Journal of Computer Science 4 (12): 1056-1060, 2008
ISSN 1549-3636

A New Computation Algorithm for a Cryptosystem Based on Lucas Functions

[1]Mohamed Othman, [2]Esam M. Abulhirat, [3]Zulkarnain Md Ali,
[4]Mohd Rushdan Mohd Said and [1]Rozita Johari
[1]Faculty of Computer Science and Information Technology,
University Putra Malaysia, 43400 Serdang, Selangor, Malaysia
[2]Department of Human Resources, Science and Technology, African Union Organization,
African Union Headquarters, P.O. Box 3243, Roosevelt Street, W21K19, Addis Ababa, Ethiopia
[3]Faculty of Technology and Information Science,
University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
[4]Institute for Mathematical Research, Faculty of Science,
University Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Abstract: Most of public-key cryptosystems rely on one-way functions. The cryptosystems can be used to encrypt and sign messages. The LUC Cryptosystem is a cryptosystem based on Lucas Functions. The encryption process used a public key which was known publicly and the decryption used a private key which was known only by sender and receiver of the messages. The performance of LUC cryptosystem computation influenced by computation of V_e the public key process and V_d the private key process. Very large scales of computations and timing overhead involved for large values of e and d. We are presenting the so-called Doubling with Remainder compared to the existing technique. It shows better performance in LUC computations by reducing time consumed in its computations. The experimental results of existing and new algorithm are included.

Key words: Cryptography, Computation algorithm

INTRODUCTION

Since the concept of public-key cryptosystems was first published in[2], there are a lot of possible trapdoor functions proposed. Probably, the best known and most widely used trapdoor function is the exponentiation based cryptosystems. This system is known as RSA public key cryptosystems[7].

After two decades, the authors in[8] introduced a public key based on Lucas Functions instead of exponentiation based. This system is believed offers good alternative to the RSA.

Lucas Functions are special form of second-order linear recurrence relations using large public integer as modulus. The key distribution concept[2] can be constructed using Lucas Functions. Another interested point is its cryptographic strength. It is much stronger than or at least strong as the exponentiation based systems.

The performance of cryptographic functions is the most critical issues. The effectiveness determined by the performance of its computation. Smith and Lennon[8] concluded that, it has big complications in terms of storage and timing overheads. With very big number e (V_e), the encryption of LUC Cryptosystem cost a huge time and space.

On the other hand, several researchers on fast exponentiation evaluation for RSA have been proposed. Knuth in[5] presented a simple square-multiply method based on the binary representation of the exponent.

Similarly, some researchers worked on fast computation technique for Lucas Functions. Yen and Laih[11] are among the first to propose an efficient algorithms to compute the Lucas Function. They showed the way to reduce the number of multiplications when evaluating the Lucas Function by shortens the length of the LUC Chain. They also proposed two algorithms by scanning the binary form of the exponent and sequentially evaluate the Lucas sequences. A LUC Chain is based on Addition Chain where has been discussed in detail in[5].

Chiou and Laih in[1] proposed another fast algorithm in which their computation techniques that was slightly better than works in[11]. In other related study[9] also proposed another algorithm. Joye and

Corresponding Author: Mohamed Othman, Faculty of Computer Science and Information Technology,
University Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Quisquater in[4] proposed a technique to compute both U_n and V_n.

In this study, we proposed fast computation algorithm that was based on Doubling Step. Doubling Steps technique is discussed in[10]. Our algorithm concentrates on how to use a remainder sequence in order to organize the computations and finally obtain the required value of V_n.

We proposed a Doubling with Remainder technique. Our technique follows these steps:

- Generate a remainder sequence
- Use this sequence to direct the LUC cryptosystems computations

Lucas function and LUC cryptosystems: Lucas functions can be seen as generalized linear recurrences. A Lucas Function is a sequence of integers V_n defined as $V_0=2$, $V_1=P$, $V_n=PV_{n-1}-QV_{n-2}$ for $n\geq 2$. This dentition referred as n^{th} order linear recurrence as stated by[6].

The other sequence in Lucas Function is known as U_n. It is defined as $U_0=0$, $U_1=1$, $U_n=PU_{n-1}-U_{n-2}$ for $n\geq 2$. We know that for U_n, if the parameters are selected as P=1 and Q=-1, the sequence is the well known Fibonacci sequence.

Noted that, the sequence V_n with Q=1 is usually used to devise cryptosystems by cryptographers.

