Recursive Diffusion Layers for (Lightweight) Block Ciphers and Hash Functions

Shengbao Wu[1,2], Mingsheng Wang[3], and Wenling Wu[3]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.O. Box 8718, China
[2] Graduate School of Chinese Academy of Sciences, Beijing 100190, China
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
mingsheng_wang@yahoo.com.cn,
{wushengbao,wwl}@is.iscas.ac.cn

Abstract. Diffusion layers with maximum branch numbers are widely used in block ciphers and hash functions. In this paper, we construct recursive diffusion layers using Linear Feedback Shift Registers (LFSRs). Unlike the MDS matrix used in AES, whose elements are limited in a finite field, a diffusion layer in this paper is a square matrix composed of linear transformations over a vector space. Perfect diffusion layers with branch numbers from 5 to 9 are constructed. On the one hand, we revisit the design strategy of PHOTON lightweight hash family and the work of FSE 2012, in which perfect diffusion layers are constructed by one bundle-based LFSR. We get better results and they can be used to replace those of PHOTON to gain smaller hardware implementations. On the other hand, we investigate new strategies to construct perfect diffusion layers using more than one bundle-based LFSRs. Finally, we construct perfect diffusion layers by increasing the number of iterations and using bit-level LFSRs. Since most of our proposals have lightweight examples corresponding to 4-bit and 8-bit Sboxes, we expect that they will be useful in designing (lightweight) block ciphers and (lightweight) hash functions.

Keywords: Recursive diffusion Layers, linear transformation, branch number, MDS matrix, Linear Feedback Shift Register (LFSR).

1 Introduction

Diffusion layer is one of the core components in a block cipher with confusion layer. And it is also widely used in many other block cipher-based primitives, for instance, hash functions. The choice of a diffusion layer influences both the security and the efficiency of a cryptographic primitive. On the one hand, it plays an important role in providing security against differential cryptanalysis [2] and linear cryptanalysis [12], which are the two most important cryptanalysis of block ciphers. On the other hand, with the same security, an elaborate diffusion layer may lead to a better performance of a cryptographic primitive on hardware or/and software implementation.

The strength of a diffusion layer is usually measured by the notation of branch number. A block cipher using a diffusion layer with a small branch number may suffer unexpected attacks. Therefore, how to construct diffusion layers with big branch numbers and low-cost implementations is a challenge for designers.

The most attractive diffusion layers are those with maximum branch numbers, which are also called perfect or MDS diffusion layers. The common approach to construct them is to extract MDS matrices from MDS codes [11]. Thus, these diffusion layers have matrix representations over $\mathbb{F}_{2^n}$, where n is usually consistent to the bit length of Sbox used in the confusion layer. Many block ciphers [1,14,6,7], especially AES, use this design strategy to construct their diffusion layers.

A problem using MDS matrices as that in AES is that they cannot be implemented in an extremely compact way on hardware. Thus, they are unfitted in resource constrained environments, such as RFID systems and sensor networks. To conquer this drawback while maintain the maximum branch number, a new design strategy was proposed in the document of PHOTON lightweight hash family [9] and then used in designing the diffusion layer of LED lightweight block cipher [8]. Without extracting an MDS matrix in one step, the new strategy constructs a diffusion layer with a bundle-based linear feedback shift register (LFSR)(see Fig.1). That is, in each step, only the last bundle is updated by a linear combination of all of the bundles while other bundles are obtained by shifting the state vector by one position to the left. Each L_i is chosen as a multiplication with an element in $\mathbb{F}_{2^n}$. The LFSR will iterate s times and output the final state. Suppose A is the state transition matrix of LFSR, then the diffusion layer obtained by this strategy is the matrix A^s over $\mathbb{F}_{2^n}$.

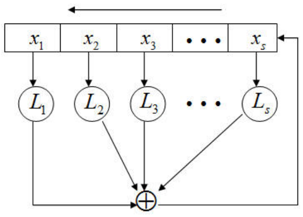

Fig. 1. LFSR for constructing diffusion layers in PHOTON

As mentioned in [9], this design is very compact in hardware implementation because it only needs to realize the LFSR and allows to re-use the existing memory with neither temporary storage nor additional control logic required. Of course, designers would like the final matrix (i.e., A^s) to be MDS, so as to maintain as much diffusion as for the previous strategies. On the other hand, AES-based method (i.e., lookup tables) can be used to implement such cryptographic primitives in software without suffering their efficiency.

In FSE 2012, Sajadieh *et.al* [13] extended this design strategy and proposed a list of perfect diffusion layers. They considered a linear transformation L of vector space $\mathbb{F}_2^n$ and chose $L_i = \sum_{j=-1}^{2} a_i^{(j)} \cdot L^j$, where $a_i^{(j)} \in F_2$ and $1 \leq i \leq s$. The final matrix (i.e., A^s) obtained from this strategy can be treated as an $sn \times sn$ matrix over $\mathbb{F}_2$ or an $s \times s$ matrix composed of linear transformations over $\mathbb{F}_2^n$. To make it perfect, L and $a_i^{(j)}$s should satisfy some conditions. In [13], firstly, the authors studied the sufficient conditions that make a specific diffusion layer with $s = 4$ prefect and then investigated the conditions of other proposals with a necessary statement.

As multiplications with elements in $\mathbb{F}_{2^n}$ are specific linear transformations of vector space $\mathbb{F}_2^n$, the new strategy provides more choices in constructing diffusion layers. Thus, designers may obtain perfect diffusion layers with smaller hardware implementations.

Our Contributions. In this paper, we focus on constructing recursive diffusion layers using LFSRs, following and extending the design strategy of PHOTON and [13]. We construct a list of lightweight perfect diffusion layers with maximum branch numbers from 5 to 9. They mainly distribute in two classes — one class of them are generated by one bundle-based LFSR, using the design strategy of PHOTON and [13], while another class of them are constructed by new strategies using more than one bundle-based LFSRs. Our proposals have smaller hardware implementations than diffusion layers given in PHOTON lightweight hash family and [13]. And they can be used to replace those of PHOTON lightweight hash family. The best replacement can save 22.3% gate equivalents (GE) in the diffusion layer. Finally, we construct perfect diffusion layers by increasing the number of iterations and using bit-level LFSRs.

