An Improvement of LUC$_2$ Cryptosystem Algorithm Using Doubling with Remainder


1Department of Communication Technology and Network; 2Department of Mathematics, University Putra Malaysia, 43400 UPM Serdang, Selangor D.E., Malaysia.
3Department of Computer Sciences, Faculty of Technology and Information Sciences, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor D.E., Malaysia.

Corresponding author’s e-mail: mothman@fsktm.upm.edu.my

Abstract—The major factor that influences the performance of the LUC public-key cryptosystem is the computation of $V_n$ and $V_0$, a public and private key, respectively. Its involve a huge steps of computations for large values of $e$ and $d$. We concentrated our discussion on how to utilize and manipulate the doubling step technique for an efficient LUC$_2$ computation. Therefore, we proposed the so-called Doubling with Remainder technique. It shows a better performance in LUC$_2$ computations and also a great reductions of time consume for computations. The experimental results for sequential, doubling steps and doubling with remainder are also included.

Index Terms—Public key Cryptosystems, LUC Cryptosystems and Algorithms.

I. INTRODUCTION

Via the digital world and the cyber space, several limitations of fast communication have already been eliminated. Such algorithms, models and systems are build purposely to create an ideal method to provide a secure environment for better optimization of the electronic-connected world. For this kind of situations, cryptography already accepts the challenge and plays the main role in modern and secure communication world.

Probably, RSA cryptosystem is one of the most widely used algorithms. It is also accepted as the most attractive algorithms for public key cryptosystems. It works by computing a modular exponentiation of a message block to a very large power, then reducing this number to modulo $N$, where $N$ is the product of two large prime numbers, $p$ and $q$ [6]. RSA has dominated public-key encryption for a few decades until the advent of LUC cryptosystems. LUC is based on the same difficult mathematical problem as in RSA, where the exponentiation ciphers all based on the mathematical problem known as the Discrete Logarithm (DL). Otherwise, LUC cryptosystem uses the calculation of Lucas functions instead of exponentiation (used in RSA). Its computational approaches is still based on the analogous to the DL problem [2].

Such implementations of Lucas Functions ciphers are sometimes has a big complication in terms of storage and timing overheads. Therefore, we proposed a Doubling with Remainder method to overcome the complication of storage and time consuming. So that, it can speed up the LUC$_2$ computations. The properties of Lucas functions mirror those of exponentiation. It is already proved that LUC cryptosystem is cryptographically fast and secure as their exponentiation-based ancestors [7].

We organize this paper as follows; in section II and III we shall only include the minimal amount of background necessary to understand this paper. For systematic treatment, see the references [2] and [7]. Then, in section IV, V and VI, we shall discuss the existing method and also our proposed method. Finally, in VII we present all the experimental results.

II. BASIC DEFINITIONS

A second-order linear recurrence is a sequence of integers \{T$_n$\} defined by $T_0 = a$, $T_1 = b$, where $(a, b)$ are integers, and

$$T_n = PT_{n-1} - QT_{n-2},$$

where $P$ and $Q$ are relatively prime integers. If $P = 1 = Q$, then the sequence of numbers obtained by choosing $T_0 = 0$ and $T_1 = 1$. This sequences can be clarify as the well-known Fibonacci sequence [7].

There are two particular solutions of the general second-order linear recurrence relation which are in our particular interest. They are denoted by \{U$_n$\} and \{V$_n$\},

$$U_n = \frac{\alpha^n - \beta^n}{\alpha - \beta},$$
$$V_n = \frac{\alpha^n + \beta^n}{\alpha + \beta}.$$

Both will be the sequences of integers, since $U_0 = 0$, $U_1 = 1$, $V_0 = 2$, $V_1 = P$.

These sequences depend only on the integers $P$ and $Q$, and the terms are called the Lucas functions of $P$ and $Q$. They are sometimes written as $U_n(P, Q)$ and $V_n(P, Q)$, to emphasis their dependence on $P$ and $Q$ (see [5] and [7]). Note that if $N$ is any number, then

$$U_n(P \mod N, Q \mod N) \equiv U_n(P, Q) \mod N,$$

because this result is certainly true when $n = 0$ or 1, and for every $n \geq 2$,

$$U_n(P, Q) \mod N \equiv (P \mod N)(U_{n-1}(P, Q) \mod N) - (Q \mod N)(U_{n-2}(P, Q) \mod N),$$
so, the stated result follows by induction, similar to

\[ V_n \equiv V_0 \equiv 0 \mod N. \]

III. LUC₂ Encryption and Decryption Processes

LUC₂ public-key cryptosystem was developed by analogy with the RSA system. The LUC₂ cryptosystem (quadratic analogue of the RSA cryptosystem) uses two keys \( (e, N) \) and \( (d, N) \), which works in pairs for encryption and decryption, respectively [7]. Noted that through out of this paper, we only interested with the recursive function \( V_n \) with \( Q = 1 \). Therefore, a plaintext message \( P \) is encrypted to ciphertext \( C \) by

\[ f_{LUC}(P) = V_e(P, 1) \mod N \equiv C \mod N, \]

where \( V_e \) is a Lucas function, or is the \( e^{th} \) term of the Lucas sequence, derived from the second order recurrence relation

\[ V_{n+2} = PV_{n+1} - V_n. \]  \hfill (2)

With the initial values \( V_0 = 2 \) and \( V_1 = P \), this is known as the LUC public-key process, gives an encrypted message \( C = \tilde{V}_e \). Due to the symmetry in modular arithmetic; encryption and decryption are mutual inverses and commutative. Therefore, the decryption function is

\[ f_{LUC}(C) = V_d(C, 1) = V_d(V_e(P, 1), 1) = V_{ed}(P, 1) \equiv P \mod N, \]

and this recovers the original message \( P \).

