GAUSS PERIODS: ORDERS AND CRYPTOGRAPHICAL APPLICATIONS

SHUHONG GAO, JOACHIM VON ZUR GATHEN AND DANIEL PANARIO

February 8, 1996

ABSTRACT. Experimental results on the multiplicative orders of Gauss periods in finite fields are presented. These results indicate that Gauss periods have high order and are often primitive (self-dual) normal elements in finite fields. It is shown that Gauss periods can be exponentiated in quadratic time. An application is an efficient pseudorandom bit generator.

1. INTRODUCTION

For a prime power q, $\mathbb{F}_q$ denotes the finite field of q elements. Let n and k be positive integers such that $r = nk+1$ is a prime, not dividing q, and $\mathcal{K}$ the unique subgroup of order k of the multiplicative group of $\mathbb{Z}_r = \mathbb{Z}/r\mathbb{Z}$. For any primitive rth root β of unity in some extension field of $\mathbb{F}_q$, the element

$$\alpha = \sum_{a\in\mathcal{K}} \beta^a$$

is called a *Gauss period* of type (n, k) over $\mathbb{F}_q$. It is easy to see that a Gauss period of type (n, k) belongs to $\mathbb{F}_{q^n}$.

Adleman & Lenstra (1986) and Mullin et al. (1988) used Gauss periods to construct field extensions and normal bases with special properties over finite fields. A *normal basis* for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ is a basis of the form $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ generated by some $\alpha \in \mathbb{F}_{q^n}$. Any such α is called a *normal element.*

A Gauss period of type (n, k) over $\mathbb{F}_q$ generates a normal basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ if and only if $\gcd(e, n) = 1$ where e denotes the index of q modulo $r = nk + 1$ (Wassermann 1990, 1993, Gao et al. 1995). Gao et al. (1995) present a method for fast multiplication and division under the normal bases generated by Gauss periods; thus exponentiation in finite fields can be sped up. We refer to that paper

1991 *Mathematics Subject Classification.* Primary 11T30, 94A60; Secondary 11Y16, 12Y05, 68Q25.

Key words and phrases. Finite fields, primitive elements, normal bases, cryptography, pseudorandom bit generators.

This paper is in final form, no version of it will be submitted for publication elsewhere.

and the books by Jungnickel (1993) and Menezes et al. (1993) for a discussion of the literature.

Gauss periods of type $(n, 2)$ over $\mathbb{F}_2$ also have other remarkable properties. Gao & Vanstone (1995) proved that they can be exponentiated in $O(n^2)$ bit operations. This is faster than any known algorithms for exponentiation of an arbitrary element in $\mathbb{F}_{2^n}$. The orders of Gauss periods of type $(n, 2)$ over $\mathbb{F}_2$ were also computed for $n \leq 1200$. The experimental results in their paper show that Gauss periods have high multiplicative order, and in fact are often primitive elements over $\mathbb{F}_2$. This is useful in cryptosystems where a fixed element needs to be raised to many large powers.

Naturally, one can ask if the above properties hold for Gauss periods of type (n, k) over $\mathbb{F}_2$ with $k > 2$. In the next section, we prove that, for any fixed k and q, a Gauss period of type (n, k) over $\mathbb{F}_q$ can indeed be exponentiated in $O(n^2)$ operations in $\mathbb{F}_q$. We computed the multiplicative orders of all Gauss periods of type (n, k) for $n \leq 1200$ and $3 \leq k \leq 20$ that generate normal bases for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ as far as the known factorizations of $2^n - 1$ permit. Our experiments show that Gauss periods of type (n, k) for $k \geq 3$ also have high orders and are often primitive. This means that Gauss periods are often primitive normal elements. When k is even, the normal bases generated by Gauss periods of type (n, k) over $\mathbb{F}_2$ are self-dual. Gauss periods thus are often primitive self-dual normal elements as well. In Section 3, we summarize our experimental results, state some conjectures about primitive normal elements, and show how to construct a primitive element from an element with high order. Finally, we mention in Section 4 some cryptographical applications. In particular, we describe a pseudorandom bit generator based on exponentiation in $\mathbb{F}_{2^n}$, and discuss its security and efficiency.

