Simple construction of ϵ-biased distribution

Long Hoang Nguyen and Andrew William Roscoe

Oxford University Department of Computer Science
Parks Road, Oxford, OX1 3QD
Email: {Long.Nguyen,Bill.Roscoe}@cs.ox.ac.uk

Abstract

ϵ-biased distribution has many applications in practice, including universal hashing computation. In this paper we will improve an existing ϵ-biased distribution construction due to Alon et al. [2] that requires to uniformly and efficiently sample irreducible polynomials of a large degree, e.g. between 80 and 160. To remove the need for such a sampling which can be computationally expensive, we will replace the irreducible polynomials by random monic polynomials of higher degree, i.e. every degree r monic polynomial whether irreducible or reducible is selected with the same probability 2^{-r}. To analyse the security of the scheme, we need to find the maximum number of degree r polynomials that divide a degree n polynomial where $n > r$. We end this paper by pointing out another use of our result in the fundamental problems of identity polynomial and primality testing in number theory

1 Introduction

A linear feedback shift register or LFSR is rarely used on its own in cryptographic applications because of its linearity property or low linear complexity that can be exploited by the attacker. Consequently there are two approaches to get around this problem to date. The first strategy is to increase the linear complexity of a LFSR-based pseudorandom number generator by introducing non-linear operators such as AND into the computation of the PRNG or combining the outputs of multiple LFSRs by a non-linear function. This method is often combined with extending the length of LFSRs from 80 to say 19937 bits as in the Mersenne Twister. Although such a PRNG can still be implemented very fast, it does not completely eliminate the linearity property, and hence still cannot be used for cryptographic applications.

The second approach is to randomise not only the initial seed but also the feedback rule of a LFSR, where the latter is also known as the characteristic polynomial of the LFSR. To our knowledge, this method has been used by Alon et al. [2] to produce an ϵ-biased distribution. There is however one major drawback of Alon et al.'s construction which is explained as follows: their construction randomly pick only irreducible monic polynomial of degree r as the characteristic polynomial because: (1) each irreducible polynomial often corresponds to a long-period LFSR; and (2) it is easier to analyse the security of the scheme. We observe that when r is small this construction is very efficient because it is feasible to store all irreducible polynomials of degree r in the memory, i.e. there are no more than $\lfloor 2^r/r \rfloor$ of them. The problem is that when $r \geq 80$, which is usually the case in practice, it is not feasible to do so, and thus if the same pseudorandom sequence must be produced by two or more parties at

the same time then they will need to agree in advance on the same algorithm that can return a unique and random irreducible monic polynomial of degree r on fly. Such a mechanism might not be always available in practice.

What we propose to investigate in this paper is that instead of randomly selecting only irreducible polynomials, we want every polynomial of degree r (i.e. both irreducible and reducible) is randomly picked with the same probability $1/2^r$. This will solve the problem of mapping each random number or seed to a unique feedback rule of a LFSR efficiently. But intuitively to achieve the same level of security, e.g. a bias of 2^{-80} as required in many cryptographic applications such as the Toeplitz-based universal hash function of Krawczyk [6] discussed in Section 4, we would need to choose a random polynomial of a significant higher degree compared to one that is based on only irreducible polynomials.

It turns out that the security of our scheme is closely related to the solution of the following problem: given integers r and n where $n > r$, what is the maximum number of degree r polynomials that can divide a degree n polynomial? Although this problem might look like it has been studied and solved by number theorists a long time ago, to the best of our knowledge there has not been any satisfactory solution for this problem in the literature. What we have discovered is that this problem has applications in not only biased distribution constructions but also the fundamental problems of identity polynomial and primality testing as recently demonstrated by Agrawal and Biswas [1]. These authors however provide a very weak analysis for this problem as discussed in their paper and in Section 5 of this paper. In section 3 we will use some number theory facts in combination with computer programming to give an estimated but highly accurate solution for this problem as demonstrated by our experimental results.

2 Notations

Definition 1. [2, 6] Let S be a distribution of sequences of length l. Let $(\alpha, s)_2$ denote the inner product modulo 2 of $\alpha \in (\mathbb{F}_2)^l$ and $s \in (\mathbb{F}_2)^l$.