Encryption and decryption for LUC cryptosystems: It is uses two keys (e,N) and (d,N) which works in pairs for encryption and decryption respectively. A ciphertext, C is obtained by $f(P)=Ve(P,1)(\text{mod } N)\equiv C(\text{mod } N)$, where V_e is a Lucas Function, or the e^{th} term of the Lucas sequence. It is derived from the second order recurrence relation:

$$V_n = PV_{n-1}-QV_{n-2} \tag{1}$$

Initial conditions $V_0 = 2$ and $V_1 = P$. Meanwhile, the decryption function is applied to ciphertext C by $f(C)=V_d(C,1)=V_d(V_e(P,1),1)=V_{ed}(P,1)\equiv P(\text{mod } N)$. This function will recover the original message, P. We can use Eq. 1 in existing method.

There are two factors that give impact to the performance and behavior of calculation of LUC Cryptosystems:

- Computation of V_e and V_d looks complicated for large values of e and d
- The private key d has to be recomputed for each block of message

An existing algorithm: We can use Eq. 1 to design this algorithm. We used SL to denote the existing algorithm. It is very simple technique. Let calculate V_{1103}. Using Eq. 1, we first compute V_2 using V_1 and V_0. This computation continues with V_3, where we have V_2 and V_1. After we get V_3, we need to calculate V_4, until finally we compute V_{1103}. In general, the computation of V_n is done by computation of V_2, V_3, …, V_{n-1} and finally V_n. Algorithm 1 shows an existing algorithm in[8]. Note that, e is public key and P is message.

Algorithm 1: Existing Algorithm:
1. Input: e, P, $V_0 = 2$ and $V_1 = P$
2. Output: V_n.
3. $V_f = V_0$ and $V_g = P$ and Q = 1
4. While (k! = e)
   a. $V_j = PV_f - QV_g$
   b. $V_g = V_f$
   c. $V_x = V_j$
   d. $V_f = V_x$
   e. k++
5. End While

Properties of Lucas Functions: Williams[10] introduced a method of factorization which is known as "ρ+1 factorization" technique. He suggested that Lucas Functions can be used to find a prime divisor ρ of N when ρ+1 have only small prime factors. Smith and Lennon[8] then used some Lucas Functions relation in their public-key cryptosystems.

Some of them are:

$$V_{2n} = V_n^2-2 \tag{2}$$

$$V_{2n+1} = PV_n^2-QV_nV_{n-1}-PQ^n \tag{3}$$

$$V_{2n-1} = V_nV_{n-1}-PQ_{n-1} \tag{4}$$

$$V_n^2 = DU_n^2+Q^n \tag{5}$$

$$2V_{n+m} = V_nV_m+DU_nU_m \tag{6}$$

These properties are not limited. More results on another property can be found in[10]. Horster *et al.*[3] have also introduces another relations on Lucas Functions.

A proposed algorithm: For the purpose of this study, we only focused on Eq. 2-4. We are sure that those selected equations are very useful to reduce a number of computation steps needed to compute the sequences of V_n for LUC Cryptosystems. In this study we are only manipulating the Doubling Steps technique.

Our algorithm concentrates on how to reduce as much as multiplication processes. Because, we are sure that the reduction of multiplication processes can reduce time consumed for calculating V_n.

We give a name to our algorithm as Doubling with Remainder (DwR). Here V_n is either V_e or V_d. We have the following strategies to achieve high speed of computation technique:

- Generate the remainder sequence. This is considered as a part 1 of this proposed algorithm. It is relatively easy as we generate a remainder for any give value of n.
- Use the generated remainder sequence to direct the LUC Cryptosystems computation and it is considered as part 2 of the algorithm

The Algorithm 2 shows how to use the remainder sequence.

Algorithm 2: Algorithm to Use Remainder Sequence:

1. Input: Array k, V_0=2, V_1=P and N
2. Output: V_n
3. Calculate V_2, V_3 and V_4 using Eq. 1
4. If (k[0] = 1)
   Calculate $V_{2n} = V_3$ and $V_{2n+1} = V_4$
5. Else
   Calculate $V_{2n}=V_2$ and $V_{2n+1} = V_3$
6. End If
7. For j = x to 2
   If k[x] =1
     i. $V_t = V_{2n+1} * P - V_{2n}$ (mod N)
     ii. $V_{2n} = V_{2n+1} * V_{2n+1} - 2$ (mod N)
     iii. $V_{2n+1} = V_{2n+1} * V_t - P$ (mod N)
   Else
     i. $V_{2n+1} = V_{2n} * V_{2n+1} - P$ (mod N)
     ii. $V_{2n} = V_{2n} * V_{2n} - 2$ (mod N)
   End If
   x = x-1
8. End For
9. If k[x-1] =1
   $V_n = V_{2n+1}$
10. Else
   $V_n = V_{2n}$
11. End If

The calculation of private key d: The private key d can be computed from Eq. 7:

$$de \equiv 1 \pmod R \tag{7}$$

R = LCM((p-(D/p),q-(D/q)). Note that, LCM is Least Common Multiple, D is discriminant for either prime p or prime q. An e is public key which is known publicly.

