Jacobi Symbol Are Harder To Predict Than Legendre Symbol In Sylar Encryption System

Rajesh Kumar Sinha[1] and Tanwir Uddin Haider[2]

[1] Department of Mathematics, National Institute of Technology, Patna
Bihar, India

[2] Department of Computer Science, National Institute of Technology, Patna
Bihar, India

Abstract

A Stream cipher which is not based on LFSR's for creating random bit patterns is just like the Sylar Encryption System which use Jacobi symbols for generation of pseudo-random bit sequences. There is no restriction on the size of the secrete key which is shared, so there is no need of rewriting the code in case of block ciphers and other stream, there is need to change the size of the key for enhanced protection. Sylar Encryption System gives a near possible mathematically proven security as in case of one–Time Pads. Sylar encryption gives the advantage of computing multiple keys of the pseudo-random pad at the same time and thus support parallelism which most stream ciphers do not support. So computers which have multiple cores in the main memory can be used more efficiently and thereby contribute in increasing in the speed of encryption and decryption.

Keywords: *Jacobi Symbols, Legendre Symbol, LFSR, encryption.*

1. Introduction

Jacobi sequences are harder to predict than Legendre sequences, let Q be a polynomial with the security parameter value 1, the Legendre generator taken as input (seed) a randomly chosen k -bit prime p and a uniformly chosen k -bits number a . It produces as output the Legendre sequence modulo p with starting point a and length $p(k)$ where Legendre symbols are translated into bits such that -1 responds to a 1-bit, while 1 corresponds to a 0-bit. This sequence will be denote $L(p,a)$. Similarly, Jacobi generator P and Q be polynomials then with security parameter value 1, the Jacobi generator taken as input $Q(k)$ randomly chosen k -bit prime $p(1)$ ….. $p(k)$ and $Q(k)$ uniformly chosen k -bit number $a(1),.....a(k)$. Let a be chosen, such that a is congruent to $a(i)$ modulo $p(i)$ for $i=1....Q(k)$. The generator produces as output the Jacobi sequence modulo n with starting point a and length P, 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 [1] 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 [2]-[8]. Thus, informally taking, all we have to do in order to prove the strength of our generator is to show hard problem. This can also stated using Levin's concept of isolation 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 which takes input are chosen from a finite set X, where $\{x_m\}$ is a family of finite element and m as a security parameter. The output G(x) is a bit string use i^{th} bit is denoted by $G(x_i)$. We now have the following more formal definition of next bit-security. Thus generator G is said to be strongly unpredictable, if for all polynomials P and probabilistic circuits C holds only for finite m.

The Jacobi symbol is a generalization of the Legendre symbol. It is theoretical interest in modular arithmetic and other branches of number theory, but its main use in computational number theory, especially primarily testing and integer factorization; these in turn are important in cryptography.

Let $\left(\frac{a}{p}\right)$ represent the Legendre symbol, defined for all integers a and all odd primes p by

$$\left(\frac{a}{p}\right)=\begin{cases}0 \ \ if\ a\equiv(\bmod p)\\ +1 \ \ if\ a\neq 0\,(\bmod p)\ \ and\\ \text{some int eger } x, a\equiv x^2\,(\bmod p)\\ -1 \ \ if\ there\ \ is\ no\ such\ x\end{cases} \quad (1)$$

Following the normal convention for the empty product. For any integer *a* and any positive odd integer n the Jacobi symbol the product of the Legendre symbols corresponding to the prime factors of n

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{\alpha_1}\left(\frac{a}{p_2}\right)^{\alpha_2}\left(\frac{a}{p_3}\right)^{\alpha_1}.....\left(\frac{a}{p_k}\right)^{\alpha_k}, \quad (2)$$

These facts, even the reciprocity laws, are s traight forward deduction from the definition of the Jacobi symbol and the corresponding properties of the Legendre Symbol. But Jacobi symbols are defined when the numerator (upper argument) is an integer and the denominator (lower argument) is a positive odd integer.

If *n* is (an odd) prime, then the Jacobi symbol $\left(\frac{a}{n}\right)$ is also a Legendre symbol.

If

$$a\equiv b(\bmod n)\ .....then\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right) \quad (5)$$

Like the Legendre symbol,

If

$$\left(\frac{a}{n}\right)=-1\ \ then\ a\ is\ quadratic\ non-residue\,(\bmod n) \quad (3)$$

If a is a quadratic residue (mod n) then

$$\left(\frac{a}{n}\right)=1 \quad (4)$$

But, unlike the Legendre symbol if $\left(\frac{a}{n}\right)=1$ then a may or may not be a quadratic residue (mod n).