- S is said to pass the linear test $\alpha \in (\mathbb{F}_2)^l$ with bias e if $|\Pr_{\{s \in S\}}[(\alpha, s)_2 = 1] - 1/2| \leq e$.
- S is said to be an e-biased distribution if it passes all linear tests $\alpha \neq 0$ with bias e.

3 Randomised LFSR

We consider only polynomials over GF(2). Any $(M-1)$-bit message m can be represented by a polynomial $m(x)$ of degree $\deg(m(x)) = M - 1$, where function $\deg(X)$ returns the degree of polynomial X.

$$m(x) = m_0 + m_1 x + \cdots + m_{M-1} x^{M-1}$$

If we want to construct a distribution of sequences of length $M - 1$ from a source of $2r$ truly random bits then our algorithm is as follows:

- The first r bits determine the coefficients of a degree r monic polynomial $f(x)$ that is the feedback rule of the resulting LFSR.
- The second r random bits form the initial state of the LFSR.

Using existing results proved in [2] (Proposition 1), the value of the bias e of this distribution is calculated as the ratio between the maximum number of monic polynomials of degree r that divide a polynomial of degree $M-1$ and the total number of degree r monic polynomials.[1]

$$e = \frac{\text{MAX}_{m \in \{0,1\}^{M-1}} \sharp\{f(x) | \deg(f(x)) = r \text{ and } f(x) | m(x)\}}{2^r}$$

We know that the number of irreducible monic polynomials of degree i is

$$\frac{1}{i} \sum_{d|i} \mu\left(\frac{i}{d}\right) 2^d$$

This expression is well approximated by $2^i/i$, and hence we will treat the number of irreducible monic polynomials of degree i as if it is exactly $\lfloor 2^i/i \rfloor$.

Calculating the bias e is very simple when the feedback rule $f(x)$ is an irreducible polynomial because any degree $M-1$ polynomial will have at most $\lfloor (M-1)/r \rfloor$ distinct irreducible factors of degree r. This is however not the case when any $f(x)$ is randomly selected as in our scheme.

Without loss of generality, we can assume that

$$\begin{aligned} M-1 &= 1\lfloor 2^1/1 \rfloor + 2\lfloor 2^2/2 \rfloor + 3\lfloor 2^3/3 \rfloor + \cdots + t\lfloor 2^t/t \rfloor \\ &\leq 2^1 + 2^2 + 2^3 + \cdots + 2^t = 2^{t+1} - 1 \end{aligned} \quad (1)$$

The product of all distinct irreducible polynomials of degree $i \leq t$ is therefore a polynomial of degree $M-1$, and we refer to such a polynomial as $m_1(x)$. We also define P_t the set of all distinct irreducible polynomials of degree $i \leq t$, and hence

$$m_1(x) = \prod_{g(x) \in P_t} g(x)$$

Suppose that

$$m(x) = g(x)h(x)$$

where $g(x)$ is an irreducible monic polynomial of degree greater than t then thanks to Equality 1 it is always possible to construct another polynomial $g'(x)$ where $\deg(g(x)) = \deg(g'(x))$ and $g'(x) = g_1(x)g_2(x)\cdots g_l(x)$ is a product of some l distinct irreducible polynomials from P_t and none of which divides $h(x)$. In other words, $m'(x) = g'(x)h(x)$ will have more factors as monic polynomials of degree r than $m(x)$. What this argument shows is that a polynomial $m(x)$ of degree $M-1$ will have more distinct factors as monic polynomials of degree r when $m(x)$ is the product of only irreducible monic polynomials of degree less than or equal to t. Consequently, $m_1(x)$ as defined above potentially gives us a tight lower bound for the worst-case representation of a message of $M-1$ bits.