Outline of This paper. In Section 2, we introduce the definitions of linear transformation, determinant of a matrix over commutative rings and branch number. Previous results on judging perfect diffusion layers are also discussed. Our strategy and some criteria for constructing perfect diffusion layers are described in Section 3. In Section 4 and Section 5, we illustrate our results generated by bundle-based LFSRs. Then, we compare our results with known perfect diffusion layers in Section 6. In Section 7, we investigate some possible manners of constructing new perfect diffusion layers using LFSRs. Finally, we conclude this paper.

2 Preliminaries

In this section, we first introduce the definitions of linear transformation and determinant of a matrix over commutative rings. Then, we introduce the notation of branch number and several statements for constructing prefect diffusion layers.

2.1 Linear Transformation

If V is a vector space over $\mathbb{F}_2$, then a *linear transformation* of V is a map $L : V \to V$ such that

$$L(\mathbf{u} \oplus \mathbf{v}) = L(\mathbf{u}) \oplus L(\mathbf{v}) \tag{1}$$

holds for any $\mathbf{u}, \mathbf{v}$ in V. L is invertible if it is injective and surjective. If linear transformation $L_3 = L_2 \circ L_1$, that is, $L_3(\mathbf{v}) = L_2(L_1(\mathbf{v}))$, then L_3 is invertible if and only if L_1 and L_2 are invertible.

Since there is a square matrix M over $\mathbb{F}_2$ such that $L(\mathbf{v}) = M \cdot \mathbf{v}$, the invertibility of L is equivalent to the non-singularity of M. Thus, in the subsequent discussions, we directly use a matrix to represent a linear transformation. One familiar class of linear transformations is the multiplication with an element in $\mathbb{F}_{2^n}$, that is, $L(\mathbf{v}) = a \cdot \mathbf{v}$, where $a, \mathbf{v} \in \mathbb{F}_{2^n}$. Notice that a can be represented as an $n \times n$ matrix over $\mathbb{F}_2$ if we treat $\mathbf{v}$ as a vector in $\mathbb{F}_2^n$.

2.2 Matrix over Commutative Rings

In this section, we first review several statements of matrix theorem which are true over any commutative ring $\mathcal{R}$. More information is advised to [4]. Then, we introduce a specific commutative ring which is used in this work.

Similar to the classical definition of the determinant, we have

Definition 1. *[4] Let $A = (A_{i,j})_{1 \le i \le s, 1 \le j \le s}$ be an $s \times s$ matrix with entries in a commutative ring $\mathcal{R}$. The determinant of A, denoted by $det(A)$, is the following element of $\mathcal{R}$:*

$$det(A) = \sum_{\sigma \in P(s)} sgn(\sigma) A_{1,\sigma(1)} A_{2,\sigma(2)} \cdots A_{s,\sigma(s)},$$

where $P(s)$ denotes the set of all permutations on s letters and $sgn(\sigma) \in \{1, -1\}$ is the sign of $\sigma \in P(s)$.

Then, $det(AB) = det(A)det(B)$ and $det(A^T) = det(A)$. Here, A^T is the transposition of A. Similarly, we have

Theorem 1. *[4] Let $A = (A_{i,j})_{1 \le i \le s, 1 \le j \le s}$, then A is invertible if and only if $det(A) \in U(\mathcal{R})$, where $U(\mathcal{R})$ is the set of all invertible elements in ring $\mathcal{R}$.*

Now, suppose L is an $n \times n$ non-singular matrix over $\mathbb{F}_2$ and

$$S = \{\sum a_{-i} L^{-i} + a_0 + \sum a_j L^j : i, j \in \mathbb{Z}^+, a_{-i}, a_0, a_j \in \mathbb{F}_2\}$$

is a set which includes all polynomials of L and L^{-1}. Then, the set S together with the addition of $\mathbb{F}_2$ and the multiplication of polynomials, form a commutative ring. We denote it by $\mathbb{F}_2[L, L^{-1}]$. Then, we have

Proposition 1. *Let B be an element of $\mathbb{F}_2[L, L^{-1}]$, then $B \in U(\mathbb{F}_2[L, L^{-1}])$ if and only if B is a $n \times n$ non-singular matrix over $\mathbb{F}_2$.*

Proof. If $B \in U(\mathbb{F}_2[L, L^{-1}])$, then there is a $C \in \mathbb{F}_2[L, L^{-1}]$ such that $BC = I$. Thus, when treat B, C and I as matrices over $\mathbb{F}_2$, the determinant $|B| = 1$, i.e, B is non-singular. On the contrary, if B is non-singular over $\mathbb{F}_2$, then there is a positive integer m such that $B^m = I$ (Notice that $m \leq 2^n - 1$ is a finite integer). The smallest m is called the order of B and can be efficiently computed by methods introduced in [5]. Since $B \in \mathbb{F}_2[L, L^{-1}]$, then $B^{n-1} \in \mathbb{F}_2[L, L^{-1}]$ and it is the inverse of B in $\mathbb{F}_2[L, L^{-1}]$. Thus, B is invertible.

2.3 Branch Number

Suppose $\mathbf{v}$ is a vector with s bundles, i.e., $\mathbf{v} = (v_1, v_2, \ldots, v_s)$, where each $v_i \in \mathbb{F}_2^n$ is a vector over a finite field $\mathbb{F}_2$ with length n. The bundle weight of a vector $\mathbf{v}$, denoted by $w_b(\mathbf{v})$, is equal to the number of non-zero bundles. Then, we have

Definition 2. *[7] The differential branch number of a linear diffusion layer D is given by*

$$\mathcal{B}_d(D) = \min_{\mathbf{v} \neq \mathbf{0}}(w_b(\mathbf{v}) + w_b(D(\mathbf{v}))), \tag{2}$$

where D can be represented as an $sn \times sn$ matrix over $\mathbb{F}_2$ or an $s \times s$ matrix consisting of linear transformations of $\mathbb{F}_2^n$.

Similarly, we can define the linear branch number.