Our work also contributes to the construction of primitive polynomials and primitive normal polynomials, since their irreducible polynomials are normal and primitive when Gausss periods are primitive. The related literature is mentioned at the end of Section 3.

2. Fast exponentiation of Gauss periods

In this section we show that, for fixed k and q, a Gauss period of type (n, k) can be exponentiated in $O(n^2)$ operations in $\mathbb{F}_q$; see also Gao et al. (1995).

A pair (n, k) is a *Gauss pair* over $\mathbb{F}_q$ if $r = nk + 1$ is a prime not dividing q and $\gcd(e, n) = 1$, where e is the index of q modulo r, i.e., $e = nk/\mathrm{ord}_r(q)$. When q is understood, we simply say that (n, k) is a Gauss pair. It is always assumed that (n, k) is a Gauss pair in the sequel.

In the notation of the introduction, $\mathcal{K}$ is the unique subgroup of order k of $\mathbb{Z}_r^\times$, and

$$\alpha = \sum_{a \in \mathcal{K}} \beta^a$$

where β is a primitive rth root of unity. Let

$$\mathcal{K}_i = q^i \mathcal{K} = \{aq^i \bmod r : a \in \mathcal{K}\} \text{ for } 0 \leq i < n.$$

Then $\mathbb{Z}_r^\times$ is the disjoint union of $\mathcal{K}_0, \mathcal{K}_1, \cdots, \mathcal{K}_{n-1}$. We write $\alpha_i = \alpha^{q^i}$ for $0 \le i < n$. Then $(\alpha_0, \alpha_1, \cdots, \alpha_{n-1})$ is a normal basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. It is shown in Gao et al. (1995) that

$$\alpha \cdot \alpha_i = \delta_i k + \sum_{0 \le j < n} t_{ij} \alpha_j$$

where $t_{ij} = |(1 + \mathcal{K}_i) \cap \mathcal{K}_j|$, $\delta_i = 1$ if $i = i_0$ and 0 otherwise, and i_0 is such that $-1 \in \mathcal{K}_{i_0}$. We note that each $\alpha \cdot \alpha_i$ has at most k nonzero terms. We store these terms only.

Theorem 2.1. *Let α be a Gauss period of type (n, k) over $\mathbb{F}_q$, with k and q bounded. For any $0 \le M < q^n$, α^M can be computed in $O(n^2)$ operations in $\mathbb{F}_q$.*

Proof. We want to compute α^M expressed in the normal basis $(\alpha_0, \alpha_1, \cdots, \alpha_{n-1})$. We use a redundant representation of $A \in \mathbb{F}_{q^n}$, writing

$$A = \left(\sum_{0 \le i < n} a_i \alpha_i \right) + a_n,$$

where $a_0, \cdots, a_{n-1}, a_n \in \mathbb{F}_q$. Thus A is represented by an $(n+1)$-tuple $(a_0, \cdots, a_{n-1}, a_n)$. Of course, this representation is not unique; we just want to compute any one of the representations. For example, the unit 1 can either be represented as $(0, \cdots, 0, 1)$ or $(-1, \cdots, -1, 0)$, since $-1 = \sum_{0 \le i < n} \alpha_i$. Our algorithm benefits from this flexibility.

Thus A^q is the $(n+1)$-tuple obtained from that of A by shifting cyclically the first n coordinates to the right by one position (the last coordinate remains fixed). So the cost for computing a qth power is negligible.

