Finite Fields and Their Applications **8,** 256–267 (2002)
doi:10.1006/ffta.2001.0339, available online at http://www.idealibrary.com on IDEAL®

Efficient Linear Feedback Shift Registers with Maximal Period

Boaz Tsaban

Department of Mathematics, Bar-Ilan University, 52900 Ramat-Gan, Israel
E-mail: tsaban@macs.biu.ac.il

and

Uzi Vishne

Landau Research Center for Mathematical Analysis, Hebrew University of Jerusalem, Israel
E-mail: vishne@math.huji.ac.il

Communicated by Peter Jau-Shyong Shive

Received October 24, 2000; revised June 4, 2001

We introduce and analyze an efficient family of linear feedback shift registers (LFSR's) with maximal period. This family is word-oriented and is suitable for implementation in software, thus provides a solution to a recent challenge [8]. The classical theory of LFSR's is extended to provide efficient algorithms for generation of irreducible and primitive LFSR's of this new type. © 2002 Elsevier Science (USA)

Key Words: linear feedback shift registers; linear transformation shift registers; fast software encryption.

1. LINEAR FEEDBACK SHIFT REGISTERS

Linear feedback shift registers (*LFSR*'s) are fundamental primitives in the theory and practice of pseudorandom number generation and coding theory (see, e.g., [1–4,6,7], and references therein).

Figure 1 describes a typical LFSR over the two-element field $\mathbb{F}_2 = \{0, 1\}$, where each step consists of adding some of the state bits (we follow the convention that the elements of $\mathbb{F}_2$ are called *bits*), and the result is inserted to the register in a FIFO manner.

Such a construction is slow in the sense that it produces only one new bit per step. Moreover, it is difficult to implement in software, since many bit

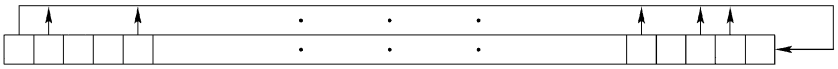

FIG. 1. A typical LFSR.

manipulations are required. In certain cases (but not always [10]), it is possible to use LFSR's with only two feedback taps. This makes a slightly faster LFSR. (See also Section 7.)

In the 1994 conference on fast software encryption, a challenge was set forth to design LFSR's which exploit the parallelism offered by the word oriented operations of modern processors [8, Section 2.2]. In this paper, we suggest a solution and study its properties.

2. LINEAR TRANSFORMATION SHIFT REGISTERS

Fix an arbitrary finite field F. A sequence $\sigma = \langle s_n \rangle_{n=0}^{\infty}$ of elements from F is *linear recurring with characteristic polynomial*

$$f(\lambda) = a_0 + a_1\lambda + \cdots + a_d\lambda^d \in F[x]$$

if $a_d = 1$, and

$$a_0 s_n + a_1 s_{n+1} + \cdots + a_d s_{n+d} = 0$$

for all $n = 0, 1, 2, \ldots$. The *minimal polynomial* of a linear recurring sequence σ is the characteristic polynomial of σ of least degree. Let σ be a nonzero linear recurring sequence with an irreducible characteristic polynomial $f(\lambda)$. It is well known (cf. [2]) that the period of σ is equal to the order of λ in the multiplicative group of the field $K = F[\lambda]/\langle f(\lambda)\rangle$. If λ generates the whole group, we say that $f(\lambda)$ is *primitive*. (In this case σ has the maximal possible period $|K| - 1 = |F|^d - 1$, where $d = \deg f(\lambda)$.) Likewise, for any natural number d, if T is a linear transformation of F^d and $v \in F^d$ is nonzero, then the sequence $\langle T^n(v)\rangle_{n=0}^{\infty}$ of vectors in F^d has period $|F|^d - 1$ if and only if the characteristic polynomial of the linear transformation T is primitive over $F[\lambda]$. If this is the case we say that T is *primitive*.