Definition 3. *[7] The linear branch number of a linear diffusion layer D is given by*

$$\mathcal{B}_l(D) = \min_{\mathbf{v} \neq \mathbf{0}}(w_b(\mathbf{v}) + w_b(D^T(\mathbf{v}))), \tag{3}$$

where D^T is the transposition of D.

Theorem 2. *[7] A linear diffusion layer D has a maximum differential branch number if and only if it has a maximum linear branch number.*

For a diffusion layer acting on s bundles, the maximal $\mathcal{B}_d$ and $\mathcal{B}_l$ is $s+1$, known as the singleton bound [11]. And D is called a perfect or MDS diffusion layer if it takes its maximal $\mathcal{B}_d$ and $\mathcal{B}_l$.

2.4 Linear Diffusion Layers with Maximum Branch Numbers

In MDS codes, the most widely used property for constructing an MDS matrix is

Theorem 3. *[11] An $[m, s, d]$ code with generator matrix $G = [I_{s\times s} D_{s\times(m-s)}]$ is an MDS code if and only if every square submatrix of D is non-singular, where D is a matrix over $\mathbb{F}_{2^n}$.*

In the Proposition 3.1 and Proposition 3.2 of [3], Blaum *et.al* showed that Theorem 3 is also valid even we substitute every element of D as a linear transformation of vector space $\mathbb{F}_2^n$. Notice that now

$$D = \begin{pmatrix} D_{1,1} & D_{1,2} & \cdots & D_{1,m-s} \\ D_{2,1} & D_{2,2} & \cdots & D_{2,m-s} \\ \vdots & \vdots & \ddots & \vdots \\ D_{s,1} & D_{s,2} & \cdots & D_{s,m-s} \end{pmatrix} \tag{4}$$

is a block matrix with s rows and $m-s$ columns, where each $D_{i,j}$ is an $n \times n$ matrix over $\mathbb{F}_2$.

We denote by $\mathcal{D}_{i\times j}$ a submatrix obtained from D by deleting $(s-i)$ block rows and $(m-s-j)$ block columns while maintaining the order of other elements in D. When considering a linear diffusion layer, D is a square matrix, that is, $m = 2s$. Then, the results of [3] can be re-described as the following statement.

Theorem 4. *A linear diffusion layer D has a maximum branch number if and only if every square submatrix of D, i.e., $\mathcal{D}_{k\times k}$ for $1 \le k \le s$, is non-singular.*

To detect a perfect linear diffusion layer, we need to judge whether all $\sum_{k=1}^{s}\binom{s}{k}\binom{s}{k} = \binom{2s}{s} - 1$ submatrices of D are non-singular according to Theorem 4.

Remark 1. In the paper [13], Sajadieh *et. al* used the necessary part of this statement to describe the conditions of some perfect diffusion layers. Now, we know that they are enough to make those diffusion layers perfect.

3 Our Strategy for Constructing Diffusion Layers

In this paper, we will construct perfect diffusion layers using different kinds of LFSRs. The strategy introduced in this section are suitable for bundle-based LFSRs, that is, one or several bundles are updated in each step while others are obtained by the shift operation. The procedure has four steps.

1. Construct an $s \times s$ matrix $A = (A_{i,j})_{1\le i,j\le s}$ with each $A_{i,j} = \sum a_k^{(i,j)} \cdot L^k \in \mathbb{F}_2[L, L^{-1}]$. Of course, matrix A will be chosen with some structures for low-cost hardware implementations and reducing the search space.
2. Choose an integer d and compute $D = A^d$ $(d \ge 1)$ as the final diffusion layer. Since D is a matrix over $\mathbb{F}_2[L, L^{-1}]$, from Theorem 1 and Theorem 4, we deduce that D is perfect if and only if the determinant of each square submatrix of D is an invertible element in $\mathbb{F}_2[L, L^{-1}]$.
3. Generate the determinants of all square submatrices of D (i.e., $\mathcal{D}_{k\times k}$ for $1 \le k \le s$) as the conditions, which is a set of polynomials in $\mathbb{F}_2[L, L^{-1}]$. Notice that if zero is in the condition set, we know D can not be MDS. In this case, we will change the choice of A or d.
4. Search whether there exists any L such that all polynomials obtained in step 3 are invertible elements in $\mathbb{F}_2[L, L^{-1}]$. Based on Proposition 1, we need to check whether all conditions are non-singular matrix over $\mathbb{F}_2$.

The procedure above can be performed systematically on a computer. To find perfect diffusion layers with low-cost faster, several criteria are used in this paper.

1. Choose $A_{i,j}$ with few terms. That is, the number of 1's in the coefficient list $[\ldots, a_{-1}^{(i,j)}, a_0^{(i,j)}, a_1^{(i,j)}, \ldots]$ should be as few as possible. The degree of L and L^{-1} are also chosen to be low. Thus, $A_{i,j}$ may be chosen as 0, that is, zero transformation.

2. The integer d should be chosen as small as possible, since it determines the efficiency of getting the final diffusion layer D in both hardware and software implementation.
3. The linear transformation L should be low-cost in hardware implementation. Our chief targets are linear transformations with no more than one XOR gate (Note: one XOR gate needs about 2.66 GE in hardware implementation). Multiplications with elements in $\mathbb{F}_{2^n}$ will be the secondary choices.

Additionally, for practical applications, we expect that

4. The perfect diffusion layers proposed in this paper should have examples for $n = 4$ and $n = 8$, since 4-bit Sboxes and 8-bit Sboxes are involved in many cryptographic primitives.

Remark 2. Suppose $\mathbf{y} = L{\cdot}\mathbf{x}$, where L is an invertible matrix and $\mathbf{x}, \mathbf{y}$ are column vectors in $\mathbb{F}_2^n$. A linear transformation without any XOR gate is a permutation of the input bits, that is, $\mathbf{y}_i = \mathbf{x}_{i'}$, where $[1', 2', \ldots, n']$ is a permutation of $[1, 2, \ldots, n]$. Thus, L is a matrix with exactly n nonzero entries, satisfying that each row and each column of L have exactly one nonzero entry. Similarly, a linear transformation with only one XOR gate is a matrix with exactly $n + 1$ nonzero entries, satisfying (1) each row and each column have at least one nonzero entries and (2) there exists a unique row that has two nonzero entries.

