ON THE RANDOMNESS OF LEGENDRE AND JACOBI SEQUENCES

Ivan Bjerre Damgård[1]

Aarhus University, Mathematical Institute.
Ny Munkegade,
DK 8000 Aarhus C,
Denmark.

Introduction

Most of the work done in cryptography in the last few years depend on the hardness of a few specific number theoretic problems, such as factoring, discrete log, etc. Since no one has so far been able to prove that these problems are genuinely hard, it is clearly of interest to find new candidates for hard problems. In this paper, we propose such a new candidate problem, namely the problem of predicting a sequence of consecutive Legendre (Jacobi) symbols modulo a prime (composite), when the starting point and possibly also the prime is unknown. Clearly, if this problem turns out to be hard, it can be used directly to construct a cryptographically strong pseudorandom bitgenerator. Its complexity seems to be unrelated to any of the well known number theoretical problems, whence it may be able to survive the discovery of fast factoring or discrete log algorithms. Although the randomness of Legendre sequences has part of the folklore in number theory at least since the thirties, they have apparently not been considered for use in cryptography before.

We first survey some known results about the distribution of squares and nonsquares modulo a prime. These results all support the assumption that Legendre sequences look random with respect to elementary statistical tests.

We then use Levin's Isolation Theorem [BoHi] to relate the complexity of predicting Legendre sequences to the complexity of predicting Jacobi sequences. The main result of this is that if Legendre sequences are unpredictable in a very weak sense, then Jacobi sequences modulo composites with enough prime factors are strongly unpredictable, as required for cryptographic strength.

We end the paper by giving results of some emphirical tests on Legendre sequences, carried out for primes of length 25 to 400 bits. Also some possibilities for generalizing the ideas are mentioned. These ideas give significant efficiency improvements over the basic Legendre generator.

---

[1]This research was supported by the Danish Natural Science Research Council.

1. Notation

Let p be a k-bit prime, and let $a \in Z_p^*$. We then define the Legendre symbol of a modulo p, $(\frac{a}{p})$ to be 1 if a is a square modulo p, and -1 if a is a non square. For convenience, we define $(\frac{0}{p})$ to be 1.

When $n = p_1 \cdots p_r$ is a k-bit composite with prime factors $p_1, \ldots, p_r$, we define the Jacobi symbol of a modulo n to be $(\frac{a}{n}) = (\frac{a}{p_1}) \cdots (\frac{a}{p_r})$, for $0 \le a \le n-1$.

The *Legendre sequence* with starting point a and length l is the +-1 sequence

$$(\frac{a}{p}), (\frac{a+1}{p}), \cdots, (\frac{a+l}{p}).$$

Jacobi sequences are defined correspondingly.

We can now state formally our basic problem:

Problem P1.
Let L be the Legendre sequence modulo p with starting point a and length $P(k)$, for some polynomial P. Given L (but not a or p), find $(\frac{a+P(k)+1}{p})$ □

Correspondingly for Jacobi symbols:

Problem P2.
Let J be the Jacobi sequence modulo n with starting point a and length $P(k)$, for a polynomial P. Given J, find $(\frac{a+P(k)+1}{n})$ □

2. Known Results On the Distribution of Squares Modulo a Prime

The distribution of quadratic residues and non residues has been studied at least since the end of the last century. One of the first major contributions was made by Davenport [Da]:

Let S be a finite sequence of +-1's of length l. Let $p(S)$ be the number of occurrences of S in the complete Legendre sequence of p, i.e. the number of $a \in Z_p^*$ such that

$$(\frac{a}{p}), (\frac{a+1}{p}), \cdots, (\frac{a+l}{p}) = S.$$

Davenport proved that

$$p(S) = \frac{p}{2^l} + O(p^\varepsilon),$$

where ε is a constant between 0 and 1 which is only a function of l. In other words: the distribution of subsequences of fixed length tends to the uniform distribution exponentially

in $\log_2(p)$. A uniform distribution is of course what one would expect from a really random sequence.