We now introduce the family of *linear transformation shift registers* (TSR's). For convenience of presentation, we pack $m \cdot n$-dimensional vectors in an array $(v_0, \ldots, v_{n-1})$ of n vectors in F^m (n and m will be fixed throughout the paper). In the intended application, $F = \mathbb{F}_2$ and m is the number of bits in the processor's word. Typical values of m are 8, 16, 24, 32, and 64. This way, the array $(v_0, \ldots, v_{n-1})$ is stored in n processor words. Following this interpretation, elements of F^m will be called *words*.

DEFINITION 2.1. Let T be a linear transformation of F^m, and let $S = \langle a_0, \ldots, a_{n-1} \rangle \in F^n$. A *TSR step* $\langle T, S \rangle$ of the array $R = (v_0, \ldots, v_{n-1}) \in M_{m \times n}(F)$ is the linear transformation

$$\langle T, S \rangle (R) := (v_1, v_2, \ldots, v_{n-1}, T(a_0 v_0 + a_1 v_1 + \cdots + a_{n-1} v_{n-1})).$$

The system $\langle T, S, R \rangle$ is called a *TSR*.

Figure 2 illustrates a typical example of a TSR. An obvious advantage over the standard LFSR is that here a whole new word (rather than a single bit) is produced per step.

Linear transformations on processor words can be performed very efficiently, either using lookup tables, or by using specific linear transformations which are efficient when working on processor words, e.g., Galois-type shift registers. The latter example has the advantage that no additional memory is required (see, e.g., [9, pp. 378–379]). Note further that choosing each of the a_i's to be either 0 or 1 eliminates the complexity of the multiplications $a_i v_i$. One cannot, however, eliminate the complexity of the transformation T as well by using the identity transformation $T = I$: In this

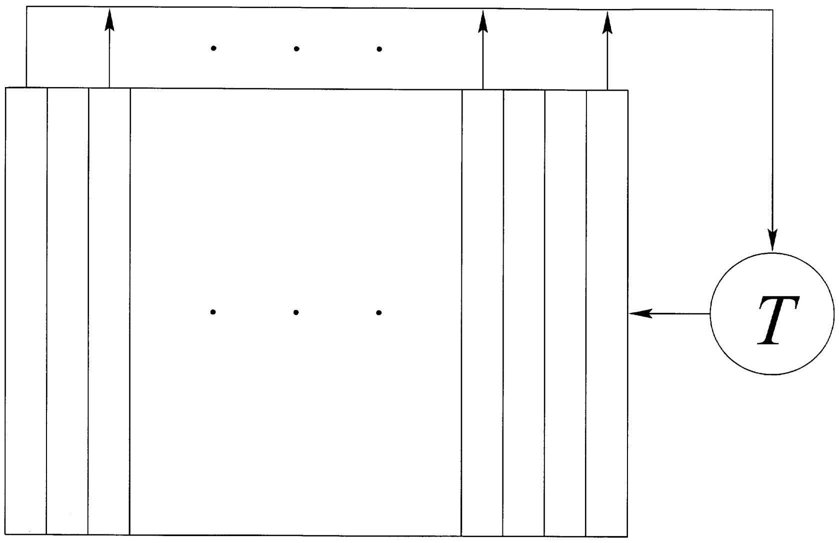

FIG. 2. A typical TSR.

case the period cannot be greater than $|F|^n - 1$, whereas in principle, memory of n words can yield period $|F|^{mn} - 1$.

Simulations show that there exist choices for T and S such that the resulted TSR step is primitive, and thus yields a sequence of vectors with period $2^{mn} - 1$. In the following sections we provide necessary conditions on T and S in order that the resulted TSR step is primitive. Choosing T and S to satisfy these conditions increases the probability that the resulted TSR is primitive with respect to random choice of these parameters. Thus, we will get an efficient algorithm for generation of primitive TSR's.

3. THE CHARACTERISTIC POLYNOMIAL OF A TSR

Identify the linear transformation T operating on words with the matrix $T \in M_m(F)$ such that $T \cdot v = T(v)$, $v \in F^m$.