We next will locate the upper bound for the worst-case representation of a message of $M-1$ bits. We observe that for any irreducible monic polynomial $g(x)$ of degree $i \leq t$,

[1] More information about this result can be found in Theorem 2 of Annex A of this paper or in [2], but it can be briefly explained as follows. Suppose that degree r polynomial $f(x)$ represents the feedback rule of a r-bit LFSR and k is a the $(M-1)$-bit stream outputted from the LFSR initiated by a r-bit seed, then there are two possibilities: (1) when $f(x)$ divides $m(x)$ then the inner product modulo 2 of m and k is zero regardless of what the initial r-bit seed is; and (2) when $f(x)$ does not divide $m(x)$ then the inner product modulo 2 of m and k is a linear combination of the r bits of the initial seed, and thus is non-zero. Hence the bias of this distribution equals the probability that the polynomial $f(x)$ divides the polynomial $m(x)$.

it is no better that $g(x)$ occurs more than $\lfloor r/i \rfloor$ times in the prime decomposition of a polynomial of degree $M-1$. As a consequence, we will refer to $m_2(x)$ as the product of every irreducible polynomial $g(x) \in P_t$ and each $g(x)$ appears exactly $\lfloor r/\deg(g(x)) \rfloor$ times in the prime decomposition of $m_2(x)$.

$$m_2(x) = \prod_{g(x) \in P_t} g(x)^{\lfloor r/\deg(g(x)) \rfloor}$$

We therefore arrive at

$$\begin{aligned} \deg(m_2(x)) &= \sum_{i \leq t} \lfloor r/i \rfloor i \lfloor 2^i/i \rfloor \\ &> \lfloor r/t \rfloor \deg(m_1(x)) = M-1 \end{aligned}$$

Since the degree of $m_2(x)$ is greater than $M-1$, we have gone over the top to define an upper bound for the worst-case representation of a message of $M-1$ bits. In the next subsection, we will demonstrate that our lower and upper bounds corresponding to $m_1(x)$ and $m_2(x)$ significantly narrow down the range where the worst-case scenario lies.

3.1 Experiments

Since there are many ways a degree r polynomial $f(x)$ can be constructed as a product of some of the irreducible factors from P_t, we have not been able to find a formula that can give us the exact number of degree r polynomials that divide either $m_1(x)$ or $m_2(x)$, but fortunately these numbers can be computed exhaustively by computers and these task can be done very quickly. In the following, we give the pseudo-code of the algorithms which can compute the numbers.

M1(r,t) returns the number of monic polynomials of degree r that divide $m_1(x)$
M1(r,t)
1. if$(r=0)$ then return 1
2. else if$(t=0$ and $r=1)$ then return 2
3. else if$(t=0$ and $r=2)$ then return 1
4. else if$(t=0$ and $r>2)$ then return 0
5. else if$(t>0)$ then
6. $L = \lfloor 2^t/t \rfloor$
7. $l = min(r/t, L)$
8. $sum = 0$
9. for $i=0$ to l
10. $sum = sum + \binom{L}{i} * \text{M1}(r - i*t, t-1)$
11. return sum

M2(r,t) returns the number of monic polynomials of degree r that divide $m_2(x)$
M2(r,t) 1. if($r=0$ or $t=0$) then return 1 2. else 3. $L=\lfloor 2^t/t \rfloor$ 4. $l=r/t$ 5. $sum=0$ 6. for $i=0$ to l 7. if($t\neq 1$ or $r=i*t$) then $sum=sum+$ M2'$(L,i)*$ M2$(r-i*t,t-1)$ 8. return sum
M2'(L,i) 1. if($i=0$) then return 1 2. else 3. $z=min(L,i)$ 4. $sum=0$ 5. for $j=0$ to z 6. $sum=sum+\binom{L}{j}*$ M2'$(j,i-j)$ 7. return sum

In the first experiment, we fix $M=2^{t+1}-1$ or the length of input message and let the LFSR bitlength r vary from 0 to 700. This experiment is repeated four times with different values for t: 10, 20, 30 and 40. We plot $\log(1/e)$ against r in the graph of Figure 1.

In the second experiment, we fix the bias e and let the message length vary from 2^{10} to 2^{60} bits. This experiment is repeated twice with different values for e: 2^{-80} and 2^{-40}. We plot the LFSR bitlength r against t in the graph of Figure 2.

4 Toeplitz matrix based construction

The following universal hashing construction is invented independently by Krawczyk [6] and Mansour et al. [8]. Since this construction uses a Toeplitz matrix multiplication, we give the definition for a Toeplitz matrix below.