Perron [Pe] proved a more specific result: let $SQ(p)$ be the set of quadratic residues in Z_p^*. Then for any a, the set $a+SQ(p)$ contains almost exactly as many squares as non squares, the difference being 0 or 1, depending on whether p is 1 or 3 modulo 4. A similar result holds for $Z_p^*-SQ(p)$. This has a number of immediate consequences:

- By setting $a=1$, we get $p(1,1)\approx p(-1,1)\approx p(1,-1)\approx p(-1,-1)$, where the differences are at most 1.
- In general, pairs of symbols separated by a fixed distance are uniformly distributed.
- Define a *block* to be a run of consecutive 1's or -1's. Then by setting $a=1$ we see that half the 1's in the Legendre sequence of p are at the end of a block, and similarly for -1's. Therefore the average length of a block is 2.

Later, Burde [Bu] extended Perron's result for p's congruent to 3 modulo 4. He obtained a system of linear equations with the number of occurrences of subsequences of a fixed length as unknowns. The rank of the system is quadratic as a function of the length of the subsequences considered, and therefore the equations quickly become insufficient to determine the complete distribution. 3 is the largest length for which it can be done, and as for length 2, the distribution is uniform, apart from an "error" of order p^{-1}.

Thus the results of Perron and Burde also support the assumption that Legendre sequences will look random with respect to elementary statistical tests.

From the work of Bach [Ba], one can get very interesting estimates on the distribution of subsequences whose length is allowed to grow with the size of the prime, in contrast with Davenport's results. This is based on results from algebraic geometry by Weil. For example, for p congruent to 3 modulo 4, one can obtain that

$$p(S) \le \frac{p}{2^l} + \frac{(l-1)\sqrt{p}}{2}$$

for any S of length l. Thus, there is a limit to "how bad" the distribution of subsequences can be, for example at least

$$2^l \frac{\sqrt{p}}{\sqrt{p} + 2^{l-1}(l-1)}$$

different subsequences of length l must occur in the complete Legendre sequence. In fact, the results are much more general, and can give information also about the distribution of other character values.

But since the bounds clearly get looser as the length l increases, we are still a long way from results that would imply the impossibility of predicting Legendre sequences in polynomial time.

Many other researchers (Eliott [El], Burgess[Bur]) have looked at this problem from other angles, typically they have been concerned with finding the smallest quadratic non residue, finding the first occurrence of a given substring, etc. Thus these results do not say much about the overall distribution, which is of course what we are interested in.

The main conclusion of all this is a negative one: nothing has been found in the last 100 years or so, which immediately renders Legendre sequences useless for pseudorandom bit generation.

3. Jacobi Sequences are Harder to Predict than Legendre Sequences

Let us first define formally the *Legendre generator*:

Definition 3.1
Let Q be a polynomial. Then with security parameter value k, the Legendre generator takes as input ("seed") a randomly chosen k-bit prime p and a uniformly chosen k-bit number a. It produces as output the Legendre sequence modulo p with starting point $a \bmod p$ and length $Q(k)$, where Legendre symbols are translated into bits such that -1 corresponds to a 1-bit, while 1 corresponds to a 0-bit. This sequence will be called $L(p,a)$, and its i'th element will be denoted $L(p,a)_i$ □

Similarly, we define the *Jacobi generator*:

Definition 3.2
Let P and Q be polynomials. Then with security parameter value k, the Jacobi generator takes as input $Q(k)$ randomly chosen k-bit primes $p_1, \ldots, p_{Q(k)}$ and $Q(k)$ uniformly chosen k-bit numbers $a_1, \ldots, a_{Q(k)}$. Put $n = p_1 \cdots p_{Q(k)}$, and let a be chosen, such that a is congruent to a_i modulo p_i for $i = 1 \cdots Q(k)$ ($p_i = p_j$ for $i \neq j$ only happens with negligible probability). The generator produces as output the Jacobi sequence modulo n with starting point a and length $P(k)$, where Jacobi symbol are translated into bits as above. The sequence will be called $J(n,a)$, and its i'th element will be called $J(n,a)_i$ □

Yao [Kr] has proved that, if given a prefix of the output from a pseudorandom bit generator, it is still hard to predict the next bit, then output from the generator cannot be distinguished from truly random sequences by any feasible algorithm. Thus, informally speaking, all we have to do in order to prove the strength of our generators is to show that $P1$ or $P2$ are hard problems. This can also be stated using Levin's concept of *isolation* [BoHi]: consider some prefix of the output from the generator as a function of the seed. Then we would like the bit following the prefix to be isolated from the prefix itself.