For any $A \in \mathbb{F}_{q^n}$ and $0 \le j < n$, we compute $\alpha_j A$. Assume that $A = (\sum_{0 \le i < n} a_i \alpha_i) + a_n \in \mathbb{F}_{q^n}$, with $a_0, \cdots, a_n \in \mathbb{F}_q$. Then

$$\alpha_j A = \left(\sum_{0 \le i < n} a_i (\alpha \alpha_{i-j})^{q^j} \right) + a_n \alpha_j.$$

Since $\alpha \alpha_i$ is a sum of at most k terms, $\alpha_j A$ can be computed in $O(nk)$ operations in $\mathbb{F}_q$.

Now to compute α^M, we use the q-ary representation $M = \sum_{0 \le j \le \ell} m_j q^j$, with $0 \le m_j < q$ for all j. Then

$$\alpha^M = \prod_{0 \le j \le \ell} \left(\alpha^{q^j} \right)^{m_j} = \prod_{0 \le j \le \ell} \alpha_j^{m_j}.$$

This suggests that we compute α^M iteratively. Initially, set $A := 1$. For j from 0 to ℓ set $A := \alpha_j^{m_j} A$. Then, at the end, we have $A = \alpha^M$. We compute $\alpha_j^{m_j} A$ by multiplying α_j to A repeatedly m_j times. This algorithm computes α^M in

$$O\left(\left(\sum_{0 \le j < n} m_j \right) nk \right) = O(\sigma_q(M) nk)$$

operations in $\mathbb{F}_q$ where $\sigma_q(M)$ is the sum of digits of M in q-ary representation. For $0 \le M < q^n$, since $\sigma_q(M) \le (q-1)n \le qn$, α^M can be computed in $O(n^2qk)$ operations in $\mathbb{F}_q$. When the values of q and k are small or bounded, this is faster than any known algorithms for exponentiation of an arbitrary element, and the running time can be written as $O(n^2)$. Furthermore, no storage is needed here. $\square$

Exponentiation of an arbitrary element in $\mathbb{F}_{q^n}$ (with q bounded) costs $O(n^2 \operatorname{loglog} n)$ operations in $\mathbb{F}_q$ by the currently fastest algorithm, with storage for $O(n/\log_q^2 n)$ elements in $\mathbb{F}_{q^n}$ (Gao et al. 1995).

Recently, von zur Gathen & Pappalardi (1996) proved that, for any fixed k and q, there are infinitely many values of n such that (n, k) is a Gauss pair over $\mathbb{F}_q$. In fact, they determine a positive density for the primes $nk+1$, where (n, k) is a Gauss pair, in the set of all primes. Thus, there are infinitely many fields $\mathbb{F}_{q^n}$ in which Gauss periods can be exponentiated easily. In the next section, we present experimental results which indicate that Gauss periods almost always have high order.

3. Experimental results

We computed the multiplicative orders of Gauss periods for all Gauss pairs (n, k) over $\mathbb{F}_2$ with $3 \le k \le 20$ and $2 \le n < 527$, and also did the corresponding calculations for $527 \le n \le 1200$ as far as current knowledge of the factors of $2^n - 1$ permits. The results are tabulated in the table of the appendix, where "Ind" denotes index, which equals $2^n - 1$ divided by the corresponding multiplicative order. An entry with a question mark "i?" in the "Ind" column means that the corresponding index was computed from the partial factorization of $2^n - 1$, thus the true index is i times some of the unknown prime factors of $2^n - 1$ (we believe that these extra factors are unlikely to occur).

Our experiments show that Gauss periods have the expected multiplicative properties: they almost always have high multiplicative orders and are frequently primitive. More precisely, in the range $2 \le n < 527$ and $2 \le k \le 20$ there are 1179 Gauss pairs (n, k), all the corresponding Gauss periods have order $\ge (2^n - 1)/n$ except for 8 pairs, and 906 of them are primitive. In the range $527 \le n \le 1200$ and $2 \le k \le 20$, there are 1239 Gauss pairs (n, k), and the corresponding Gauss periods have order $\ge (2^n - 1)/n$ except for 5 pairs, and 965 of them are primitive, provided that the corresponding index entries i? are the true indices.