The characteristic equation of the Lucas sequence introduced here is the quadratic \( x^2 - Px + 1 = 0 \) with the discriminant, \( D = P^2 - 4 \). The coefficient \( P \) is an integer, the discriminant is non-zero and the quadratic has distinct roots.

Next, a relatively large integer \( e \) is chosen so that \( e \) is relatively prime to

\[ (p - 1)(q - 1)(p + 1)(q + 1), \]

the block of message \( P \) to be encrypt must be chosen less than \( N \) \( (P < N) \) and relatively prime to \( N \). This is not a big restriction on \( P \). Because \( p \) and \( q \), are large enough that the probability of \( P \) being divisible by one of them is less than the probability of the secret key being revealed by some unforeseen event.

Finally, the selection of the decryption key \( d \) such that

\[ de \equiv 1 \mod S(N), \]

where,

\[ S(N) = \text{lcm} \left( \left( p - \left( \frac{D}{p} \right) \right), \left( q - \left( \frac{D}{q} \right) \right) \right), \]

\[ D = (\tilde{P})^2 - 4, \text{ and } \left( \frac{D}{p} \right), \left( \frac{D}{q} \right) \text{ are the Legendre symbols of } D \text{ with respect to } p \text{ and } q. \]

While, the \( \text{lcm} \) denotes least common multiple.

IV. Behavior of Computations

There are two major factors that would gives an impact to performance and behavior of calculation of LUC₂ public-key cryptosystem:

- 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.

The main concern here is dealing with the computation of \( V_e \) and \( V_d \). It is having almost the same degree of complexity as computation of powers (as in RSA). The double step formula is

\[ V_{2n} = V_n^2 - 2 \mod N \quad \text{where} \quad V_n = V_e(P, 1), \]

in the case in which we are interested, the double step techniques should be compared to the formula for the computation of powers of numbers needed in RSA cryptosystems,

\[ 2^2 \mod N = (P^2 \mod N)^2 \mod N. \]

The doubling step is already used in conjunction with other multiplication steps especially in the Russian peasant method of multiplication [4]. It is also an equivalent algorithm exists for the calculation of Lucas Function of high order [8]. We shall concentrate on doubling step in more details in section V.

In general, the total amount of computation for the LUC public-key cryptosystem is approximately equal to the amount needed for the RSA public-key cryptosystem [7].

The size of encryption keys are measured in bits and the difficulty of trying all possible keys grows exponentially with the number of bits used. This process can be doubles when adding one bit to the keys. It is also can be more than thousand if adding another ten bits. As the encryption keys become big, the computation process could cost maximum strength of computing power.

Yen and Laih [9] were proposed an efficient algorithm to reduce the number of multiplications when they evaluated the Lucas function. They concluded in order to speed up the computation, they need to shortened the length of the LUC chain. In other paper, Chiu and Laih [1] proposed a more efficient algorithm for computing the same LUC chain, based on a special type of the computations of Lucas sequence.

Another works that also related to this matter is done by Joye and Quisquater in [3] claimed that they have generalized the method to any type of Lucas sequences for asymmetric treatment. Here, we shall consider the important part of LUC₂ cryptosystem computations; to compute \( V_k(P, 1) \).

V. Computation of LUC₂ with Doubling Step

We use the binary representation of any number \( k \), which can be expressed as \( k = K_0 \), where

\[ K_j = \sum_{i=0}^{n-1} k_i 2^{i-j}, \quad k_i \in \{0, 1\} \text{ and } k_{n-1} = 1. \]

In 1982, Williams introduced a method of factorization which is known as \( p + 1 \) factorization technique (see [8]).
The author suggested that, by using Lucas functions, we can find a prime divisor \( p \) of \( N \) when \( p + 1 \) has only small prime factors. He also introduced the following equations,

\[
V_{2n} = V_n^2 - 2 \quad \text{and} \quad V_{2n+1} = V_nV_{n+1} - P
\]

\( V_n = V_n(P, 1). \) (3)

For an efficient computation of the LUC2 cryptosystem, the doubling-step is needed. It is also known as Doubling-Rule that has been used the same way as repeated squaring in the Russian peasant method of multiplication [4]. The algorithm reduce the value of multiplication steps to compute \( V_n \) for LUC2.

Another attempt has been proposed by Joye and Quisquater in [3]. Here, we illustrates that algorithm. This algorithm can reduces the number of steps needed for multiplication to compute the sequences of \( V_n \) for LUC2.