In general, we can think of a pseudorandom bit generator as a probabilistic algorithm G which takes input x chosen from a finite set X_m, where $\{X_m\}_{m=1}^{\infty}$ is a family of finite sets, and m can be thought of as a security parameter. The output $G(x)$ is a bitstring whose i'th bit is denoted $G(x)_i$. We now have the following more formal definition of next-bit-security:

Definition 3.3
The generator G is said to be *strongly unpredictable*, if for all polynomials P and probabilistic circuits C, the following holds only for finitely many m:

there exists an i, such that

$$Prob(C(G(x)_1, \ldots, G(x)_{i-1}) = G(x)_i) \geq \frac{1}{2} + \frac{1}{P(m)},$$

where $C(G(x)_1, \ldots, G(x)_{i-1})$ denotes the output of C on input $G(x)_1, \ldots, G(x)_{i-1}$, and the probability is taken over the coinflips of C and a uniform choice of $x \in X_m$ □

Definition 3.4
The generator G is said to be *weakly unpredictable*, if the statement in Definition 3.3 holds, with the probability $\frac{1}{2} + \frac{1}{P(m)}$ replaced by $1 - \frac{1}{P(m)}$ □

Thus, for weak unpredictability, we allow algorithms that guess better as m increases, as long as the success probability does not approach 1 too rapidly.

For the next results, we need some more notation:

Definition 3.5
Let G be a pseudorandom bitgenerator as above, and let Q be a polynomial. Then G^Q is a pseudorandom bitgenerator which takes as input $x = (x_1, \ldots, x_{Q(m)})$, where $x_i \in X_m$ are cosen uniformly and independently. It produces as output the sequence whose j'th element is

$$G^Q(x)_j = G(x_1)_j \oplus \cdots \oplus G(x_{Q(m)})_j$$

We now quote from [BoHi] the following definition and theorem:

Definition 3.6
Let $\{X_m\}_{i=1}^{\infty}$ be an infinite family of finite sets, and let B be a function mapping X_m to $\{0,1\}$. Let f be a function such that $f: X_m \to \{0,1\}^{P(m)}$ for some polynomial P. Then we say that B is (p,T)-*isolated* from f if every circuit with $Q(m)$ inputs and size at most T satisfies

$$|Prob(C(f(x)) = B(x)) - \frac{1}{2}| \leq \frac{p}{2},$$

where x is chosen uniformly from X_m □

Thus, isolation measures the hardness of predicting $B(x)$ given $f(x)$.

Theorem 3.1 (Levin's Isolation Theorem)
If the functions $b_i(x_i)$ are (p,T) isolated from $f_i(x_i)$ for all $1 \leq i \leq n$, then for every $\varepsilon > 0$, the function $b_1(x_1) \oplus \cdots \oplus b_n(x_n)$ is $(p^n + \varepsilon, \varepsilon^2(1-p)^2 T)$-isolated from $f_1(x_1), \ldots, f_n(x_n)$ □

From this follows:

Theorem 3.2
Suppose that the pseudorandom bitgenerator G is weakly unpredictable. Then G^{m^2} is strongly unpredictable.

Proof.
For $x \in X_m$, let $P_i(G(x))$ denote the prefix of $G(x)$ of length $i-1$. Then weak unpredictability implies that for all i smaller than the outputlength of G, $G(x)_i$ is $(1-\frac{2}{R(m)}, T(m))$-isolated from $P_i(G(x))$ for arbitrary polynomials R and T and all sufficiently large m. Put $p(m) = 1 - \frac{2}{m}$. Using the notation of Definition 3.5, we obtain from Levin's theorem by choosing $R(m) = m$ that $G^{m^2}(x)_i$ is $(p(m)^{m^2} + \varepsilon, \varepsilon^2(1-p(m))^2 T(m))$- isolated from $P_i(G(x_1)), \ldots, P_i(G(x_{m^2}))$. From this last bit string, it is easy to compute $P_i(G^{m^2}(x))$. Moreover, $p(m)^{m^2}$ converges to 0 faster than any polynomial fraction. From these facts, we get the strong unpredictability of G^{m^2} by choosing $\varepsilon = \frac{1}{P(m)}$, where P is the polynomial from Definition 3.3 □

Alternatively, we could have used Yao's xor-Theorem, although the conclusion of that theorem as stated in [Kr] is slightly weaker than that of Levin's Theorem.

Theorem 3.2 was already known (see [Kr]) for the special case of generators constructed with an unapproximable predicate and a friendship function [BlMi].