The following steps show the computation of private key d:

- Find discriminant D, such that $D = C^2-4$, where D is discriminant and C is ciphertext.
- Find Legendre Symbols for (D/p) and (D/q). Here we could have four possible values of Legendre Symbols. We used LS(D/p) to denote Legendre Symbols for (D/p).
- Find LCM for either LCM((p+(D/p),q+(D/q)), LCM((p+(D/p),q-(D/q)), LCM((p-(D/p),q+(D/q)), or LCM((p-(D/p),q-(D/q))

In Algorithm 3, the function with the name of ExtendedEuclid() is the Extended Euclid Algorithm. It is a classical computational number theory that can be found in most numbers theory text books.

The LCM is also classical computational number theory which was known as Least Common Multiple. Algorithm 3 has been tested with the maximum number of digit up to 2000 digits.

Once we got private key d, we can compute V_d as the same way we compute V_e to get back the original messages, P. We recorded time consume for both encryption and decryption processes. Algorithm 3 shows how to compute private key d.

Algorithm 3: Algorithm to Compute d:

1. Input: C, p and q
2. Output: d
3. Calculate $D = C^2 - 4$
4. Calculate LS(p) = (D/p) and LS(q) = (D/q)
5. If LS(p) = -1 And LS(q) = -1
   a. X = p+1
   b. Y = q+1
6. End If
7. If LS(p) = 1 And LS(q) = -1
   a. X = p-1
   b. Y = q+1
8. End If
9. If LS(p) = -1 And LS(q) = 1
   a. X = p+1
   b. Y = q-1
10. End If
11. If LS(p) = 1 And LS(q) = 1
   a. X = p-1
   b. Y = q-1
12. End If
13. R = lcm(X,Y)
14. ExtendedEuclid(d,R,e)

Implementations: Algorithm 1, 2 and 3 are implemented in C Language. We used SL to denote existing algorithm and DwR to denote new algorithm.

The computation time for both algorithms is our main results. Last but not least we also discuss a difference between two algorithms.

We implemented the SL to compute the value of V_n. We can only start the computation with V_2, because we only know V_0 and V_1. To calculate V_3, we need to know V_1 and V_2. It followed by V_4 because we know the values of V_2 and V_3. This process continues until we achieved the calculation of V_n. The computation will be V_2, V_3, V_4, V_5, ... V_n. Very simple computation steps involved. The algorithm is shown in Algorithm 1.

Meanwhile, the remainder sequence is the heart of DwR. For example, compute V_{1103}. In this case n = 1103. Table 1 shows how to generate remainder sequence. Use this sequence to direct the doubling steps. The illustration on using remainder sequence is shown in Table 2.

Table 1: Illustration on generating remainder sequence

Value n	x	k[x] = y
1103	0	k[0] = 1
551	1	k[1] = 1
275	2	k[2] = 1
137	3	k[3] = 1
68	4	k[4] = 0
34	5	k[5] = 0
17	6	k[6] = 1
8	7	k[7] = 0
4	8	k[8] = 0

Table 2: Illustration on using the remainder sequence

k[x]	x	V_{2n}	V_{2n+1}
k[8]	0	V_4	V_5
k[7]	0	V_8	V_9
K[6]	1	V_{16}	V_{17}
k[5]	0	V_{32}	V_{33}
k[4]	0	V_{64}	V_{65}
k[3]	1	V_{128}	V_{129}
k[2]	1	V_{256}	V_{257}
k[1]	1	V_{512}	V_{513}
k[0]	1	V_{1102}	V_{1103}

RESULTS

Table 3-5 show time consumed SL and DwR for both encryption (Enc) and decryption (Dec) processes. As a result, the algorithm reduces iterations, speedup the computation and at the same time reduces the computation time.

The results in Table 3-5 are based on the running time for each algorithm in C language in Windows XP Environment, Crusoe Processor TM5800 with 658 MHz and 240 MB of RAM. All computation times are in seconds.

DISCUSSION

The most important feature to discuss here is the total number of iterations in the computation of V_n. in order to compute V_{1103}, SL Algorithm required exactly 1103 iterations (refer to Algorithm 1) while DwR only need 8 iterations (refer to Algorithm 2 and also Tables 1 and 2). Surely, for the bigger size of public key, we suffered huge iterations in SL algorithm.

