^{(i)}_{1,k-1} \\ MP^{(i)}_{2,k-1} \\ \vdots \\ MP^{(i)}_{k-1,k-1} \end{pmatrix} \;\forall i = 1, \ldots, m,\;\; k = 2, \ldots, n. \tag{19}$$

The boxes in Figure 3 illustrate this relation: The black boxes show the vector $(MP^{(i)}_{1,k-1}, \ldots, MP^{(i)}_{k-1,k-1})^T$ on the right hand side of the equation, whereas the blue boxes represent the vector $(MP^{(i)}_{1,k}, \ldots, MP^{(i)}_{k-1,k})^T$ on the left hand side. Note that any blue box can be obtained by multiplying the corresponding black box by γ_i.
Algorithm 4 uses this fact to evaluate a LRS-system much faster than a random system.

Algorithm 4 Evaluation of LRS polynomials

Input: internal state $\mathrm{IS} \in \mathbb{F}^n$, LRS system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$
Output: result $\mathrm{res} = \mathcal{P}(\mathrm{IS})$

1: $\hat{\mathbf{x}} \leftarrow (\mathrm{IS}_1, \ldots, \mathrm{IS}_n, 1)$
2: **for** $l = 1$ to m **do**
3: $\quad \mathrm{temp}_1 \leftarrow \hat{\mathbf{x}}_1$
4: $\quad$ **for** $i = 2$ to $n + 1$ **do**
5: $\quad\quad \mathrm{temp}_i \leftarrow \gamma_i \cdot \mathrm{temp}_{i-1} + MP^{(1)}_{ii} \cdot \hat{\mathbf{x}}_i$
6: $\quad$ **end for**
7: $\quad \mathrm{res}_l \leftarrow \sum_{j=1}^{n+1} \mathrm{temp}_j \cdot \hat{\mathbf{x}}_j$
8: **end for**
9: **return** res

Algorithm 4 works as follows: Each of the m polynomials of the system $\mathcal{P}$ is evaluated individually. From line 3 to 6 we compute the product $\hat{\mathbf{x}} \cdot MP^{(i)}$ and store the result in the vector temp. During the computation of temp_i $(i = 2, \ldots, m)$ we can reuse the value of temp_{i-1}, which enables us to compute the vector temp using only $2 \cdot n$ field multiplications. Finally, in line 7, Algorithm 4 computes the result of the evaluation by computing the scalar product $\text{temp} \cdot \hat{\mathbf{x}}$. For this step we need $n + 1$ field multiplications.

Therefore, we can evaluate the whole system using only

$$3 \cdot m \cdot n + m \tag{20}$$

field multiplications. For a QUAD system with $m = n = 26$ over GF(256) Algorithm 4 reduces the number of field multiplications by a factor of 6.0 (compared to the evaluation of a random polynomial with the standard approach). In contrast to Algorithm 3, Algorithm 4 is easily parallellizable.

5 Results

We checked our theoretical results by a straightforward C implementation of the QUAD stream cipher. The results are shown in Table 2 and 3.

	message size	1MB		10MB	
			r. f. [1]		r. f. [1]
random system standard approach	time (ms)	6,263	-	62,294	-
	cycles (10^9)	15.8	-	157	-
random system alternative approach	time (ms)	8,738	-	86,783	-
	cycles (10^9)	21.1	-	210	-
part. circ. system	time (ms)	1,121	5.5	11,145	5.5
	cycles (10^9)	2.86	5.5	28.2	5.5
LRS-system	time (ms)	1,092	5.8	10,734	5.8
	cycles (10^9)	2.74	5.8	27.23	5.8

[1] reduction factor to random systems with the standard approach

Table 2. Running time of QUAD

	Data Throughput (kB/s)	CPU cycles/byte	speed up factor
random system standard approach	157.3	15,777	-
random system alternative approach	111.3	21,860	-
part. circ. system	853.6	2,820	5.5
LRS-system	872.7	2,730	5.8

Table 3. Data Throughput of QUAD

As can be seen from the table above, evaluating random polynomials following the standard approach is more efficient than doing it with the alternative approach. By using partially circulant systems of polynomials we can achieve a speed up factor of 5.5, when using LRS polynomials the speed up factor is 5.8.

6 Conclusion

In this paper we have shown a way how to speed up the QUAD stream cipher by a factor of up to 5.8 by using structured systems of polynomials instead of random ones. Furthermore, we have given evidence that Gröbner Basis algorithms like MAGMA's F_4 can not use the structure of our polynomials to speed up the computation. Therefore, using structured polynomials for QUAD instead of random ones might be an alternative worth considering.

References

[1] C. Berbain, H. Gilbert and J. Patarin: QUAD: A Practical Stream Cipher with Provable Security. EUROCRYPT 2006, LNCS vol. 4004, pp. 109-128. Springer 2006.
[2] D.J. Bernstein: The Salsa20 family of stream ciphers. The eSTREAM Finalists, LNCS vol. 4986, pp. 84-97. Springer, 2008.
[3] D.J. Bernstein, J. Buchmann and E. Dahmen (eds.): Post-Quantum Cryptography. Springer 2009.
[4] L. Blum, M. Blum, and M. Shub: A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing 15, pp. 364–383, 1986.
[5] P. Gaborit, C. Laudaroux and N. Sendrier. SYND : a very fast code-based cipher stream with a security reduction. ISIT'07, pp. 186 - 190, 2007.
[6] A. Petzoldt, S. Bulygin and J. Buchmann: Fast Verification for Improved Versions of the UOV and Rainbow Signature Schemes. PQCrypto 2013, to appear.
[7] A. Petzoldt and S. Bulygin: Linear Recurring Sequences for the UOV Key Generation Revisited. ICISC 2012, LNCS vol. 7378, pp. 441 - 456. Springer, 2013.
[8] H. Wu: A New Stream Cipher HC-256. FSE 2004, LNCS vol. 3017, pp. 226–244. Springer 2004.