All the Gauss periods in the table generate normal bases over $\mathbb{F}_2$. When the index is 1, the corresponding basis is a primitive normal basis. Thus Gauss periods yields many primitive normal bases over $\mathbb{F}_2$. Also, when k is even, the normal basis generated by a Gauss period of type (n, k) over $\mathbb{F}_2$ is self-dual (Gao et al. 1995). We see that Gauss periods generate many primitive self-dual normal bases as well.

Gao & Vanstone (1995) observe that if n and $2n+1$ are both primes then Gauss periods of type $(n, 2)$ are primitive elements in $\mathbb{F}_{2^n}$ for $n \le 1200$. Our experimental data show that their observation still holds for general Gauss periods. It is formulated as follows.

Conjecture 3.1. *If n and $nk+1$ are both primes and $k < \log_2(n+1)$, then Gauss periods of type (n,k) form a primitive normal basis for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$.*

We note that normality is not a problem here. Since the order m of 2 modulo $nk+1$ is at least $\log_2(nk+1) \geq \log_2(n+1) > k$, it follows that n divides $m = nk/e$ and thus $\gcd(e,n) = 1$.

Wassermann (1993) proves that for a given n there exists a Gauss pair (n,k) over $\mathbb{F}_2$ if and only if $8 \nmid n$. There are 61 values of $n \leq 1200$ for which there is no Gauss pair (n,k) with $k \leq 20$. For each of these n, we list in the appendix the smallest Gauss pair (n,k) and the corresponding index (or index?).

Our computations lead us to believe that for every n not divisible by 8 there is a Gauss pair (n,k) yielding a primitive Gauss period $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. As an experiment, we verified this for all $n < 527$. If there is no primitive Gauss period for $k \leq 20$, then the last entry of k is the smallest k whose Gauss period is primitive and normal in $\mathbb{F}_{2^n}$. The largest such k occurs in (490, 69).

Conjecture 3.2. *For any positive integer n not divisible by 8, there exists an integer $k \geq 1$ such that the Gauss period of type (n,k) is primitive normal in $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$.*

Motivated by this work and the previous work of Gao & Vanstone (1995), von zur Gathen & Shparlinski (1995) prove that Gauss periods of type $(n,2)$ have order at least $2^{\sqrt{2n}-2}$.

Next, we show how to construct primitive elements from elements of high order. The constructed primitive elements will still be essentially as easy to exponentiate as Gauss periods.

Theorem 3.3. *Let $\alpha \in \mathbb{F}_{2^n}$ with order $m \geq (2^n-1)/n^c$ where c is a constant. A primitive element can be constructed from α deterministically in polynomial time.*

Proof. Let $e = (2^n-1)/m$ be the index of α. Then $e \leq n^c$. Write $e = e_1 e_2$, where $\gcd(e_2, m) = 1$ and every prime divisor of e_1 divides m. Let $\beta_1 \in \mathbb{F}_{q^n}$ satisfy

$$\beta^{e_1} = \alpha,$$

and let β_2 be a primitive e_2th root of unity in $\mathbb{F}_{2^n}$. Then β_1 has order me_1, and $\beta_1\beta_2$ has order $me_1e_2 = 2^n - 1$, as $\gcd(me_1, e_2) = 1$. This means that $\beta_1\beta_2$ is primitive in $\mathbb{F}_{2^n}$. For any $e \leq n^c$ and $\alpha \in \mathbb{F}_{2^n}$, the polynomial $x^e - \alpha$ can be factored (deterministically) in polynomial time. Thus β_1 and β_2 as above can be constructed in polynomial time. $\square$

In the special case $e = 2^k - 1$, the equation $x^e - \alpha$ can be written as $x^{2^k} = \alpha x$, which corresponds to a system of linear equations over $\mathbb{F}_2$, and can be solved by any efficient algorithms for linear equations. Our experimental data shows that many Gauss periods have indices $3, 7, 15$, etc, which are of the form $2^k - 1$. Thus from our table of Gauss periods, it is easy to construct primitive elements if one really needs primitive elements instead of elements of high orders. In the table, we give for each $n < 527$ and $8 \nmid n$, the smallest k such that a Gauss period of type (n,k)

has index at most n. One can see that for these n it is possible to find a reasonably small such k.