This is because for a to be a residue (mod *n*) it has to be a residue modulo every prime that divides *n*, but the Jacobi symbol will equal one if for example a is a non-residue for exactly two primes which divides *n*.

This formula lead to an efficient algorithm for calculating the Jacobi symbol, analogous to the

$$\left(\frac{a}{n}\right)=\begin{cases}0\ if\ \gcd(a,n)\neq 1\\ \pm 1\ \ if\ \gcd(a,n)=1\end{cases} \quad (6)$$

$$\left(\frac{ab}{n}\right)=\left(\frac{a}{n}\right)\left(\frac{b}{n}\right),\ so\ \left(\frac{a^2}{n}\right)=1\,(or\ 0) \quad (7)$$

$$\left(\frac{a}{mn}\right)=\left(\frac{a}{m}\right)\left(\frac{a}{m}\right),\ so\ \left(\frac{a}{n^2}\right)=1\,(or\ 0) \quad (8)$$

The law of quadratic reciprocity:

If *m* and *n* are odd positive integers, then

$$\left(\frac{m}{n}\right)=\left(\frac{n}{m}\right)(-1)^{\frac{m-1}{2}\frac{n-1}{2}}=\begin{cases}\left(\frac{n}{m}\right)\ \ if\ n\equiv 1(\bmod 4)\ or\ m\equiv 1(\bmod 4)\\ -\left(\frac{n}{m}\right)\ \ if\ n\equiv m\equiv 3(\bmod 4)\end{cases} \quad (9)$$

and its supliments

$$\left(\frac{-1}{n}\right)=(-1)^{\frac{n-1}{2}}=\begin{cases}1\ \ if\ n\equiv 1(\bmod 4)\\ -1\ \ if\ n\equiv 3(\bmod 4)\end{cases} \quad (10)$$

$$\left(\frac{2}{n}\right)=(-1)^{\frac{n^2-1}{8}}=\begin{cases}1\ \ if\ n\equiv 1,7(\bmod 8)\\ -1\ \ if\ n\equiv 3,5(\bmod 8)\end{cases} \quad (11)$$

Euclidean algorithm for finding GCD of two numbers. The 'numerator' is reduced modulo the 'denominator' using rule (2). Any multiples of 2 are pulled out using rule (4) and calculated using rule (8). The symbol is flipped using rule (6), and the algorithm recourses. Until the 'numerator' is 1 (by rule (4)) or 2 (by rule (8)), or the 'numerator' equals the 'denominator' (by rule (3)).

The Legendre symbol $\left(\frac{a}{p}\right)$ is only defined for odd prime *p*. It obeys the same rule as the Jacobi symbol i.e reciprocity and the supplementary formulas for $\left(\frac{-1}{p}\right)$ and $\left(\frac{2}{p}\right)$ and multiplicatively of the 'numerator'.

Let us consider 9907 is prime and calculate $\left(\frac{1001}{9907}\right)$.

Using the Legendre symbol:

$$\left(\frac{1001}{9907}\right)=\left(\frac{7}{9907}\right)\left(\frac{11}{9907}\right)\left(\frac{13}{9907}\right) \quad (12)$$

$$\left(\frac{7}{9907}\right) = -\left(\frac{9907}{7}\right) = -\left(\frac{2}{7}\right) = -1 \quad (13)$$

$$\left(\frac{11}{9907}\right) = -\left(\frac{9907}{11}\right) = -\left(\frac{7}{11}\right) = \left(\frac{11}{7}\right) = \left(\frac{4}{7}\right) = 1 \quad (14)$$

$$\left(\frac{13}{9907}\right) = -\left(\frac{9907}{13}\right) = -\left(\frac{1}{13}\right) = 1 \quad (15)$$

$$\left(\frac{1001}{9907}\right) = -1 \quad (16)$$

Using the Jacobi symbol

$$\left(\frac{1001}{9907}\right) = \left(\frac{9907}{1001}\right) = \left(\frac{898}{1001}\right) = \left(\frac{2}{1001}\right)\left(\frac{449}{1001}\right) = \left(\frac{449}{1001}\right)$$

$$= \left(\frac{1001}{449}\right) = \left(\frac{103}{449}\right) = \left(\frac{449}{103}\right) = \left(\frac{37}{103}\right) = \left(\frac{103}{37}\right)$$

$$= \left(\frac{29}{37}\right) = \left(\frac{37}{29}\right) = \left(\frac{8}{29}\right) = \left(\frac{4}{29}\right)\left(\frac{2}{29}\right) = -1 \quad (17)$$