Let I denote the $m \times m$ unit matrix. A TSR step $\langle T, S = \langle a_0, \ldots, a_{n-1}\rangle\rangle$ of the array $R = (v_0, \ldots, v_{n-1}) \in (F^m)^n$ is equivalent to multiplication of $(v_0, \ldots, v_{n-1})^t$ from the left by the block matrix $[\langle T, S\rangle] \in M_{nm}(F)$, where

$$[\langle T, S\rangle] = \begin{pmatrix} 0 & I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & I \\ a_0T & a_1T & \cdots & a_{n-2}T & a_{n-1}T \end{pmatrix}.$$

Let $f_S(\lambda) = a_0 + a_1\lambda + \cdots + a_{n-1}\lambda^{n-1}$ (so that the characteristic polynomial of $[\langle T, S\rangle]$ in the case $m = 1$ and $T = (1)$ is $\lambda^n - f_S(\lambda)$), and let $f_T(\lambda) = |\lambda I - T|$ denote the characteristic polynomial of T (note that the degree of $f_T(\lambda)$ is m.)

PROPOSITION 3.1. *Let T be a linear transformation of F^m, and $S = \langle a_0, \ldots, a_{n-1}\rangle \in F^n$. Then the characteristic polynomial of the TSR step $\langle T, S\rangle$ is*

$$f_{\langle T,S\rangle}(\lambda) = f_S(\lambda)^m \cdot f_T\left(\frac{\lambda^n}{f_S(\lambda)}\right).$$

Proof. We multiply each row block by λ, and add the result to the next one. Then we use the $-I$ blocks to cancel the terms in the first column block.

$$|\lambda I - \langle T, S\rangle| = \begin{vmatrix} \lambda I & -I & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda I & -I \\ -a_0 T & -a_1 T & \cdots & -a_{n-2} T & \lambda I - a_{n-1} T \end{vmatrix}$$

$$= \begin{vmatrix} \lambda I & -I & 0 & \cdots & 0 \\ \lambda^2 I & 0 & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \lambda^{n-1} I & 0 & \cdots & 0 & -I \\ -a_0 T & -a_1 T & \cdots & -a_{n-2} T & \lambda I - a_{n-1} T \end{vmatrix}$$

$$= \begin{vmatrix} 0 & -I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & -I \\ \lambda^n I - f_S(\lambda) T & -a_1 T & \cdots & -a_{n-2} T & \lambda I - a_{n-1} T \end{vmatrix}$$

$$= (-1)^{m(n-1)} \begin{vmatrix} \lambda^n I - f_S(\lambda) T & -a_1 T & \cdots & -a_{n-2} T & \lambda I - a_{n-1} T \\ 0 & -I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & -I \end{vmatrix}$$

$$= (-1)^{m(n-1)} \cdot |\lambda^n I - f_S(\lambda) T| \cdot |-I|^{n-1}$$

$$= f_S(\lambda)^m \cdot \left| \frac{\lambda^n}{f_S(\lambda)} I - T \right| = f_S(\lambda)^m \cdot f_T\left(\frac{\lambda^n}{f_S(\lambda)}\right). \qquad \blacksquare$$

A naive algorithm for generation of a TSR with maximal period would be to choose the linear transformation T and the set S at random, calculate the characteristic polynomial $f_{\langle T,S\rangle}(\lambda)$ using Proposition 3.1, and then check whether it is primitive, repeating this process until a primitive polynomial is found. In most of the cases, the polynomial will not be primitive for the reason that it is not even irreducible. The following corollary shows that much unnecessary work can be avoided.