Definition 2. A Toeplitz matrix A is a (not necessary square) matrix where each left-to-right diagonal is fixed, i.e. for all pairs of indexes (i,j): $A_{i,j}=A_{i+1,j+1}$.

If we want to compute a b-bit universal hash of a M-bit message m, then $(M+b-1)$-bit key k is drawn randomly from a distribution $R=\{0,1\}^{M+b-1}$. We can generate a Toeplitz matrix $A(k)$ of M rows and b columns from key k, i.e. we assume a linear map from $(\mathbb{F}_2)^{M+b-1}$ to the set of Toeplitz matrices in $(\mathbb{F}_2)^{M\times b}$, and then

$$h(k,m) \quad = \quad m \times A(k) \tag{2}$$

The symbol '$\times$' in Equation 2 represents a product of vector m and matrix $A(k)$ over $\mathbb{F}_2$.

If key k is drawn randomly and uniformly from R, then the collision probability is $\epsilon_c=2^{-b}$. Unfortunately such a random and uniform distribution is not always feasible to construct when $M+b-1$ is of a significant size. In practice for large messages we need to expand a standard key into a long $(M+b-1)$-bit key stream efficiently, and one therefore frequently uses a pseudorandom number generator. In the case of this construction, a linear pseudorandom

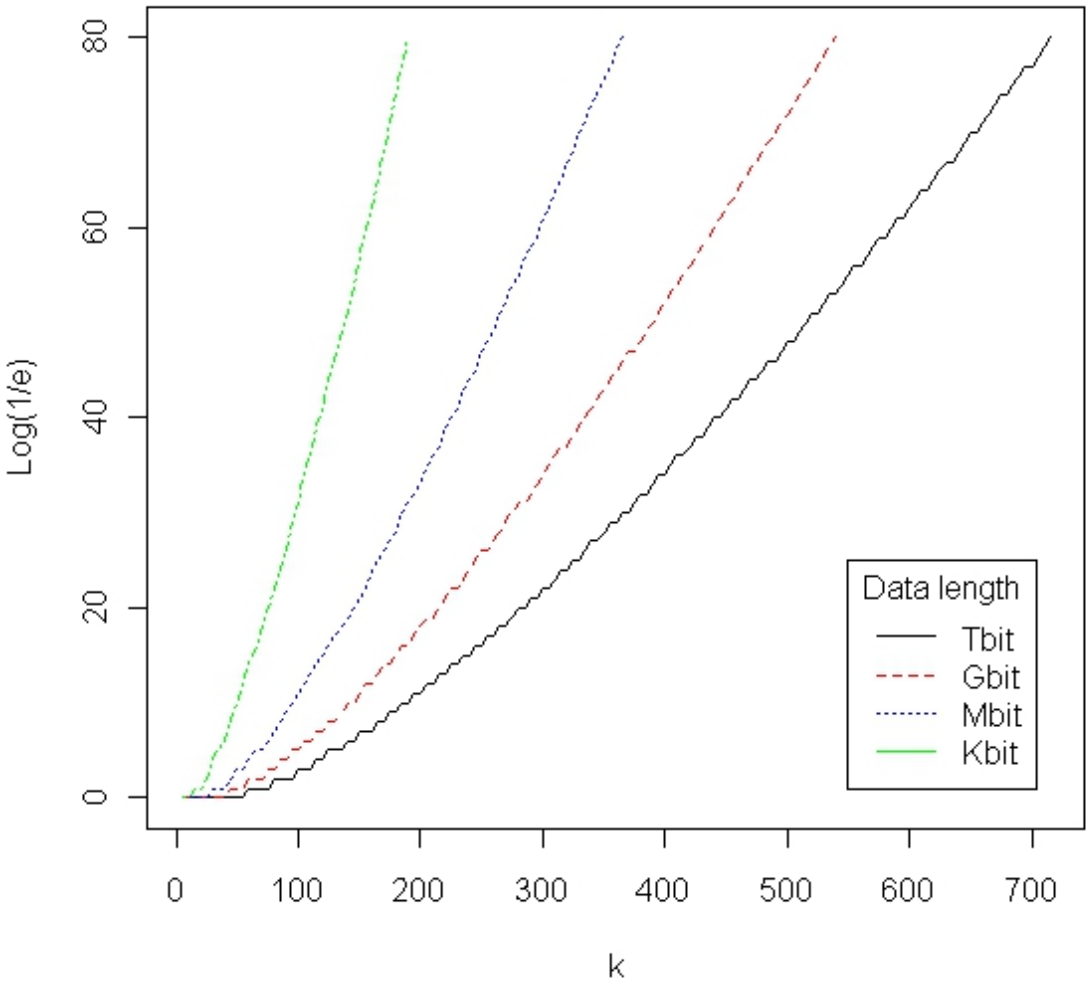
Log(1/e)
0
20
40
60
80
0
100
200
300
400
500
600
700
k
Data length
Tbit
Gbit
Mbit
Kbit