All choices of L with no more than one XOR gate is $n! + n! \cdot (n^2 - n)$. For $n = 4$ and $n = 8$, we may enumerate all of them efficiently. For $n \geq 16$, only a small part of them can be enumerated. In this paper, we fix $L[i, i + 1] = 1$ (for $1 \leq i \leq n - 1$) and $L[n, 1] = 1$, and the search space is reduced to $n^2 - n + 1$.

In the subsequent two sections, we illustrate our results in constructing perfect diffusion layers using bundle-based LFSRs. One class of them are obtained by iterating one LFSR several times, following the design strategy of PHOTON and [13]. In this design, only one bundle is updated while others are obtained by shifting the state vector by one position to the left. Then, we extend the strategy to find them using several bundle-based LFSRs in an iteration.

4 Construct Perfect Diffusion Layers with One Bundle-Based LFSR

In this section, we revisit the design strategy of PHOTON and [13], which constructs recursive diffusion layers with one bundle-based LFSR. That is, we try to construct its state transition matrix A and choose an iteration number d such that $D = A^d$ is perfect, where

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ L_1 & L_2 & L_3 & \cdots & L_s \end{pmatrix} \tag{5}$$

and $L_i = \bigoplus a_k^{(i)} \cdot L^k$. For simplicity, we only extract the final row of A, that is,

$$A_{lfsr}^{(s)} = [L_1, L_2, \dots, L_s],$$

to illustrate our choice of the LFSR. The cost of A in hardware implementation is $(s-1)n + \sum_{i=1}^{s} \#L_i$ XOR gates if all L_is are nonzero elements of $\mathbb{F}_2[L, L^{-1}]$, for which $\#L_i$ XOR gates are allocated to the linear transformation L_i. To compare our results with those given in LED, PHOTON and [13], we limit $d \leq s$ in this section.

4.1 Perfect Diffusion Layers for $s = 4$

The best result we find for $s = 4$ is

$$A_{lfsr}^{(4)} = [L, 1, 1, L^2] \tag{6}$$

with $d = 4$. It costs $3n + \#L + \#L^2$ XOR gates in hardware implementation.

For the convenience of comprehension, we use this example to display how to generate a condition set. Other proposals in this paper are done similarly.

Following the strategy discussed in Section 3, once $A_{lfsr}^{(4)} = [L, 1, 1, L^2]$ and $d = 4$ are chosen, we calculate

$$D = A^4 = \begin{pmatrix} L & 1 & 1 & L^2 \\ L^3 & L^2 + L & L^2 + 1 & L^4 + 1 \\ L^5 + L & L^4 + L^3 + 1 & L^4 + L^2 + L + 1 & L^6 + 1 \\ L^7 + L & L^6 + L^5 + L + 1 & L^6 + L^4 + L^3 & L^8 + L^4 + L + 1 \end{pmatrix}.$$

Now, based on Theorem 4, we need to calculate the determinants of all the square submatrices $\mathcal{D}_{k \times k}$ of D. Suppose F is a determinant of $\mathcal{D}_{k \times k}$ for some k, which is a polynomial in $\mathbb{F}_2[L, L^{-1}]$, it can be factorized as

$$F = F_1^{i_1} \cdot F_2^{i_2} \cdots F_j^{i_j},$$

where $F_1, \dots, F_j$ are irreducible polynomials and $i_1, \dots, i_j$ are positive integers. Then, F is non-singular if and only if its factors $F_1, \dots, F_j$ are non-singular. Thus, $F_1, \dots, F_j$ are added to the condition set. For example, suppose $k = 1$ and $\mathcal{D}_{1 \times 1} = D_{2,4} = L^4 + 1$, then $L + 1$ is added to the condition set since $L^4 + 1 = (L + 1)^4$.

After enumerating the determinants of all 69 square submatrices $\mathcal{D}_{k \times k}$ $(1 \leq k \leq 4)$, we conclude that $D = A^4$ with $A_{lfsr}^{(4)} = [L, 1, 1, L^2]$ has branch number 5, if the following 12 matrices

$$\begin{array}{ccc} L, & L + 1, & L^2 + L + 1, \\ L^3 + L + 1, & L^3 + L^2 + 1, & L^4 + L^3 + 1, \\ L^4 + L^3 + L^2 + L + 1, & L^5 + L^2 + 1, & L^5 + L^4 + L^3 + L + 1, \\ L^6 + L^5 + L^4 + L + 1, & L^6 + L^5 + L^4 + L^2 + 1, & L^7 + L^6 + L^5 + L^4 + 1 \end{array}$$

are non-singular.

Table 1. Lightweight linear transformations L for $A_{lfsr}^{(4)} = [L, 1, 1, L^2]$

Length of the input	example of L
$n = 4$	$[[2,3],3,4,1]$
$n = 8$	$[[5,6],7,5,8,4,3,1,2]$
$n = 16$	$[[1,2],3,4,\ldots,16,1]$
$n = 32$	$[[2,4],3,4,\ldots,32,1]$
$n = 64$	$[[2,6],3,4,\ldots,64,1]$

Finally, we introduce some lightweight linear transformations of $\mathbb{F}_2^n$ that satisfy the above conditions (see Table 1). Each of them costs only one XOR gate (2.66 GE) in hardware implementation. Thus, L^2 costs two XOR gates. Note that there are many other similar linear transformations. For simplicity, we extract the nonzero positions in each row of a matrix to represent it. For example, $[[2,3],3,4,1]$ is the representation of matrix $\begin{pmatrix} 0\,1\,1\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,0\,0\,0 \end{pmatrix}$.

Other Information from the Condition Set. From the condition set, we observe that $L^4 + L + 1$ does not belong to it. Thus, for $n = 4$, L can be chosen as the multiplication with α, i.e., $L(v) = \alpha \cdot v$ ($v \in \mathbb{F}_{2^4}$), where α is a root of the irreducible polynomial $x^4 + x + 1$. This L also costs one XOR gate in hardware implementation [8].