2. Proposed Work

Both the communicating parties have a s hared secret key. This key (p) can be arbitrarily long without any fixed size and has no restriction whether the number should be a prime number or not. An initialization vector (IV) is also sent in plain text so as make the bit sequence more random. This secret key [9]-[12] is then truncated by fixed number of bits. This truncated secret key (a) is then multiplied with the first half of the IV and added to second half of the IV modulo (p). The end result (a) is used as the seed of the random bit generator. For encryption of n^{th} bit the Jacobi symbol (a + (n-1)/p) is calculated and is XOR-ed with the corresponding bit at receiver's end the same process is repeated and the message is recovered from the encryption.

Unlike other stream and block cipher there is no restriction on the size of the of the of the secret key which is shared, so there is no need of rewriting the code in case we have to change the size of the key for enhanced protection; Sylar encryption system gives a near possible mathematically proven security as in case of One-Tme pads. Sylar encryption gives the advantage of computing multiple keys of the pseudo-random pad at the same-time and thus support parallelism which most stream ciphers do not support. So computers which have multiple cores in the main memory can be used more efficiently and thereby contribute in increasing in the speed of encryption and decryption.

3. Pseudo-Code

```
P:= Secret_key ;

a′:= Truncate (p>>100) //binary division done 100 times
over the secret shared key written in binary

Size= (log (IV)/2

a′:= a′ * (IV's Most significant size bits)
//IV is initialization vector

a := a′  + (IV's Least  significant size bits)
          //This value a is used as th seed for this session
            for i=1

if stream exists
       do
J:= Jacobi (a+i-1),  p);
Xor  (plaintext bit (i) , J) ;
//encrypt if sender, decrypt if receiver
        i:=i+1;
        else exit
```

Pseudo code for Jacobi symbol :

```
Jacobi (a,b)  ( If (b<=0 or (b mod 2)==0) return(0);
      J=1;
      If (a<0)
        { a= -a ;
          If ((b Mod 4== 3)   j = -j; }

 Do while  (a!=0) {

 Do while  (( a Mod 2 ) == 0)
 {                                  //Process factors of 2:
      Jacobi (2,b) = -1 if (b=3, 5 (mod 8))
      a = a/2

 If (( b Mod 8) =3 or (b Mod 8) == 5)  j = -j ;}
// Quadratic reciprocity: Jacobi (a,b) = - Jacobi (b,a) if
a=3, b=3 (mod 4)
Interchange (a,b) ;

If ((a Mod 4) == 3 and (b Mod 4) == 3) j = -j;
         a = a Mod b;}

 If (b == 1)   (return (j)}
         else return (0) ; }
```

4. Conclusions

The difference between the two calculations is that when the Legendre symbol is used the 'numerator' has to be factored into prime powers before the symbol is flipped. This makes the calculation using the Legendre symbol significantly slower than the one using the Jacobi symbol, as there is hard to know polynomial time algorithm for factoring integers. I n fact, this is why Jacobi introduced the symbol.

References

[1] A division less form of the Schur Berlekamp-Massey algorithm by Christopher J.Zarowski.
[2] Cryptographic secure Pseudo-Random bits generation by Pascal Junod.
[3] Colomb,S. Shift Register sequences, Aegean Park Press, 1982.
[4] Elaine Barker and John Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST Special Publication 800-90. Revised March 2007.
[5] H. Gilbert and M. Minier, "New results on the pseudorandomness of some block cipher constructions", In M. Matsui (Ed.) Fast Software Encryption - FSE 2001, Lecture Notes in Computer Science 2355, Springer-Verlag, 2002, 248–266.
[6] M.Grangetto, E. Magli and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding", IEEE Transactions on M ultimedia, vol. 8, no. 5, 2006.
[7] Md.Tanwir Uddin Haider & Rajesh Kumar Sinha et. al.IJCSE Vol. 02, No. 09, 2010, 2836-2837.
[8] Massey,J. "Shift Register synthesis and bch decoding" IEEE Transactions on Information Theory, Vol. 15. No., pp 122-127, 1969.
[9] Menezeset al. Handbook of Applied Cryptography.
[10] Modified Berlekamp-Massey algorithm for approximating the k-error linear complexity of Binary sequences by Alexandra Alecu and Ana Salagean.
[11] Schneier Bruce, Applied Cryptography, 2nd edition, Addison Wiley, New York 1996.
[12] Lidl R and Niederreriter H. Encyclopedia of Mathematics and its Applications, vol.20, Cambridge University Press,(1996)