Figure 1:

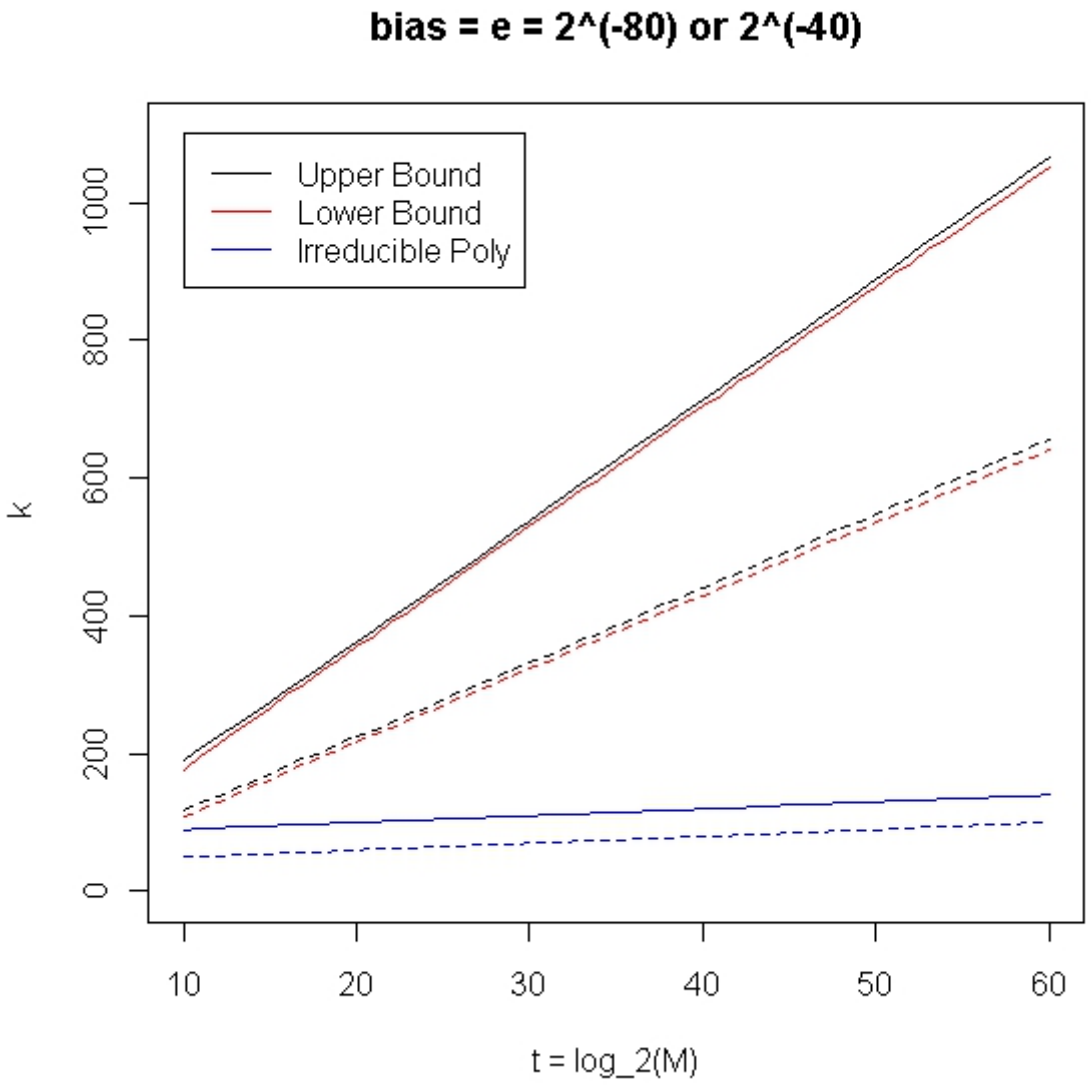
bias = e = 2^(-80) or 2^(-40)
Upper Bound
Lower Bound
Irreducible Poly
k
0
200
400
600
800
1000
10
20
30
40
50
60
t = log_2(M)