COROLLARY 3.1. *If $f_T(\lambda)$ is reducible over F, then so is $f_{\langle T,S\rangle}(\lambda)$.*

Proof. Suppose $f_T(\lambda) = q_1(\lambda)q_2(\lambda)$ is a nontrivial factorization of $f_T(\lambda)$ over F, $m_i = \deg q_i(\lambda)$. Then $f_S(\lambda)^{m_i} q_i(\frac{\lambda^n}{f_S(\lambda)})$ are polynomials, and $f_{\langle T,S\rangle}(\lambda) = (f_S(\lambda)^{m_1} q_1(\frac{\lambda^n}{f_S(\lambda)})) \cdot (f_S(\lambda)^{m_2} q_1(\frac{\lambda^n}{f_S(\lambda)}))$ is a nontrivial factorization. ■

Remark 3.1. In general, the probability that a monic polynomial of degree m chosen at random is irreducible is close to $1/m$. Thus, by Corollary 3.1, the probability that $f_{\langle T,S\rangle}(\lambda)$ is irreducible provided that $f_T(\lambda)$ is irreducible should be about m times larger than the probability when $f_T(\lambda)$ is arbitrary.

4. IRREDUCIBILITY THROUGH EXTENSION FIELDS

The algorithm stated in the previous section considered polynomials of a special form as candidates to be primitive. In this section we study polynomials of this form, with the aim of improving the algorithm.

Let F be a fixed finite field. Let $q(\lambda) = q_0 + q_1\lambda + \cdots + q_m\lambda^m \in F[\lambda]$. We write $\mathfrak{p}_{q(\lambda)}(x, y)$ for the homogeneous polynomial

$$x^m \cdot q(y/x) = q_0x^m + q_1x^{m-1}y + \cdots + q_my^m.$$

We wish to find necessary conditions for polynomials of the form $\mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ to be irreducible. Clearly, if $g(\lambda), f(\lambda) \in F[\lambda]$ are not relatively prime, then the polynomial $\mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is reducible. Also, by Corollary 3.1, if $q(\lambda)$ is reducible, then so is $\mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$. We are thus interested in the following type of polynomials.

DEFINITION 4.1. We say that a polynomial

$$\mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda)) := g(\lambda)^{\deg q(\lambda)} \cdot q\left(\frac{f(\lambda)}{g(\lambda)}\right)$$

is a *candidate* if:

1. $g(\lambda), q(\lambda), f(\lambda) \in F[\lambda]$,
2. $f(\lambda)$ and $g(\lambda)$ are relatively prime, and
3. $q(\lambda)$ is monic and irreducible.

THEOREM 4.1. *Assume that $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is a candidate, and let α be a root of $q(\lambda)$ in the splitting field L of $q(\lambda)$. Then the number of distinct* *irreducible factors of $Q(\lambda)$ over F is equal to the number of distinct irreducible factors of $f(\lambda) - \alpha g(\lambda)$ over L.*

Proof. Denote by $\alpha_1, \dots, \alpha_m \in L$ the (distinct) roots of $q(\lambda)$ in L, so that $q(\lambda)$ has the form $\prod_{i=1}^{m} (\lambda - \alpha_i)$. We have that over L,

$$Q(\lambda) = g(\lambda)^m \cdot q\left(\frac{f(\lambda)}{g(\lambda)}\right) = \prod_{i=1}^{m} (f(\lambda) - \alpha_i g(\lambda)). \tag{1}$$

We can extend the standard norm map $L \to F$ to a norm $N_{L/F} : L \times [\lambda] \to F[\lambda]$ by $N_{L/F}(h(\lambda)) = \prod_{\sigma \in Gal(L/F)} \sigma(h(\lambda))$, where $\sigma(\lambda) = \lambda$ for all $\sigma \in Gal(L/F)$. Fix any $\alpha \in \{\alpha_1, \dots, \alpha_m\}$. Using this notation, Eq. (1) is

$$Q(\lambda) = N_{L/F}(f(\lambda) - \alpha g(\lambda)).$$

We will use the following lemma.

Lemma 4.1. *Let the field L be an extension of F. Assume that $r(\lambda) \in L[\lambda]$ is irreducible. Then $R(\lambda) = N_{L/F}(r(\lambda))$ is equal to an irreducible polynomial over F raised to the power $[L : L_0]$, where $L_0 \subseteq L$ is the subfield generated by the coefficients of $r(\lambda)$ over F.*

Proof. Since $N_{L/F} = N_{L_0/F} \circ N_{L/L_0}$ and $N_{L/L_0}(r(\lambda)) = r(\lambda)^{[L : L_0]}$, it is enough to prove the claim in the case $L_0 = L$.