Finally, we make some comments on the related work in the literature. As mentioned in the introduction, our work also contributes to the construction of primitive polynomials and primitive normal polynomials, since their irreducible polynomials are normal and primitive when Gausss periods are primitive. Hansen & Mullen (1992) and Morgan & Mullen (1994) give tables of primitive polynomials and primitive normal polynomials of degree m over $\mathbb{F}_p$ for all prime powers $p^m \leq 10^{50}$ with $p \leq 97$. Živković (1994a, 1994b) gives a more extensive table of primitive polynomials of degree $m \leq 1200$ (and a few values of m between 1200 and 5000) over $\mathbb{F}_2$ when the factorization of $2^m - 1$ is known. In their work, they search for sparse polynomials, i.e., those with the smallest number of nonzero terms. Such polynomials are useful in efficient implementation of feedback shift registers. In the extreme case, there is much interest in constructing irreducible trinomials over $\mathbb{F}_2$. Zierler & Brillhart (1968, 1969) give a table of irreducible trinomials of degree ≤ 1000. Blake et al. (1994) extend this list to all irreducible trinomials of degree ≤ 2000 over $\mathbb{F}_2$ (and a table for degree ≤ 5000 is available from the authors).

It is, however, not clear how primitive elements from sparse polynomials can be exponentiated faster than an arbitrary primitive element. The primitive polynomials from Gauss periods may not in general be sparse, but they do provide computational advantage in fast exponentiation as shown by Theorem 2.1.

4. Cryptographical applications

Let $\alpha \in \mathbb{F}_{q^n}$ be a primitive element (or an element of high order, say at least $(q^n - 1)/n^c$ for some constant c). Computing the exponentiation function that maps $x \in \{0, \ldots, q^n - 1\}$ to α^x is easy, but computing its inverse function, i.e., computing x given α^x, called the discrete logarithm problem, is believed to be hard in general. This one-wayness of exponentiation has found many applications in public-key cryptography: Diffie-Hellman key exchange (Diffie & Hellman 1976), password schemes (Lamport 1981), ElGamal cryptosystem (ElGamal 1985), cryptosystems over $\mathbb{F}_2$ (Agnew et al. 1991, Agnew et al. 1993), smart cards (Beth 1988, Schnorr 1990, 1991), US Digital Signature Algorithm (NIST 1994), pseudorandom bit generators (Blum & Micali 1984, Long & Widgerson 1988). Pseudorandom bit generators based on discrete logarithms in finite fields are used by Zheng & Seberry (1992) and Lim & Lee (1993) to construct cryptosystems that leak no partial information and are secure against adaptively chosen ciphertext attacks. In these applications, one needs a fixed element of high order and computes α^t for many random large integers t. For example, in a signature scheme, for each signature one needs to generate a random integer t and compute α^t. Sometimes the computing power of the signature generating device is limited, e.g., in a smart card. So α has to be chosen such that exponentiation of α is easy. In practice, the currently popular choice for α is from elements in $\mathbb{F}_p$. If the prime p has n bits, then computing α^t needs $O(n^3)$ bit operations using repeated square and multiply method, or $O(n^2 \log n \operatorname{loglog} n)$ bit operations using FFT-based fast multiplication algorithms.

However, if we choose α to be a Gauss period of type (n, k) in $\mathbb{F}_{2^n}$ for a small k then the cost of exponentiating α is reduced to $O(n^2)$ bit operations, which is just the cost of one multiplication by the "classical" method. Our experimental results show that α almost always has high order and is often primitive. Our exponentiation algorithm is also easy to implement. Gauss periods are therefore highly attractive in these applications.