A question is that why the multiplication with α is also a valid choice. That is, after replacing L by the multiplication with α, can we make sure that all 12 conditions in the condition set are invertible elements of $\mathbb{F}_{2^4}$? The answer is yes and the reasons are shown as following.

– $1, \alpha, \alpha^2$ and α^3 compose a basis of $\mathbb{F}_{2^4}$, since α is a root of the irreducible polynomial $x^4 + x + 1$. That is, for each $\beta \in \mathbb{F}_{2^4}$, there is a unique vector $[a_0, a_1, a_2, a_3] \in \mathbb{F}_2^4$ such that $\beta = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3$.
– Suppose $g(x) \neq x^4 + x + 1$ is another irreducible polynomial, then

$$g(\alpha) \equiv a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 \bmod \alpha^4 + \alpha + 1$$

is a nonzero element. Thus, $g(\alpha)$ is invertible. For a further step, all conditions in the condition set are irreducible polynomials and not equal to $x^4 + x + 1$, which implies that they are invertible elements of $\mathbb{F}_{2^4}$ if L is chosen as the multiplication with α.

In general, if we observe that an irreducible polynomial $f(x) = x^n + \phi(x)$ does not belong to the condition set, then the multiplication with one of its roots can be chosen as a candidate of L to obtain a perfect diffusion layer over $\mathbb{F}_{2^n}$.

Table 2. Perfect diffusion layers we find for $5 \leq s \leq 8$, together with the cost in hardware implementation and the number of conditions

	$A_{lfsr}^{(s)}$ $(d = s)$	Cost (XOR gates)	No. of Cond.
$s = 5$	$[1, L^2, L^{-1}, L^{-1}, L^2]$	$4n + 2(\#L^2 + \#L^{-1})$	21
$s = 6$	$[1, L^{-2}, L^{-1}, L^2, L^{-1}, L^{-2}]$	$5n + \#L^2 + 2(\#L^{-2} + \#L^{-1})$	90
$s = 7$	$[1, L, L^{-5}, 1, 1, L^{-5}, L]$	$6n + 2(\#L + \#L^{-5})$	592
$s = 8$	$[1, L^{-3}, L, L^3, L^2, L^3, L, L^{-3}]$	$7n + \#L^2 + 2(\#L + \#L^3 + \#L^{-3})$	2629

Table 3. Lightweight linear transformations L for $A_{lfsr}^{(s)}$ with $5 \leq s \leq 8$

Length of the input	example of L	fit for
$n = 4$	$[[2, 3], 3, 4, 1]$	$s = 5, 6, 7, 8$
$n = 8$	$[[5, 6], 7, 5, 8, 4, 3, 1, 2]$	$s = 5, 6, 7, 8$
$n = 16$	$[[1, 2], 3, 4, \ldots, 16, 1]$	$s = 5, 6$
$n = 16$	$[[2, 6], 3, 4, \ldots, 16, 1]$	$s = 7, 8$
$n = 32$	$[[2, 10], 3, 4, \ldots, 32, 1]$	$s = 5, 6, 7, 8$
$n = 64$	$[[2, 3], 3, 4, \ldots, 64, 1]$	$s = 6, 7$
$n = 64$	$[[2, 17], 3, 4, \ldots, 64, 1]$	$s = 5, 8$

Table 4. Comparison of our diffusion layers with those used in PHOTON

	P_{100}	P_{144}	P_{196}	P_{256}	P_{288}
(s, n)	$(5, 4)$	$(6, 4)$	$(7, 4)$	$(8, 4)$	$(6, 8)$
PHOTON	75.33 GE	80 GE	99 GE	145 GE	144 GE
Ours	58.52 GE	74.48 GE	95.76 GE	117.04 GE	127.68 GE
Reduced(%)	22.3	6.9	3.3	19.3	11.3

4.2 Results for $5 \leq s \leq 8$

The best results we find for $5 \leq s \leq 8$ are listed in Table 2. In this table, we also list their cost in hardware implementation. With the increment of s, the number of conditions that must be satisfied increases rapidly. Due to the lack of space, we only introduce the conditions for $s = 5$ in Appendix A.

Several lightweight examples of these diffusion layers are given in Table 3. All of them and their inverses only cost one XOR gate in hardware implementation. An interesting observation is that $L^4 + L + 1$ is not included in any of these condition sets. Thus, the multiplication with α is also a choice for $n = 4$, where α is a root of irreducible polynomial $x^4 + x + 1$.

Application. PHOTON lightweight hash family has 5 variants according to the size of its internal permutation. Our perfect diffusion layers can be used to substitute those of PHOTON and obtain smaller hardware implementations. The specification is given in Table 4. Note that our diffusion layers may perform better in practice under some available techniques. For instance, in the document

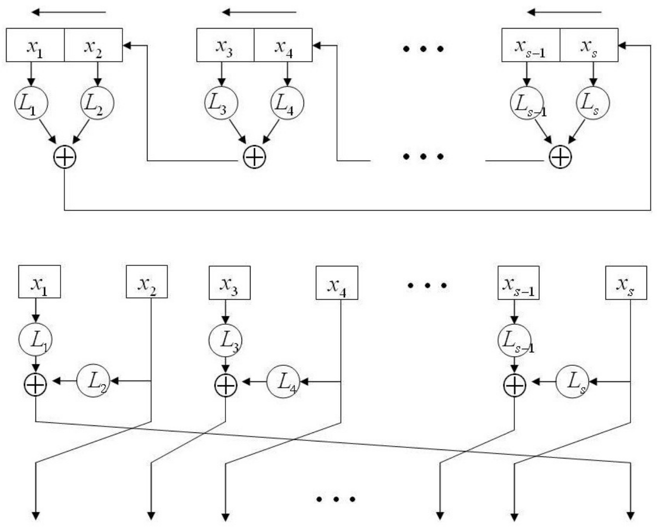

Fig. 2. Several bundle-based LFSRs for constructing diffusion layers

of PHOTON, the authors mentioned that using their library, the multiplications with α, α^2 and α^3 can be implemented in hardware with 2.66 GE, 4.66 GE and 7 GE when using the irreducible polynomial $x^4 + x + 1$, respectively. However, we evaluate them with 2.66 GE, 5.32 GE and 7.98 GE in Table 4, respectively. We would like to remark that the diffusion layer of P_{288} can be further improved using the results of the next section.