Let $R(\lambda) = R_1(\lambda) \cdots R_t(\lambda)$ be an irreducible factorization of $R(\lambda)$ over F. Obviously $r(\lambda)$ divides $R(\lambda)$ in $L[\lambda]$, and since $r(\lambda)$ is irreducible we have that $r(\lambda)$ divides one of the factors, say $r(\lambda)$ divides $R_1(\lambda)$.

Let L_1 be the splitting field of $R_1(\lambda)$ over F. Note that $L \subseteq L_1$, since the coefficients of $r(\lambda)$ (which divides $R_1(\lambda)$) generate L. Let L_2 be the splitting field of $r(\lambda)$ over L, then $L_2 \subseteq L_1$ and $\deg R(\lambda) = [L : F] \cdot \deg r(\lambda) = [L_2 : F]$ divides $[L_1 : F] = \deg R_1(\lambda)$. Thus $R(\lambda) = R_1(\lambda)$ and is irreducible. ∎

Let

$$f(\lambda) - \alpha g(\lambda) = u_1(\lambda)^{s_1} \cdots u_t(\lambda)^{s_t}$$

be a factorization into irreducible polynomials over L.

Taking the norm from $L[\lambda]$ to $F[\lambda]$, we get the factorization

$$Q(\lambda) = U_1(\lambda)^{s_1} \cdots U_t(\lambda)^{s_t}$$

over F, where $U_i(\lambda) = N_{L/F}(u_i(\lambda))$. By Lemma 4.1, The polynomials $U_i(\lambda)$ are irreducible (the coefficient of $\lambda^{\deg g(\lambda)}$ in $f(\lambda) - \alpha g(\lambda)$ generates L). It thus remains to show that the $U_i(\lambda)$ are relatively prime. We will show that u_i is prime to $\sigma(u_j)$ for any $\sigma \in Gal(L/F)$ and $j \neq i$. Indeed, if $\sigma = 1$ then u_i is prime to u_j by the assumption. Otherwise, u_i divides $f - \alpha g$ and $\sigma(u_j)$ divides

$f - \sigma(\alpha)g$, but $f - \alpha g$ and $f - \sigma(\alpha)g$ are distinct and irreducible, thus relatively prime. ■

COROLLARY 4.1. *Assume that $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is a candidate, and let L be the splitting field of $q(\lambda)$. Let α be a root of $q(\lambda)$ in L. Then $Q(\lambda)$ is irreducible over F if, and only if, $f(\lambda) - \alpha g(\lambda)$ is irreducible over L.*

According to [5, Chapter 4], checking irreducibility of a degree d polynomial amounts to performing gauss elimination of a matrix of size $d \times d$. In a finite field F this requires roughly d^3 operations of multiplication and addition. Assume that $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is a candidate, and set $n = \max\{\deg f(\lambda), \deg g(\lambda)\}$. Checking the reducibility of $Q(\lambda)$ directly over F requires roughly $\deg Q(\lambda)^3 = m^3 n^3$ operations. Checking its reducibility via Corollary 4.1 requires roughly n^3 operations, but here multiplication is more expensive: each multiplication in L requires roughly m^2 multiplications in F. Thus, the algorithm implied by Corollary 4.1 is roughly m times faster, where $m = \deg q(\lambda)$. See also Remark 6.2.

5. PRIMITIVITY

Assume that $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is a candidate, L is the splitting field of $q(\lambda)$, and α is a root of $q(\lambda)$ in L. By Corollary 4.1, $Q(\lambda)$ is irreducible over F if, and only if, $f(\lambda) - \alpha g(\lambda)$ is irreducible over L. The analogue result for primitivity follows: $Q(\lambda)$ is primitive if, and only if, it is irreducible and its roots generate K^*, where K is the splitting field of $Q(\lambda)$. Now, observe that K is also the splitting field of $f(\lambda) - \alpha g(\lambda)$, and that $Q(\lambda)$ and $f(\lambda) - \alpha g(\lambda)$ share the same roots in K. This result, however, does not yield an improvement of the algorithm stated in the previous section.