Table 3: The computation time in second for each algorithm for different key size

Key size E	Enc SL	Enc DwR	Private key d	Dec SL	Dec DwR
19	320	9	199	56815	160
79	1848	37	199	61600	174
159	3227	65	199	69644	197
239	4793	97	199	80519	227
559	16212	329	199	89824	254
719	34397	711	199	94769	268

Table 4: The computation time in second for each algorithm for different primes size

Primes size p and q	Enc SL	Enc DwR	Private key d	Dec SL	Dec DwR
50	1743	35	99	15390	43
100	3194	65	198	21829	161
110	3500	71	219	68937	195
160	9278	188	319	260788	738
180	10481	213	359	330412	935
220	11912	242	437	445442	1261
280	15400	313	559	76321	2160
300	18201	370	599	930909	2635

Table 5: The computation time in second for each algorithm for different message size

Message size P	Enc SL	Enc DwR	Private key d	Dec SL	Dec DwR
20	1238	25	399	368469	1043
80	1321	26	398	370120	1047
160	1345	27	398	377754	1041
190	1375	28	398	373905	1058
250	1828	37	398	381578	1080
330	1922	39	399	386299	1093

In the same situation, DwR required small number of iterations. The calculation of $V_{1111111111111111103}$, obviously shows SL technique needs exactly 1111111111111111103 iterations. Meanwhile, for DwR the computations were only required 59 iterations. The generating of remainder sequence would reduce numbers of modular multiplications.

Therefore, the computation time is reduced in the proposed method. The remainder sequences achieved less numbers of modular multiplications.

The computation of private key d is possible because we know the values of prime p and q. We also know the value of ciphertext C and public key e. All these values are needed in the computation of private key. In real world, it is not easy to compute private key d.

In our experiments, the times recorded for decryption process also include the time for calculation of Legendre Symbols, Lease Common Multiple and Extended Euclid Algorithm. These three processes required approximately 35% of total decryption process. If we can construct or apply any fast computations algorithm for these three processes, we are sure that we can reduce a computation time.

CONCLUSION

We can speed up the LUC Cryptosystem computation by Doubling with Remainder. The comparison as shown in Table 3-5 proved that the speed can be increased by reducing the number of steps of multiplication. It makes the LUC cryptosystem computations more efficient for security implementation.

Likewise, the reduction of multiplications with the DwR algorithm, enabled us to achieve a good reduction of computation time. It also leads to high reduction in the multiplications required for both the encryption and decryption operations without sacrificing the key size of LUC cryptosystem security.

However, the construction of shorter sequence than the remainder sequence could be interesting research topics. Another interesting research topic is the reduction of some modular multiplications in Lucas Functions itself.

ACKNOWLEDGEMENT

Researchers are glad to thank many people on their suggestions and comments throughout this research. The third author would like to thank the Government of Malaysia and UKM for awarding PhD scholarship.

REFERENCES

1. Chiou, S. and C. Laih, 1995. An efficient algorithm for computing the LUC chain. IEEE Proc. Comput. Digital Tech., 147: 263-265.
2. Diffie, W. and M.E. Hellman, 1976. New direction in cryptography. IEEE Trans. Inform. Theor., 22: 644-654.
3. Horster, P., M. Michels and H. Petersen, 1996. Digital signature schemes based on lucas functions. Proceeding of the 1st International Conference on Communications and Multimedia Security, September 1995, Chapman and Hall, Graz, 178-190
4. Joye, M. and J.J. Quisquater, 1996. Efficient computation of full lucas sequences. IEEE Elect. Lett., 32: 537-538.
5. Knuth, D.E., 1998. The Art of Computer Programming, Vol 2 (Seminumerical Algorithms), Third Edition, Addison-Wesley, Reading, Massachusetts : 356-379.
6. Ribenboim, P., 1996. The New Book of Prime Number Records, Third Edition (February 1996), Springer, New York : 53-74
7. Rivest, R.L., A. Shamir and L.M. Adleman, 1978. A method for obtaining digital signatures and public-key Cryptosystems. Commun. ACM., 2: 120-126.
8. Smith, P. and M. Lennon, 1993. LUC: A new public key system. Proceeding of the 9th IFIP Symposium on Computer Security, North-Holland, Amsterdam (1993), Elsevier Science Publishers: 103-117.
9. Wang, C.T., C.C. Chang and C.H. Lin, 1999. A method for computing lucas sequences. Int. J. Comput. Math. Appl., 38: 187-196.
10. Williams, H.C., 1982. A ρ+1 method of factoring. Math. Comput., 39: 225-234.
11. Yen, S. and C. Laih, 1995. Fast algorithm for LUC digital signature computation. IEEE Proc. Comput. Digital Tech., 142: 165-169.