5 Construct Perfect Diffusion Layers with Several Bundle-Based LFSRs

In this section, we construct perfect diffusion layers with more than one bundle-based LFSRs. We consider $\frac{s}{2}$ LFSRs (see Fig.2, upper part, s is even) in an iteration, where each LFSR composed of two bundles and they form a head-tail connecting circle. In each step, the last bundle of each LFSR is updated by a linear combination of all of the bundles in the next LFSR while other bundles are obtained by shifting the state vector of each LFSR by one position to the left. Similar to the diffusion layers constructed by one LFSR, this mode also allows to re-use the existing memory with neither temporary storage nor additional control logic required.

From the point of view of block cipher structures, these LFSRs consist of a Type-II Generalized Feistel Structure (GFS, [15]) (see Fig.2, nether part). Thus, to obtain perfect diffusion layers, we firstly construct a matrix

$$A = \begin{pmatrix} T & U_2 & 0 & \cdots & 0 & 0 \\ 0 & T & U_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & T & U_{\frac{s}{2}} \\ U_1 & 0 & 0 & \cdots & 0 & T \end{pmatrix}, \tag{7}$$

and then make $D = A^d$ perfect, where "0" is a $2n \times 2n$ zero matrix over $\mathbb{F}_2$ while $T = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ and $U_i = \begin{pmatrix} 0 & 0 \\ L_{2i-1} & L_{2i} \end{pmatrix}$ $(1 \le i \le \frac{s}{2})$ are 2×2 block matrices. We also focus on $L_i = \bigoplus a_k^{(i)} \cdot L^k$ and use

$$A_{gfs}^{(s)} = [L_1, L_2, \ldots, L_s]$$

to indicate our choice of A. The cost of implementing the LFSRs is $\frac{s}{2} \cdot n + \sum_{i=1}^{s} \#L_i$ XOR gates if all L_is are nonzero elements in $\mathbb{F}_2[L, L^{-1}]$.

5.1 Perfect Diffusion Layers for $s = 4$

The best results we find for $s = 4$ is

$$A_{gfs}^{(4)} = [L, 1, 1, L] \tag{8}$$

with $d = 4$, if the following 7 matrices

$$L, \quad L+1, \; L^2+L+1, \; L^3+L+1,$$
$$L^3+L^2+1, \; L^4+L^3+1, \; L^4+L^3+L^2+L+1$$

are non-singular.

It costs $2n+2\#L$ XOR gates in hardware implementation. Since the condition set of this choice is included in that of $A_{lfsr}^{(4)} = [L, 1, 1, L^2]$ with $d = 4$, all examples listed in Table 1 and the multiplication with a root of the irreducible polynomial $x^4 + x + 1$ for $n = 4$ fit the above seven conditions.

5.2 Results for $s = 6$ and $s = 8$

The best results we find for $s = 6$ is

$$A_{gfs}^{(6)} = [L, 1, 1, L^2, L, L^2] \tag{9}$$

with $d = 6$. It costs $3n + 2\#L + 2\#L^2$ XOR gates in hardware implementation. And 196 conditions need to be satisfied. However, we do not find linear transformations with no more than one XOR gates when $n = 4$ and $n = 8$. For $n = 4$, we find all irreducible polynomials with degree 4 are included in the condition set. Thus, there is no choices for $L \in \mathbb{F}_{2^4}$. For $n = 8$, we find four irreducible polynomials $L^8 + L^6 + L^5 + L^2 + 1$, $L^8 + L^6 + L^3 + L + 1$,

$L^8 + L^6 + L^5 + L^4 + 1$ and $L^8 + L^7 + L^3 + L^2 + 1$ are not included in the condition set. Therefore, L can be chosen as the multiplication with a root of these polynomials, which costs 3 XOR gates in hardware implementation. Using this diffusion layer $(2.66 \times (3 \times 8 + 2 \times 3 + 2 \times 6) = 112$ GE) to substitute that used in P_{288} of PHOTON, we may save 22.2% gate equivalents.

The best results we find for $s = 8$ is

$$A^{(8)}_{gfs} = [1, L^4, 1, L^{-1}, 1, L, 1, L^2] \tag{10}$$

with $d = 8$. It costs $4n + \#L^4 + \#L + \#L^{-1} + \#L^2$ XOR gates in hardware implementation. 8692 conditions need to be satisfied. However, we do not find linear transformations with no more than one XOR gate when $n = 4$ and $n = 8$. What's more, all irreducible polynomials with degree 4 and 8 are included in the condition set. Thus, neither $L \in \mathbb{F}_{2^4}$ nor $L \in \mathbb{F}_{2^8}$ satisfies the condition set.

6 Comparison with Known Results

In this section, we compare our bundle-based proposals with those given in the document of LED, PHOTON and [13]. The comparison mainly consists of two parts — the cost in hardware implementation and low-cost examples for $n \in \{4, 8, 16, 32, 64\}$. Table 5 illustrates the comparison results. In this table, we generalize the choices in the LED and PHOTON hash family, which only considered the examples over $\mathbb{F}_{2^4}$, to check whether they have lightweight examples for $n \geq 4$. And diffusion layers proposed in [13] are also re-calculated under the process of Section 3. "Y" means we find an example with only one XOR gate, "Y_F" means we do not find examples with no more than one XOR gate, but there is an example if L is chosen as the multiplication with an element in $\mathbb{F}_{2^n}$. "N" means we find neither examples with no more than one XOR gate nor examples in $L \in \mathbb{F}_{2^n}$.

Some observations are given as follows.