In this section we show that if $f(0) = 0$ and the base field is $F = \mathbb{F}_2$ (these assumptions hold in the intended environment for the TSR), then a candidate $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is primitive only if $q(\lambda)$ is primitive. Thus, the TSR-generation algorithm should begin with *primitive* transformations T, yielding an additional speedup factor $\phi(|L^*|)/|L^*|$, which is roughly 2 when $\deg q(\lambda)$ is a power of 2, cf. [5].

It will be convenient to use the following definition.

DEFINITION 5.1. Let L be a finite field. The *index* of a nonzero element $\alpha \in L$ is the index $|L^*|/|\langle\alpha\rangle|$ of the cyclic group generated by α as a subgroup of L^*.

An irreducible polynomial is primitive if, and only if, its roots have index 1 in its splitting field. Note further that for d dividing $|L^*|$, $\alpha \in L$ has index d if, and only if, $\alpha = g^d$ for some generator g of the cyclic group L^*.

Lemma 5.1. *Let $h(\lambda) \in L[\lambda]$ be an irreducible monic polynomial of degree n over L, with splitting field K and a root μ. Then $\mu^{|K^*|/|L^*|} = (-1)^n h(0)$.*

Proof. Let $\mu_0, \ldots, \mu_{n-1}$ denote the (distinct) roots of $h(\lambda)$. Then $h(\lambda) = (\lambda - \mu_0) \cdots (\lambda - \mu_{n-1})$ is the factorization over K, thus $h(0) = (-1)^n \mu_0 \cdots \mu_{n-1}$. On the other hand, the Galois group of K/L is generated by the Frobenius automorphism $u \mapsto u^{|L|}$, thus the roots of $h(\lambda)$ are $\mu, \mu^{|L|}, \ldots, \mu^{|L|^{n-1}}$, and $\mu_0 \cdots \mu_{n-1} = \mu^{1+|L|+\cdots+|L|^{n-1}} = \mu^{\frac{|L|^n-1}{|L|-1}}$. ■

Theorem 5.1. *Assume that $F = \mathbb{F}_2$ and $Q(\lambda) = \mathfrak{p}_{q(\lambda)}(g(\lambda), f(\lambda))$ is an irreducible candidate with $f(0) = 0$. If $q(\lambda)$ is not primitive then $Q(\lambda)$ is not primitive.*

Proof. Let K be the splitting field of $Q(\lambda)$ over F, and $L \subseteq K$ the splitting field of $q(\lambda)$. Let $\mu \in K$ be a root of $Q(\lambda)$, and $\alpha \in L$ a root of $q(\lambda)$.

Let d_μ denote the index of μ in K, and d_α the index of α in L. We will show that $d_\alpha = (|L^*|, d_\mu)$. Thus, $d_\mu = 1$ implies $d_\alpha = 1$.

By Corollary 4.1, $h(\lambda) = f(\lambda) - \alpha g(\lambda)$ is irreducible over L. Since every polynomial is monic over $\mathbb{F}_2$, we can apply Lemma 5.1 to get that $h(0) = \mu^{|K^*|/|L^*|}$. But $h(0) = f(0) - (-1)^n \alpha g(0) = \alpha g(0)$. As $f(\lambda)$ and $g(\lambda)$ are relatively prime, $g(0) \neq 0$, thus $g(0) = 1$, and $h(0) = \alpha$.

Let g be a generator of K^* such that $\mu = g^{d_\mu}$.

Then $\alpha = \mu^{|K^*|/|L^*|} = g^{d_\mu |K^*|/|L^*|}$, and its order in K^* is

$$|K^*|/(|K^*|, d_\mu |K^*|/|L^*|) = |L^*|/(|L^*|, d_\mu),$$

as asserted. ■

6. THE FINAL GENERATION ALGORITHM

In light of the results obtained in the previous sections, we end up with the following algorithm for TSR-generation over $F = \mathbb{F}_2$:

Algorithm 1 (Primitive TSR generation).