In the following, we describe Blum & Micali's pseudorandom bit generator based on exponentiation in $\mathbb{F}_p$, then we adapt it to the fields $\mathbb{F}_{2^n}$ and comment on its security and efficiency.

A *pseudorandom bit generator* produces sequences of bits (0 or 1) that cannot be distinguished from truly random sequences of bits of equal length by any (probabilistic) polynomial time algorithm. Blum & Micali (1984) presented the following pseudorandom bit generator. Let $m \geq 1$ be a fixed integer. Given $n \geq 2$, select an odd prime p of n bits, and a primitive root α. Pick a random integer a_0 (the seed) in the range $1 < a_0 < p-1$. Set

$$a_{k+1} \equiv \alpha^{a_k} \bmod p \text{ for } k \geq 0,$$

and

$$b_{k+1} = 1 \text{ if } a_{k+1} > (p-1)/2, \text{ and } b_{k+1} = 0 \text{ otherwise.}$$

Then $\{b_k : 1 \leq k \leq n^m + m\}$ is a sequence of $n^m + m$ bits generated from the n-bit seed a_0. Blum & Micali (1984) proved that if the discrete logarithm problem is hard then this much longer sequence of bits is pseudorandom. Blum & Micali's generator outputs only one bit at each iteration, i.e., each bit costs one exponentiation mod p. Long & Wigderson (1988) extended this to output about $\log n$ bits at each iteration.

We now adapt the above generator to the fields $\mathbb{F}_{2^n}$. Under a fixed basis for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, an element $x \in \mathbb{F}_{2^n}$ is represented as a sequence of n bits (0 and 1). We use $\overline{x}$ to denote the integer whose binary representation is the same as x. Let $\alpha \in \mathbb{F}_{2^n}$ be a primitive element (or an element of high order). Pick a random element $x_0 \in \mathbb{F}_{2^n}$. Set

$$x_{k+1} = \alpha^{\overline{x}_k} \text{ for } k \geq 0,$$

and let z_{k+1} be the least significant bit

$$z_{k+1} = \overline{x}_{k+1} \bmod 2.$$

Then $\{z_k : 1 \leq k \leq n^m + m\}$ is a sequence of bits generated from the seed x_0. We want to show that this sequence is pseudorandom.

Let f be a one-way function, i.e, it is easy to compute but hard to invert. A Boolean predicate B (i.e., $B(x) = 0$ or 1) is said to be *hard* for f if an oracle for $B(f(x))$ allows one to invert f easily. Blum & Micali (1984, Theorem 2) proved that if B is a hard predicate for a one-way function f then the following sequence is pseudorandom:

$$B(f(a_0)), B(f^2(a_0)), B(f^3(a_0)), \cdots, B(f^k(a_0)), \cdots$$

where a_0 is randomly chosen. To show that our sequence is pseudorandom, it is enough to prove Theorem 4.1 below. For any $\beta \in \mathbb{F}_{2^n}$, the smallest positive integer

x such that $\beta = \alpha^x$ is called the discrete logarithm of β with respect to α, denoted by $\log_\alpha \beta$. If no such x exists, we set (arbitrarily) $\log_\alpha \beta = 0$.

Theorem 4.1. *Every bit of the discrete logarithm in $\mathbb{F}_{2^n}$ is a hard predicate (for the exponentiation function).*

Proof. Let $B(x)$ be the least significant bit of an integer x. We show how to compute discrete logarithms via an oracle for $B(\log_\alpha \beta)$ which returns the least significant bit of the discrete logarithm of any $\beta \in \mathbb{F}_{2^n}$. Note that for any integer i, $\log_\alpha \beta^{2^i} \equiv 2^i \log_\alpha \beta \bmod 2^n - 1$. Suppose that

$$\log_\alpha \beta = \sum_{k=0}^{n-1} a_k 2^k = (a_0, a_1, \cdots, a_{n-1})_2.$$