Figure 2:

number generator or one with a low linearity complexity must not be used because this construction is *linear* in the key, i.e. $h(k_1 + k_2, m) = h(k_1, m) + h(k_2, m)$. To justify this argument, in Annex A, we describe an attack on this construction if key expansion is done by a linear feedback shift register (or LFSR) whose feedback rule is fixed and known to an attacker.

We note that Krawczyk [6] suggested that k could be drawn from a biased distribution on sequences of length $K = M + b - 1$. Such a biased distribution can be generated by the scheme of Alon et al. [2][2] that requires to uniformly and efficiently sample irreducible monic polynomial of degree k. As we have explained earlier that this sampling can be very inefficient, and thus we propose to replace the irreducible polynomials by random monic polynomial as explained earlier.

Theorem 1. Let R be an ϵ-biased distribution on sequences of length K, then the Toeplitz matrix based scheme $(2^{-b} + \epsilon)$-almost universal.

Proof of this theorem can be found the paper of Krawczyk [6] (Theorem 13 and Theorem 5).

5 Identity polynomial testing

A restricted but common form of the identity polynomial testing problem is: given a degree n univariate binary polynomial $m(x)$, check if $m(x)$ is identically zero. This task is trivial if $m(x)$ is explicitly given as a sum of monomials, however in many cases $m(x)$ is given in an implicit form, including a product of many polynomials or an arithmetic circuit.

In the following we introduce a new and very simple algorithm that can solve this problem efficiently. Given a binary polynomial $m(x)$ of degree n, or if we are given an arithmetic circuit of size s then the corresponding polynomial degree n is bounded by 2^s, we randomly choose a monic polynomial $f(x)$ of degree r. We accept if $m(x)$ is divisible by $f(x)$.

If $m(x)$ is zero then the algorithm always accepts. If $m(x)$ is non-zero then the probability of false acceptance is at most the ratio between the maximum number of degree r polynomials that can divide a degree n polynomial and 2^r, which is the problem we solve in Section 3. Obviously this algorithm requires r random bits and the degree of the factor polynomial $f(x)$ is also r. We observe that the bigger r is the less likely we have a false acceptance when $m(x)$ is non-zero. The probability of false acceptance can be estimated by our approximated solution given in Figure 1 of Section 3.

Previously Agrawal and Biswas [1] introduced a very similar algorithm but theirs suffers from two major weaknesses: (1) the degree r of the randomly selected factor or polynomial is restricted to $\log n$ which leads to a very big probability of false-acceptance, in contrast the degree r is unrestricted in ours; and (2) in the analysis of Agrawal and Biswas, they do not consider the random polynomial $f(x)$ with multiple irreducible factors of degree less than r or with irreducible factors of degree less than $r/2 + 1$ as in ours, and thus their analysis will not give good estimated solution even when they let the degree of $f(x)$ be comparable to ours.

Being aware of these weaknesses, Agrawal and Biswas introduced a more complicated algorithm which randomly selects the factor polynomial $f(x)$ from a set of nearly co-prime polynomials. While this algorithm only needs $\log n$ random bits, the degree of the factor

[2]Other schemes introduced by Alon et al. [2] would involve the computation of either a Legendre symbol or a modular exponentiation per each pseudorandom bit, both of which are computationally expensive.

polynomial $f(x)$ is at least $\log^3 n$ and thus is much more expensive to check for divisibility than ours.