1. Choose at random a primitive transformation T on $\mathbb{F}_2^m$.
2. Choose a random sequence $S = \langle a_0, \ldots, a_{n-1} \rangle \in \mathbb{F}_2^n$ such that $a_0 \neq 0$.
3. Choose a root α of $f_T(\lambda)$ in its splitting field L.
4. Check that $\lambda^n - \alpha f_S(\lambda)$ is irreducible over L (otherwise return to step 1).
5. Check that $Q(\lambda) = \mathfrak{p}_{f_T(\lambda)}(f_S(\lambda), \lambda^n)$ is primitive: Choose a root μ of $Q(\lambda)$ in its splitting field K, and check for all prime p dividing $|K^*|$ that $\mu^{|K^*|/p} \neq 1$ (in fact, as we show below, it is not needed to consider the cases where p divides $|L^*|$). If $Q(\lambda)$ is not primitive, return to step 1.

Remark 6.1. Assuming that generally the probability that $Q(\lambda) = \mathfrak{p}_{f_T(\lambda)}(f_S(\lambda), \lambda^n)$ is primitive is roughly the same for every primitive transformation T, it would be more efficient to repeat steps 2 to 5 of the algorithm several times before starting again from step 1. Thus, the complexity of step 1 will be negligible with respect to the total running time. Moreover, we argue below that step 5 usually occurs only once.

Remark 6.2. In all of the mentioned algorithms, one can get a speedup factor of $\tilde{m}$, where $\tilde{m}$ is the size of the word in the processor where the search for the TSR is made (note that this need not be the same processor on which the TSR will be implemented, thus $\tilde{m}$ need not be equal to m). This is done by exploiting the processors word-oriented operations to define parallel versions of the basic operations used in the algorithms.

For a natural number n, we denote by C_n the (multiplicative) cyclic group of order n. If g is a generator of C_n, then g^x is a generator as well if, and only if, $(x, n) = 1$. This is why the number of generators of C_n is exactly $\phi(n)$, where ϕ is Euler's function, and the probability that a uniformly chosen element generates C_n is $\phi(n)/n$. An irreducible polynomial $Q(\lambda)$ is primitive if a root μ of $Q(\lambda)$ generates the multiplicative group of its splitting field K. There is a natural 1 to $[K : F]$ correspondence between irreducible monic polynomials of degree $[K : F]$ and elements of K which do not belong to a proper subfield of K. This correspondence implies that the probability that an irreducible $Q(\lambda)$ is primitive is close to $\phi(|K^*|)/|K^*|$.

We now consider *irreducible candidates*. We wish to estimate the probability that a candidate passing the test in step 4 of the algorithm will also pass the final test of step 5. A candidate $Q(\lambda) = \mathfrak{p}_{f_T(\lambda)}(f_S(\lambda), \lambda^n)$ is *good* if T is primitive and $Q(\lambda)$ is irreducible. We will find, heuristically, the probability that a good candidate is primitive. Let L be the splitting field of $f_T(\lambda)$, and K be the splitting field of $Q(\lambda)$. Factor $|K^*| = k_L \cdot a$, where k_L is the product of all the prime factors of $|K^*|$ which divide $|L^*|$ (allowing powers of primes). Then the group K^* is isomorphic to $C_{k_L} \times C_a$, where a prime p divides $|L^*|$ if, and only if, it divides k_L. A root μ of $Q(\lambda)$ generates K^* if, and only if, its projections in C_{k_L} and C_a are both generators.

In the proof of Theorem 5.1 we showed that d_α, the index of a root α of $f_T(\lambda)$ in L, is equal to $(|L^*|, d_\mu)$, where d_μ is the index of μ in K. As T is primitive (i.e., $d_\alpha = 1$), we have that d_μ is prime to $|L^*|$. Thus, d_μ is prime to k_L, that is, k_L divides the order of μ in K^*. Therefore, the projection of μ in C_{k_L} is a generator of that group. We assume, *heuristically*, that the projection of μ on C_a is (close to being) uniformly distributed. Thus, the probability of its being a generator of C_a is close to $\phi(a)/a$.