Then for $i \in \mathbb{N}$

$$2^i \log_\alpha \beta \equiv \sum_{k=0}^{n-1} a_k 2^{k+i} \equiv \sum_{k=0}^{n-1} a_{k-i} 2^k \bmod 2^n - 1,$$

where the index of a is computed modulo n, and

$$\log_\alpha \beta^{2^i} = (a_{n-i}, \cdots, a_{n-1}, a_0, \cdots, a_{n-i-1})_2.$$

Therefore

$$B(\log_\alpha \beta^{2^i}) = a_{n-i} \text{ for } 0 \leq i \leq n-1,$$

and

$$\log_\alpha \beta = (B(\log_\alpha \beta), B(\log_\alpha \beta^{2^{n-1}}), \cdots, B(\log_\alpha \beta^2))_2.$$

This means that $\log_\alpha \beta$ can be computed by n calls to the oracle. Since the bits can be shifted cyclically, $\log_\alpha \beta$ can also be computed by n calls to an oracle for any bit. Therefore every bit is a hard predicate. $\square$

We have assumed that the oracle is perfect, that is, it always gives correct answers. Blum & Micali (1984) dealt with the more general case of nonperfect oracles (an oracle that may give wrong answers, but it gives correct answers sufficiently more frequently than incorrect ones). It seems likely that Theorem 4.1 still holds for nonperfect oracles.

Our theorem says that every bit of the discrete logarithms is a hard predicate, that is, every bit is individually secure, provided the discrete logarithm is hard to compute. This is different from the discrete logarithms modulo an odd prime p, where its least significant bit is not secure while the most significant bit is indeed secure (Peralta 1986). It seems also possible to modify the proof in Long & Wigderson (1988) to show that any $O(\log n)$ consecutive bits of the discrete logarithms in $\mathbb{F}_{2^n}$ are simultaneously secure. In this case, the modified pseudorandom bit generator could output $O(\log n)$ bits per iteration.

In implementing our pseudorandom bit generator in $\mathbb{F}_{2^n}$, one can choose α to be a Gauss period of type (n, k) for a small k. The cost of exponentiating α at each iteration is only $O(n^2)$ bit operations. This is advantageous compared to the generator in $\mathbb{F}_p$, where exponentiation needs $O(n^2 \log n \log\log n)$ bit operations by using fast multiplication algorithms.

Acknowledgement. We used the updated factorization tables of the Cunningham project (Brillhart et al. 1988), maintained by S.S. Wagstaff. We also used Maple in our computations.

References

1. L.M. Adleman and H.W. Lenstra, Jr., *Finding irreducible polynomials over finite fields*, Proc. 18th Annual ACM Symp. on Theory of Computing (1986), 350–355.
2. G.B. Agnew, R.C. Mullin, I.M. Onyszchuk and S.A. Vanstone, *An implementation for a fast public key cryptosystem*, J. of Cryptology, **3** (1991), 63–79.
3. G.B. Agnew, R.C. Mullin and S.A. Vanstone, *An implementation of elliptic curve cryptosystems over $F_{2^{155}}$*, IEEE J. on Selected Areas in Communications, **11** (1993), 804–813.
4. T. Beth, *Efficient zero-knowledge identification scheme for smart cards*, Advances in Cryptology (Proc. Eurocrypt '88), LNCS **330** (1988), 77–84.
5. I.F. Blake, S. Gao and R. Lambert, *Constructive problems for irreducible polynomials over finite fields*, in Information Theory and Applications (A. Gulliver and N. Secord, eds), Springer LNCS **793** (1994), 1–23.
6. M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Computing, **13** (1984), 850–864.
7. J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman and S.S. Wagstaff, *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ Up to High Powers*, Contemporary Mathematics, **22**, 2nd ed., AMS, 1988.
8. W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Trans. Info. Th., **22** (1976), 644–654.
9. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Info. Th., **31** (1985), 469–472.
10. S. Gao, J. von zur Gathen and D. Panario, *Gauss periods, primitive normal bases, and fast exponentiation in finite fields*, preliminary version in Proc. Latin'95, Valparaíso, Chile, Springer LNCS **911** (1995), 311–322; full version in Technical Report 296/95, Department of Computer Science, University of Toronto, 1995.
11. S. Gao and S.A. Vanstone, *On orders of optimal normal basis generators*, Math. Comp., **64** (1995), 1227–1233.
12. J. von zur Gathen and F. Pappalardi, *Density estimates related to Gauss periods*, 1996, preprint.
13. J. von zur Gathen and I. E. Shparlinski, *Order of Gauss periods in finite fields*, Proc ISAAC '95, Cairns, Australia, (1995), to appear.
14. C.F. Gauss, Disquisitiones Arithmeticae, Braunschweig, 1801. English Edition, Springer-Verlag, 1986.
15. T. Hansen and G.L. Mullen, *Primitive polynomials over finite fields*, Math. Comp., **59** (1992), 639–643.
16. D. Jungnickel, Finite Fields: Structure and Arithmetics, Bibliographisches Institut, Mannheim, 1993.
17. L. Lamport, *Password authentication with insecure communication*, Comm. ACM, **24** (1981), 770–772.