**Algorithm 1** Speed Up \( V_n \) computations (see [3]).

| Input : \( P, k \) | \( k = 2^s \sum_{i=0}^{n-1} k_i 2^{i-s} \) where \( (k_s) = 1 \). |
| Output : \( (V_n) \). |

\begin{align*}
V_0 &= 2, V_1 = P \\
\text{For } j \text{ from } n - 1 \text{ to } s + 1 \text{ by -1} \\
&\quad \text{If } k[j] == 1 \text{ Then} \\
&\quad \quad V_1 = V_h * V_l - P; \\
&\quad \quad V_h = V_h - 2; \\
&\quad \quad \text{Else} \\
&\quad \quad V_h = V_h * V_l - P; \\
&\quad \quad V_l = V_l * V_l - 2; \\
&\quad \text{End If} \\
\text{End For} \\
V_l = V_h * V_l - P; \\
\text{For } j \text{ from 1 to } s \\
&\quad V_l = V_l * V_l - 2; \\
\text{End For} \\
V_n(P, 1) = V_1;
\end{align*}

VI.Doubling With Remainder for LUC2

Now we present our proposed technique; Doubling with Remainder. In order to use and utilize the technique of the doubling-step, we need to overcome the problem of fuzzy and indirect calculations of doubling-step to obtain \( V_n \).

In other words, we need a control sequence for doubling-steps to organize the computations and finally obtain the required value of \( V_n \). For this reason, we then proposed our technique, Doubling with Remainder.

**Algorithm 2** Doubling with Remainder Algorithm

Input : \( k = 2^s \sum_{i=0}^{n-1} k_i 2^{i-s} \) where \( (k_s) = 1 \).

Output : \( (V_n) \).

Compute \( V_2, V_3, V_4 \)

If \( k[s_0] == 1 \) Then

\( V_{2n} = V_3; V_{2n+1} = V_4; \)

Else

\( V_{2n} = V_2; V_{2n+1} = V_3; \)

End If

For \( j = 0 \) to \( s_n - 2 \)

If \( k[j] == 1 \) Then

\( V_l = V_{2n+1} * P - V_{2n}; \)

\( V_{2n} = V_{2n+1} * V_{2n+1} - 2; \)

\( V_{2n+1} = V_{2n+1} * V_l - P; \)

Else

\( V_{2n+1} = V_{2n} * V_{2n+1} - P; \)

\( V_{2n} = V_{2n} * V_{2n} - 2; \)

End If

End For

If \( k[s_n] == 1 \) Then

\( V_n = V_{2n+1}; \)

Else

\( V_n = V_{2n}; \)

End If

\( V_n(P, 1) = V_n; \)

VII. Results

Table I shows the number of iterations for doubling steps for a starting value of 2, 3 and 5. It is also shown the number of iteration of doubling steps to be implemented before the sequential computations return.

**TABLE I**

<table>
<thead>
<tr>
<th>Number of Iterations for Doubling Step Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>400</td>
</tr>
</tbody>
</table>

Table II then shows the number of iterations for sequential, doubling steps and doubling with remainder. By generating the remainder sequences of \( n \) for \( V_n \), we can select the iteration started by \( V_2 \) or \( V_3 \) then followed by another \( n \), depending on generated remainder sequences. Compared to doubling with remainder, the doubling steps start by 2, 3 or 5.
TABLE II
NUMBER OF ITERATIONS FOR EACH ALGORITHM

<table>
<thead>
<tr>
<th>Keys size (bits)</th>
<th>Iterations for Sequential Start Doubling Value = 2</th>
<th>Iterations for Doubling Step Start Doubling Value = 2</th>
<th>Iterations for Doubling with Remainder Start Doubling Value = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>$325 \times 10^{99}$</td>
<td>$328 \times 10^{99}$</td>
<td>$326$</td>
</tr>
<tr>
<td>260</td>
<td>$721 \times 10^{259}$</td>
<td>$857 \times 10^{254}$</td>
<td>$856$</td>
</tr>
<tr>
<td>470</td>
<td>$922 \times 10^{469}$</td>
<td>$1556 \times 10^{465}$</td>
<td>$1555$</td>
</tr>
</tbody>
</table>

VIII. CONCLUSION

The number of iterations with Doubling with Remainder technique was rapidly decreased compare to the standard implementation and doubling-step technique of LUC₂ cryptosystem.

The comparison as shown in table II for each algorithms tested, shows that it has improved by reducing the number of multiplication. It makes the LUC₂ cryptosystem computationally more efficient for security implementations over the electronic communication networks. Likewise, the reduction of iterations by using the technique of doubling with remainder algorithm enable us to achieve a good reduction of computation time.

Doubling with remainder technique also leads to a high reduction in the multiplications required for both the encryption and decryption operations without sacrificing the key size of LUC₂ cryptosystem security. Thus, increasing the key size to gain greater security is feasible by using doubling with remainder technique.

ACKNOWLEDGMENTS

The authors would like to thank the suggestions of many people.

REFERENCES