In general,

$$\frac{\phi(n)}{n} = \prod_{p|n}\left(1 - \frac{1}{p}\right),$$

and as a prime p divides a if, and only if, p divides $|K^*|$ but not $|L^*|$, we have that

$$\frac{\phi(a)}{a} = \frac{\phi(|K^*|)/|K^*|}{\phi(|L^*|)/|L^*|}.$$

We thus have a heuristic justification for the following claim.

Claim 6.1. Assume that $Q(\lambda) = \mathfrak{p}_{f_T(\lambda)}(f_S(\lambda), \lambda^n)$ is an irreducible candidate over $\mathbb{F}_2$, where $f_T(\lambda)$ is primitive. Then the probability that $Q(\lambda)$ is primitive is close to

$$\frac{\phi(|K^*|)/|K^*|}{\phi(|L^*|)/|L^*|}.$$

Example 6.1. The probability at Claim 6.1 is usually close to 1. We give here a few examples:

1. When the word's size is 8 bits and the number of words is 7, we have that $\phi(2^{56} - 1)/(2^{56} - 1) \approx 0.465$, $\phi(2^8 - 1)/(2^8 - 1) \approx 0.502$, and the division yields probability close to 0.927.
2. When the word's size is 16 bits and the number of words is 4, we get probability close to 0.998.
3. For values 24 and 3, respectively, we get 0.898.
4. For values 32 and 2 we get 0.998.

7. CONCLUDING REMARKS

We have presented the family of linear transformation shift registers which is efficient in software implementations. The theory we developed enabled us to get an efficient algorithm for generation of primitive transformations of this type (i.e., which have maximal period), thus answering a challenge raised in [8].

Variants of our construction can be found more appropriate for certain applications. Arguments similar to those we have presented here may be found useful in the study of these variants as well. A noteworthy variant of the LFSR type that we have studied is the *internal-xor*, or *Galois*, shift register (see, e.g., [9]). The number of new bits generated in one step of an internal-xor shift register is equal on average to half of the number of taps in that LFSR. Our construction suggests an obvious analogue internal-xor TSR. We get exactly the same results for this case, since the characteristic polynomial of an internal-xor TSR is equal to that of the corresponding external-xor TSR, which we have studied in this paper.

REFERENCES

1. K. Cattell and J. C. Muzio, An explicit similarity transform between cellular automata and LFSR matrices, *Finite Fields Appl.* **4** (1998), 239–251.
2. S. W. Golomb, "Shift Register Sequences," Holden-Day, San Francisco, 1967.
3. R. Göttfert, H. Niederreiter, On the minimal polynomial of the product of linear recurring sequences, *Finite Fields Appl.* **1** (1995), 204–218.
4. N. Kamiya, On multisequence shift register synthesis and generalized-minimum-distance decoding of Reed-Solomon codes, *Finite Fields Appl.* **1** (1995), 440–457.
5. R. Lidl and H. Niederreiter, Finite fields, *in* "Encyclopedia of Mathematics and Its Applications," **20**, Cambridge University Press, Cambridge, UK, 1983.
6. A. Munemasa, Orthogonal arrays, primitive trinomials, and shift-register sequences, *Finite Fields Appl.* **4** (1998), 252–260.
7. G. H. Norton, On shortest linear recurrences, *J. Symbolic Comput.* **27** (1999), 325–349.
8. B. Preneel, Introduction, *in* "Proceedings of the Fast Software Encryption 1994 Workshop" (Bart Preneel, Ed.), Lecture Notes in Computer Science, Vol. 1008, pp. 1–5, 1995.
9. B. Schneier, "Applied Cryptography," Wiley, New York, 1996.
10. U. Vishne, Factorization of trinomials over Galois fields of characteristic 2, *Finite Fields Appl.* **3** (1997), 370–377.