We call attention to two points in connection with this result, which are of interest from a cryptographic point of view:

- What Theorem 3.2. proves is that the next bit of the XOR of several generators is hard to predict, even when given prefixes of the output from *each individual* generator. But in a known plaintext attack on the resulting cryptosystem, a cryptanalyst only knows the XOR of the prefixes, and therefore seems to be faced with an even harder problem. It would be very interesting to find out, whether the conditions on G needed to make G^Q strong can be relaxed using this fact.
- With essentially the same argument as for Theorem 3.2, one can prove a more general statement, which loosely speaking says that if the generator G cannot be predicted with probability better than $1 - \frac{1}{R(m)}$ for some polynomial R, then the generator $G^{mR(m)}$ is strongly unpredictable. In other words: by XOR-ing more "copies" of G, one can get away with a weaker assumption on the security of G.

It is now trivial to prove:

Corollary 3.1
If the Legendre Generator is weakly unpredictable, then the Jacobi Generator is strongly unpredictable □

Proof.
By the way we translate Legendre and Jacobi symbols into bits, it is clear that the Jacobi generator in fact outputs the xor of the output of several Legendre generators. We can therefore use Theorem 3.2 □

4. Emphirical Tests

A number of elementary statistical tests were performed on primes and sequences of various lengths. The primes were of length approximately 25, 50, 100, 200 and 400 bits, and 6 primes of each length were tested. The sequence length was fixed to $100 \cdot \log_2(p)$.

The sequences were generated using special purpose hardware (a FAP4- processor), and the tests included:

- A Chi-square test for equidistribution of subsequences of length 1 to 10. For each subsequence length a Chi-square value was computed based on the occurrences found. Representative results for the distribution of these Chi-square values can be found in Fig. 1.
- A Chi-square test on the distribution of block lengths. This produced results quite similar to those of the subsequence test.
- A test on the linear complexity of the strings. Using the Berlekamp-Massey algorithm, the linear complexity of prefixes of the Legendre sequences was computed. For a really random sequence, the linear complexity is expected to be close to half the sequence length [Ru]. Our sequences seem to fit nicely with this expectation. Fig. 2. shows a typical result.

No statistical weaknesses were found during these tests, and moving to composite numbers and Jacobi symbols produced no significant change in the results.

This can hardly be said to be surprising: as mentioned, all known results indicate, that a highly non-elementary test would be needed to detect any weakness in the Legendre generator.

Finally, let us remark that the tests mentioned here are just preliminary. Many other and more sophisticated tests with larger test material could (and should) be carried out.

5. Practical Implementation

If one wants to use special purpose hardware in implementing this system, using Gauss's Reciprocity Theorem for computing Legendre symbols hardly seems an attractive solution, at least judging from the hardware available today: most modular arithmetic chips are much better suited for exponentiation.

Computing Legendre and Jacobi symbols by exponentiation is cubic in the length of the modulus, and therefore slower asymptotically than generators based on squaring modulo a composite. In practice, the difference may not be so large, however, since it is not clear at all, that one must use primes large enough to make discrete log hard, for example.

If for example we use the Jacobi generator with 2 prime factors of size about 25 bits, this produces an effective key space of more than 90 bits, which is certainly enough to prevent exhaustive search.

If one is willing to use the same amount of hardware as would be needed for a 600 bit RSA implementation, one could compute 12 bits of the keystream in parallel, which with state of the art hardware would give a speed of about 100 Kbits pr. sec. Even use of just 2 25-bit slices would still give a speed of about 10 Kbits pr. sec.

6. Generalizations

The method we present can be generalized in several ways:

6.1. The Linear Congruence Method

First, one could consider, in stead of Legendre symbols of consecutive numbers, symbols for numbers generated by the well known Linear Congruence Generator, i.e. taking Legendre symbols for a sequence $a_1 \cdots a_n$, where

$$a_{i+1} = (m \cdot a_i + b) \bmod p,$$

for constants m and b, and a prime p. Note, however, that since

$$a_i = m^{i-1} a_1 + m^{i-2} b + \cdots + mb + b,$$

then by the multiplicative property of Legendre symbols,

$$(\frac{a'_i}{p})(\frac{b}{p}) = (\frac{a_i}{p}),$$

where $a'_1 = a_1 b^{-1}$, and $a'_{i+1} = ma'_i + 1$. So if this generalization is used in a cryptosystem, there is no point in including the choice of b in the key: up to a sign, all the possible sequences can already be obtained just by using $b=1$ and varying the starting point a_1. Variation of m, on the other hand, does seem to generate new sequences compared to the basic Legendre generator. This introduces a possibility of enlarging the key-space without using a larger prime.