1. All of our proposals (for $4 \leq s \leq 8$) using one bundle-based LFSR have examples with one XOR gate when $n \in \{4, 8, 16, 32, 64\}$, together with the diffusion layers used in LED, $P_{100}, P_{144}, P_{195}$ and P_{256} of PHOTON. And our proposals have smaller hardware implementation than them.
2. We compare our proposals with 9 diffusion layers given in [13]. We find four of them can not be perfect when $d = s$. Another four of them have bigger hardware implementation than our proposals. Only the diffusion layer $A^{(5)}_{lfsr} = [1, L^2, 1, 1, L]$ with $d = 5$ has slightly better performance than our proposal when $n \geq 16$. However, this diffusion layer does not have examples when $n = 4$ and it does not have examples with no more than one XOR gate when $n = 8$.
3. Diffusion layers with the smallest hardware implementation in Table 5 are those constructed by more than one bundle-based LFSRs, especially for $s = 4$, which has examples with one XOR gate for all $n \in \{4, 8, 16, 32, 64\}$. The case with 8 branches (i.e., $s = 8$) may suffer restrictions in practice because it does not have examples when $n = 4$ and $n = 8$.

Table 5. Comparison of our diffusion layers with those given in the LED, PHOTON hash family, and [13]

s	$A^{(s)}_{lfsr}$ $(d = s)$	Cost (XOR gates)	n=4	n=8	n=16	n=32	n=64	Note
4	$[L, 1, 1, L^2]$	$3n + \#L + \#L^2$	Y	Y	Y	Y	Y	Ours
	$[L^2, 1, L, L]$	$3n + 2\#L + \#L^2$	Y	Y	Y	Y	Y	LED [8]
	$[1, L, 1, 1 + L]$	$4n + 2\#L$	Y	Y	Y	Y	Y	[13]
5	$[1, L^2, L^{-1}, L^{-1}, L^2]$	$4n + 2(\#L^2 + \#L^{-1})$	Y	Y	Y	Y	Y	Ours
	$[1, L, L^3 + 1, L^3 + 1, L]$	$6n + 2\#L + 2\#L^3$	Y	Y	Y	Y	Y	P_{100} [9]
	$[1, L^2, 1, 1, L]$	$4n + \#L + \#L^2$	N	Y_F	Y	Y	Y	[13]
	$[1, L, 1, L, L + L^{-1}]$	Not perfect when $d = 5$!	-	-	-	-	-	[13]
6	$[1, L^{-2}, L^{-1}, L^2, L^{-1}, L^{-2}]$	$5n + \#L^2 + 2(\#L^{-2} + \#L^{-1})$	Y	Y	Y	Y	Y	Ours
	$[1, L, L^3, L^2 + 1, L^3, L]$	$6n + \#L + \#L^2 + 2\#L^3$	Y	Y	Y	Y	Y	P_{144} [9]
	$[L, L + 1, 1, L, 1, L^2]$	$6n + 2\#L + \#L^2$	N	Y_F	Y_F	Y	Y	P_{288} [9]
	$[1, L^2, L^2, L, L^2, 1 + L]$	Not perfect when $d = 6$!	-	-	-	-	-	[13]
	$[1, L, L^{-1}, 1 + L, L, L + L^{-1}]$	$7n + 2\#L^{-1} + 4\#L$	N	N	Y	Y	Y	[13]
7	$[1, L, L^{-5}, 1, 1, L^{-5}, L]$	$6n + 2(\#L + \#L^{-5})$	Y	Y	Y	Y	Y	Ours
	$[1, L^2, L^2 + L, 1, 1, L^2 + L, L^2]$	$8n + 2\#L + 4\#L^2$	Y	Y	Y	Y	Y	P_{196} [9]
	$[1, L^2, 1 + L^2, L, 1 + L^2, L^2, L^2]$	$8n + \#L + 5\#L^2$	N	N	Y_F	Y	Y	[13]
	$[1, L + L^{-1}, L, L^{-1}, L, L, L^{-1} + 1]$	Not perfect when $d = 7$!	-	-	-	-	-	[13]
8	$[1, L^{-3}, L, L^3, L^2, L^3, L, L^{-3}]$	$7n + \#L^2 + 2(\#L + \#L^3 + \#L^{-3})$	Y	Y	Y	Y	Y	Ours
	$[L, L^2, L, L^3 + L + 1, L, L^3, L^2 + 1, L^2 + L]$	$11n + 5\#L + 3\#L^2 + 2\#L^3$	Y	Y	Y	Y	Y	P_{256} [9]
	$[1, 1 + L^2, L, 1 + L, 1, L + L^2, L^2, L^2]$	$10n + 3\#L + 4\#L^2$	N	N	Y_F	Y	Y	[13]
	$[1, 1 + L, 1, L^{-1}, L^{-1} + 1, L, L, L^{-1} + L]$	Not perfect when $d = 8$!	-	-	-	-	-	[13]
	$A^{(s)}_{gfs}$ $(d = s)$							
4	$[L, 1, 1, L]$	$2n + 2\#L$	Y	Y	Y	Y	Y	Ours
6	$[L, 1, 1, L^2, L, L^2]$	$3n + 2\#L + 2\#L^2$	N	Y_F	Y_F	Y	Y	Ours
8	$[1, L^4, 1, L^{-1}, 1, L, 1, L^2]$	$4n + \#L^4 + \#L + \#L^{-1} + \#L^2$	N	N	Y_F	Y_F	Y	Ours

7 Other Recursive Diffusion Layers

In this section, we discuss some possible manners to construct new lightweight perfect diffusion layers using LFSRs.

7.1 Increase the Number of Iterations

We may obtain more lightweight perfect diffusion layers if the number of iterations is increased. However, the efficiency of getting the final diffusion layer D is decreased.

When constructing perfect diffusion layers with branch number 5 (i.e., $s = 4$) using one bundle-based LFSR (see Fig.1), we find some L_is may be chosen as the zero transformation if the number of iterations can be greater than s. An example we obtained is

$$A_{lfsr}^{(4)} = [1, L, 0, 0]. \tag{11}$$

It costs $n + \#L$ XOR gates in hardware implementation and needs 22 iterations to reach branch number 5, if the following eight matrices

$$\begin{array}{ccc} L, & L+1, & L^2+L+1, \\ L^3+L+1, & L^3+L^2+1, & L^4+L^3+1, \\ L^4+L^3+L^2+L+1, & L^5+L^4+L^3+L^2+1 & \end{array}$$

are non-singular.