In this section we focus on testing whether a univariate polynomial over Z_2 is zero, but of course we are also interested in the case where the input polynomial is not only multivariate but also its coefficients are over Z_p where p is prime. Using the strategy introduced by Agrawal and Biswas any multivariate polynomial can be deterministically transformed to a univariate higher degree polynomial such that the resulting polynomial is zero iff the original one is zero. Secondly our analysis for binary polynomial of Section 3 can be easily adjusted to deal with polynomial over Z_p. From the distribution for irreducible polynomials [7], the number of irreducible polynomials of degree r over Z_q is

$$\frac{1}{r}\sum_{d|r}\mu\left(\frac{r}{d}\right)q^d$$

which is bounded above by $\frac{p^r}{r} - p^{r/2}$. Using this information, our algorithm and program presented in Section 3 can easily estimate the probability of false acceptance.

References

[1] M. Agrawal and S. Biswas. *Primality and Identity Testing via Chinese Remaindering.* FOCS'99.

[2] N. Alon, O. Goldreich, J. Hastad and R. Peralta. *Simple Constructions of Almost k-wise Independent Random Variables.* In Proceedings of the IEEE Symposium on Foundations of Computer Science, 1990, vol. 2, pp. 544-553.

[3] J.L. Carter and M.N. Wegman. *Universal Classes of Hash Functions.* Journal of Computer and System Sciences, 18 (1979), 143-154.

[4] S.W. Golomb. *Shift Register Sequences.* ISBN: 0894120484, Aegean Park Press, 1981.

[5] H. Krawczyk. *LFSR-based Hashing and Authentication.* Advances in Cryptology, CRYPTO 1994, LNCS vol. 839, 129-139.

[6] H. Krawczyk. *New Hash Functions For Message Authentication.* Advances in Cryptology - Eurocrypt 1995, LNCS vol. 921, pp. 301-310.

[7] Lidl and H. Niederrieter. *Introduction to finite fields and their applications.* Cambridge University Press, 1986.

[8] Y. Mansour, N. Nisan and P. Tiwari. *The Computational Complexity of Universal Hashing.* Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pp. 235-243, 1990.

[9] M. Matsumoto and T. Nishimura. Fast Mersenne Twister Homepage where the publicly available source code for this pseudorandom number generator can be found: `http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/SFMT/`

[10] M.N. Wegman and J.L. Carter. *New Hash Functions and Their Use in Authentication and Set Equality.* Journal of Computer and System Sciences, 22 (1981), 265-279.

A An attack on LFSR-based $h(k, m)$ of subsection 4

Let $(m, m')_2$ denote the inner product modulo 2 of m and $m' \in (\mathbb{F}_2)^M$.

Suppose that $h(k, m)$ of subsection 4 uses a r-bit LFSR to generate a $(K = M + b - 1)$-bit key $k' = k_0 \cdots k_{K-1}$ out of the r-bit key $k = k_0 k_1 \cdots k_{r-1}$. In this attack, we need to assume that the feedback rule of the LFSR is fixed and known to the intruder, such a feedback rule can be represented by an irreducible monic polynomial.

$$f(x) \quad = \quad f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots + f_{r-1} x^{r-1} + x^r \tag{3}$$

For any $n \in [r, K-1]$ we then have

$$k_n \quad = \quad f_0 k_{n-r} + f_1 k_{n-r+1} + \cdots + f_{r-1} k_{n-1} = \sum_{i=0}^{r-1} f_i k_{n-r+i} \tag{4}$$

Since we need to generate a $M \times b$ Toeplitz matrix $A(k)$ from key k, we use the following linear map from $(\mathbb{F}_2)^{M+b-1}$ to the set of Toeplitz matrices in $(\mathbb{F}_2)^{M \times b}$: $A_{i,j}(k) = k'_{i+j}$. In other words, each column (out of b columns) of $A(k)$ is a continuous sequence of M bits taken from k', and each column $i+1$ is shifted (upward) relative to the column i, with a new element set to the last position of the column $i+1$. This is a matrix where each *right-to-left* diagonal is fixed as opposed to the fixed *left-to-right* diagonal property of a Toeplitz matrix as defined in Definition 2, but it is essentially the same as a Toeplitz matrix.