One should take some care, however, in choosing m, as shown by the following Lemma:

Lemma 6.1
The period of the sequence defined by $a_1 = a$ and $a_{i+1} = (ma_i + b) \bmod p$ for a prime p is

p if $m=1$

$ord(m)$ if $m \neq 1$, and $a \neq \frac{-1}{m-1}$

1 otherwise,

where $ord(m)$ denotes the order of m as element in Z_p^*.

Proof.
The $m=1$-case is trivial. For the other cases, use the recurrence

$$a_{n+k} = (m^k a_n + \frac{m^k - 1}{m-1}) \bmod p$$

and elementary number theory □

Thus, the starting point for the sequence should be chosen different from $\frac{-1}{m-1}$, and $ord(m)$ should be large. This can be ensured by choosing p such that the factorization of $p-1$ is known, computing the order of a candidate m from this, and discard low-order m's. In practice, however, it is probably better to construct p such that $p-1$ has a large prime factor q. Then an m chosen uniformly from $]0 \cdots p-1]$ will have order divisible by q with probability $1-q^{-1}$.

Finally, let us remark that the results of Perron (see Section 2) for the basic Legendre sequences are easily seen to generalize to sequences generated by the method from this section.

6.2. Using Other Character Values

Another interesting idea is to consider other characters than the quadratic one. Such character values can also be produced easily by exponentiation: If q is a divisor in $p-1$, then $a \rightarrow a^{(p-1)/q}$ is a surjective homomorphism from Z_p^* to G, the subgroup of order q. By choosing some 1-1 correspondence between elements of G and the set of complex q'th roots of unity, each element in G corresponds to one of the q possible values of the corresponding character of Z_p^*. These elements can be represented by bit strings of length approximately $\log_2(q)$. We can therefore construct a generator by computing $a^{(p-1)/q}$ for consecutive values of a, and at each point output the corresponding bit string. Clearly, there is a limit to how large q can be chosen before the generator becomes insecure (just consider $q=p-1$!). Determining the maximal useful value of q will be an interesting field for new research. This problem can be thought of as corresponding to that of finding out how many of the least significant RSA-bits are cryptographically secure (see for example [MiSc]).

7. Conclusion and Open Problems

We have presented a new pseudorandom bit generator, based on a number theoretic problem, the complexity of which may be unrelated to the well known candidate hard problems in number theory.

We have seen that to prove the cryptographic strength of the generator, it is enough to prove that Legendre Sequences are weakly unpredictable.

A number of open problems remain, however:

- Are Legendre sequences weakly unpredictable?
- Is the complexity of predicting Legendre sequences related to other number theoretic problems?
- Are other characters than the quadratic one usable for pseudo random generators?

References

[Ba] Bach: "Realistic Analysis of Some Randomized Algorithms", Proc. of STOC 87.

[BlMi] Blum and Micali: How to Generate Cryptographically Strong Sequences of Pseudorandom Bits", SIAM J. of Comp., vol.13, 1984, pp.850-864.

[BoHi] Boppana and Hirschfeld: "Pseudorandom Generators and Complexity Classes", Manuscript, MIT, 1987.

[Bu] Burde: "Verteilungseigenschaften von Potenzresten", J. Reine Angev. Math 249, pp.133-172, 1971.

[Bur] Burgess: "The Distribution of Quadratic Residues and non-Residues", Mathematica 4 (1957), pp.106-112.

[Da] Davenport: "On the Distribution of Quadratic Residues (mod p)", J. London Math. Soc., 8 (1933), pp.46-52.

[El] Elliott: "A Restricted mean value Theorem", J. London Math. Soc.(2), 1 (1969), pp.447-460.

[Kr] Kranakis:"Primality and Cryptography", Wiley-Teubner Series in Computer Science, 1986.

[MiSc] Micali and Schorr: "Super-Efficient, Perfect Random Number Generators", these proceedings.

[Pe] Perron: "Bemerkungen uber die Verteilung der quadratischen Reste", Math. Z. 56 (1952), pp.122-130.

[Ru] Rueppel: "Linear Complexity and Random Sequences", Proc. of EuroCrypt 85, pp.167-191, Springer.