We observe that all examples listed in Table 1 and the multiplication with a root of the irreducible polynomial $x^4 + x + 1$ for $n = 4$ fit the above eight conditions.

7.2 Bit-Level LFSRs

Diffusion layers discussed above are constructed by bundle-based LFSRs. An instinctive idea is to construct them using bit-level LFSR. That is, the updating unit in the LFSR is not bundle but bit now. In each step, only a few bits, for instance, the rightmost m bits, of LFSR are updated by the XOR values of chosen bit positions while other $sn-m$ bits are obtained by shifting the LFSR by m bits to the left. When considering bit-level LFSR, Theorem 4 will be directly used to judge whether a choice of LFSR is perfect after some iterations.

We search all possible LFSRs when $s = n = 4$ and $m \in \{1, 2\}$, aim to find perfect diffusion layers with branch 5 under 4-bit Sboxes. We denote by $x[1], x[2], \ldots, x[16]$ the 16 bits in the LFSR (see Fig.1), where $x[16]$ is the least significant (rightmost) bit.

– For $m = 1$, we do not find perfect diffusion layers. However, we find many almost perfect diffusion layers [10], that is, with differential branch number 4. They can be detected by a variant of Theorem 4, which will be introduced in the full version due to the lack of space. Although these diffusion layers do not reach the maximum branch number, they may still have some applications because they are extremely lightweight in hardware implementation. For example, one LFSR we find is: $y = x[1] \oplus x[7], x[i] = x[i+1]$ for $1 \leq i \leq 15, x[16] = y$, which only costs one XOR gate and needs 44 iterations to reach differential branch number 4. Another example is: $y = x[1] \oplus x[6] \oplus x[13], x[i] = x[i+1]$ for $1 \leq i \leq 15, x[16] = y$. It costs two XOR gates and needs 15 iterations to reach differential branch number 4.

– For $m = 2$, we find a list of perfect diffusion layers. One of the best LFSR is: $y = x[1] \oplus x[6] \oplus x[8] \oplus x[10] \oplus x[13]$, $z = x[2] \oplus x[5] \oplus x[7] \oplus x[10] \oplus x[11] \oplus x[13] \oplus x[14]$, $x[i] = x[i+2]$ for $1 \leq i \leq 14$, $x[15] = y$, $x[16] = z$. It costs 10 XOR gates and needs 8 iterations to reach branch number 5.

8 Conclusion

In this paper, we construct a list of lightweight perfect diffusion layers using LFSRs. On the one hand, we revisit the design strategy of PHOTON and [13], which constructs perfect diffusion layers using one bundle-based LFSR. Our proposals have smaller hardware implementations than those given in LED, PHOTON and [13]. They can be used to replace the diffusion layers in PHOTON to gain better performance. On the other hand, we extend the strategy to construct perfect diffusion layers using more than one bundle-based LFSRs. The structure we choose is the Type-II Generalized Feistel Structure. Finally, we discuss some possible manners to construct perfect diffusion layers by increasing the number of iterations and using bit-level LFSRs.

Since most of our proposals have low-cost examples which are consistent with 4-bit Sboxes and 8-bit Sboxes, we expect that they will be useful in designing (lightweight) block ciphers and (lightweight) hash functions.

Acknowledgements. We are grateful to the anonymous reviewers for their valuable comments on this paper. This work was partially supported by the 973 Program (Grant No. 2013CB834203) and the National Natural Science Foundation of China (Grant No. 60970134 and Grant No. 11171323).

References

1. Barreto, P.S.L.M., Rijmen, V.: The Anubis block cipher. NESSIE (September 2000), (primitive submitted) http://www.cryptonessie.org/
2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
3. Blaum, M., Roth, R.M.: On Lowest Density MDS Codes. IEEE Transactions on Information Theory 45(1), 46–59 (1999)
4. Brown, W.C.: Matrices over commutative Rings. Monographs and textbooks in pure and applied mathematics. Marcel Dekker, Inc. (1993)
5. Celler, F., Leedham-Green, C.R.: Calculating the Order of an Invertible Matrix. In: Finkelstein, L., Kantor, W.M. (eds.) Groups and Computation II. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, pp. 55–60. AMS (1997)

6. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES – The Advanced Encryption Standard. Springer (2002)
8. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
9. Guo, J., Peyrin, T., Poschmann, A.: The `PHOTON` Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
10. Kang, J., Hong, S., Lee, S., Yi, O., Park, C., Lim, J.: Practical and Provable Security Against Differential and Linear Cryptanalysis for Substitution-Permutation Networks. ETRI Journal 23(4), 158–167 (2001)
11. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland Publishing Company (1978)
12. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
13. Sajadieh, M., Dakhilalian, M., Mala, H., Sepehrdad, P.: Recursive Diffusion Layers for Block Ciphers and Hash Functions. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 385–401. Springer, Heidelberg (2012)
14. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
15. Zheng, Y., Matsumoto, T., Imai, H.: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, Heidelberg (1990)

A Condition Set of $A_{lfsr}^{(5)}$

$A_{lfsr}^{(5)} = [1, L^2, L^{-1}, L^{-1}, L^2]$ with $d = 5$ is perfect, if the following 21 matrices are non-singular.

$$\begin{gathered}
L, L+1, L^2+L+1, L^3+L+1, L^3+L^2+1, L^4+L^3+1, \\
L^4+L^3+L^2+L+1, L^5+L^2+1, L^5+L^3+1, L^5+L^3+L^2+L+1, \\
L^5+L^4+L^3+L+1, L^5+L^4+L^2+L+1, L^6+L^3+1, L^6+L^5+1, \\
L^6+L^5+L^4+L+1, L^6+L^4+L^3+L+1, \\
L^7+L^3+1, L^7+L^5+L^2+L+1, L^8+L^7+L^2+L+1, \\
L^{10}+L^6+L^5+L+1, L^{10}+L^9+L^8+L^6+L^4+L^2+1.
\end{gathered}$$