18. C. H. Lim and P. J. Lee, *Another method for attaining security against adaptively chosen ciphertext attacks,* Advances in Cryptology (Proc. Crypto '93), LNCS **773** (1994), 420–434.
19. D. L. Long and A. Wigderson, *The discrete log hides $O(\log n)$ bits,* SIAM J. Computing, **17** (1988), 363–372.
20. Alfred J. Menezes, Ian F. Blake, XuHong Gao, Ronald C. Mullin, Scott A. Vanstone, and Tomik Yaghoobian, *Applications of finite fields,* Kluwer Academic Publishers, (1993).
21. I.H. Morgan and G.L. Mullen, *Primitive normal polynomials over finite fields,* Math. Comp., **63** (1994), 759–765.
22. R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone and R.M. Wilson, *Optimal normal bases in $GF(p^n)$,* Discrete Applied Math., **22** (1988/1989), 149–161.
23. National Institute for Standard and Technology, Federal Information Processing Standard for Digital Signature Standard (DSS), FIPS 186, May 1994.
24. R. Peralta, *Simultaneous security of bits in the discrete log* Advances in Cryptology (Proc. Eurocrypt '85), LNCS **219** (1986), 62–72.
25. C. P. Schnorr, *Efficient identification and signatures for smart cards* Advances in Cryptology (Proc. Crypto '89), LNCS **435** (1990), 239–252.
26. C. P. Schnorr, *Efficient signature generation by smart cards,* J. of Cryptology, **4** (1991), 161–174.
27. A. Wassermann, *Konstruktion von Normalbasen,* Bayreuther Mathematische Schriften, **31** (1990), 155–164.
28. A. Wassermann, *Zur Arithmetik in endlichen Körpern,* Bayreuther Mathematische Schriften, **44** (1993), 147–251.
29. Y. Zheng and J. Seberry *Practical approaches for attaining security against adaptively chosen ciphertext attacks,* Advances in Cryptology (Proc. Crypto '92), LNCS **740** (1993), 292–304.
30. N. Zierler and J. Brillhart, *On primitive trinomials (mod 2),* Information and Control, **13** (1968), 541–554.
31. N. Zierler and J. Brillhart, *On primitive trinomials (mod 2), II,* Information and Control, **14** (1969), 566–569.
32. M. Živković, *A table of primitive binary polynomials,* Math. Comp., **62** (1994a), 385–386.
33. M. Živković, *Table of primitive binary polynomials, II,* Math. Comp., **63** (1994b), 301–306.

Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1907, USA, sgao@math.clemson.edu

Fachbereich Mathematik-Informatik, Universität-GH Paderborn, D-33095 Paderborn, Germany, gathen@uni-paderborn.de

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada, daniel@cs.toronto.edu