The input message m can also be represented by a polynomial $m(x)$ of degree $M-1$.

$$m(x) = m_0 + m_1 x + \cdots + m_{M-1} x^{M-1}$$

What we want to prove is the following Lemma and Theorem.

Lemma 1. For any $n \geq r$, the reduction of x^n modulo $f(x)$ is a linear combination of $\{x^0, x^1, \ldots, x^{r-1}\}$, and that this linear combination is identical to the coefficients in the expression of k_n as a linear combination of $\{k_0, \ldots, k_{r-1}\}$.

Proof. Since we always have Equation 4 above, we additionally need to prove that for any $n \geq r$ we also have:

$$x^n \quad = \quad f_0 x^{n-r} + f_1 x^{n-r+1} + \cdots + f_{r-1} x^{n-1} \pmod{f(x)}$$

This can be proved by induction. This is clearly true when $n = r$. We now assume that this is true for $n = i$ such that $i \geq r$: $x^i = f_0 x^{i-r} + f_1 x^{i-r+1} + \cdots + f_{r-1} x^{i-1} \pmod{f(x)}$, then multiplying both sides by x implies that this is also true for $n = i + 1$.

Although this only shows that k_n is a linear combination of $\{k_{n-r}, \ldots, k_{n-1}\}$ which is identical to the coefficients in the expression of x^n as a linear combination of $\{x^{n-r}, \ldots, x^{n-1}\}$, if we repeatedly apply the same result to every element in these linear combinations until k_n and t^n are only repsensented by $\{k_0, \ldots, k_{r-1}\}$ and $\{x^0, x^1, \ldots, x^{r-1}\}$, respectively, we will get the proof. □

Theorem 2. If $f(x)$ devides $m(x)$ then the inner product modulo 2 of m and the first M bits output from a LFSR with a feedback rule $f(x)$ is zero regardless of what the initial r-bit seed is, i.e. if $f(x) | m(x)$ then $(k_0 \cdots k_{M-1}, m_0 \cdots m_{M-1})_2 = 0$ for any value of $k = k_0 \cdots k_{r-1}$.

Proof. We first have

$$(k_0 \cdots k_{M-1}, m_0 \cdots m_{M-1})_2 = \sum_{i=0}^{M-1} k_i m_i = \sum_{m_i=1,\ i\in[0,M-1]} k_i$$

For any $i \geq r$ k_i is a linear combination of $\{k_0, \ldots, k_{r-1}\}$, and thus $\Sigma_{m_i=1} k_i$ is also a linear combination of $\{k_0, \ldots, k_{r-1}\}$. Using Lemma 1, the latter linear combination will be identical to the coefficients in the expression of $\Sigma_{m_i=1} x^i$ as a linear combination of $\{x^0, \ldots, x^{r-1}\}$ modulo $f(x)$. Now suppose that:

$$(k_0 \cdots k_{M-1}, m_0 \cdots m_{M-1})_2 = \sum_{m_i=1,\ i\in[0,M-1]} k_i = a_0 k_0 + a_1 k_1 + \cdots + a_{r-1} k_{r-1}$$

Using Lemma 1 and the above observation, we will have

$$\begin{aligned} a_0 x^0 + a_1 x^1 + \cdots + a_{r-1} x^{r-1} &= \sum_{m_i=1,\ i\in[0,M-1]} \left(x^i \pmod{f(x)}\right) \\ &= \left(\sum_{m_i=1,\ i\in[0,M-1]} x^i\right) \pmod{f(x)} \\ &= m(x) \pmod{f(x)} \end{aligned}$$

As a result, when $f(x)|m(x)$ all of the coefficients in the expression of $(k_0 \cdots k_{M-1}, m_0 \cdots m_{M-1})_2$ as a linear combination of $\{k_0, \ldots, k_{r-1}\}$ will be zero regardless of the value of the initial seed $k = k_0 \cdots k_{r-1}$. □

Since each column in the Toeplitz matrix $A(k)$ is a continuous sequence of M bits which is taken from $k' = k_0 \cdots k_{K-1}$, applying Theorem 2, we will have $h(k, m) = 0$ for any value of k when the polynomial representing the feedback rule of the LFSR divides the polynomial representing the input message.