Bull. Aust. Math. Soc. **90** (2014), 160–171
doi:10.1017/S000497271400001X

AN EFFICIENT METHOD FOR IMPROVING THE COMPUTATIONAL PERFORMANCE OF THE CUBIC LUCAS CRYPTOSYSTEM

REZA NAGHIZADEH MAJID✉, ELANKOVAN SUNDARARAJAN and ZULKARNAIN MD ALI

(Received 11 November 2013; accepted 19 December 2013; first published online 10 April 2014)

Abstract

The cubic version of the Lucas cryptosystem is set up based on the cubic recurrence relation of the Lucas function by Said and Loxton ['A cubic analogue of the RSA cryptosystem', *Bull. Aust. Math. Soc.* **68** (2003), 21–38]. To implement this type of cryptosystem in a limited environment, it is necessary to accelerate encryption and decryption procedures. Therefore, this paper concentrates on improving the computation time of encryption and decryption in cubic Lucas cryptosystems. The new algorithm is designed based on new properties of the cubic Lucas function and mathematical techniques. To illustrate the efficiency of our algorithm, an analysis was carried out with different size parameters and the performance of the proposed and previously existing algorithms was evaluated with experimental data and mathematical analysis.

2010 *Mathematics subject classification*: primary 94A60; secondary 11B37.

Keywords and phrases: cryptography, public key cryptosystem, fast cryptography, Lucas cryptosystem.

1. Introduction

As information technology becomes more developed and matures, security in confidential data transmission has become increasingly crucial. Many different strategies are employed to protect data from intruders, such as physical protection and mathematical solutions for secrecy. Cryptography is one of the components of security methods that protects information from unauthorised interception and tampering during transfer. It is the study of techniques that use mathematical formulas and computational algorithms to encrypt intelligible messages (plaintext) into unknown codes (ciphertext), and convert them back to the original message by decrypting. There are many cryptographic algorithms for protecting messages; these are classified into symmetric key and asymmetric key (public key) algorithms, distinguished by the number and type of keys [8]. These algorithms cannot be proved to be completely

The authors would like to express their gratitude to the Ministry of Higher Education of Malaysia for supporting this work (research grant number FRGS/1/2012/SG05/UKM/02/1).

secure, but their use decreases the probability of messages being intercepted. An analysis of the advantages and disadvantages of both classes shows that asymmetric key algorithms, because of the use of pairs of keys for encryption and decryption, are more secure than symmetric key algorithms. However, huge computations are required for asymmetric key algorithms, leading to slow encryption and decryption speeds. Therefore, the computational performance of public key cryptosystems is always a challenge in cryptography research. Thus, to remedy this drawback, some fast public key cryptosystems have been proposed, including the NTRU cryptosystem which is based on algebraic structures of certain polynomial rings [6] and the Braid groups based cryptosystem [5]. Other research has investigated improving some strong existing public key cryptosystems, such as the RSA and the elliptic curve, using mathematical techniques and computational algorithms [13, 17].

Lucas functions, proposed by Smith and Lennon [14], are one of several trapdoor functions used in cryptography. These authors point out weaknesses of the most widely used cryptosystem, RSA [11], and propose the Lucas (LUC) cryptosystem based on the special form of second-order linear recurrence relations. The security of this type of public key cryptosystem is analogous to RSA and depends on the difficulty of finding a private key [1]. Smith and Lennon claim that the computational performance of the LUC cryptosystem is more efficient than the RSA, and recently Brandner showed that the LUC cryptosystem is 1.4 times more efficient than the RSA without using any improvement techniques [2]. In 2003, Said and Loxton developed a new cryptosystem by extending the quadratic version of Lucas sequence to cubic and called it the LUC_3 cryptosystem [12]. The strength of both the LUC and LUC_3 cryptosystems is dependent on using large prime numbers. However, computations with large prime numbers require huge computation times and prohibit the implementation of Lucas based cryptosystems in resource-constrained environments such as smart cards and cell phones. So, mathematical and computational approaches are needed to speed up computational efforts in the LUC_3 cryptosystem.

Previous related work on a fast LUC cryptosystem includes the Yen and Laih method based on a binary method [18]; Wang *et al.*'s addition chain method in [16]; doubling with the remainder method, which was proposed by Othman [9]; and the Majid *et al.* method to improve computations based on new properties of Lucas functions [7]. For the fast LUC_3 cryptosystem, Said and Loxton also suggested the binary method and used Toom–Cook arithmetic to reduce the computational effort of encoding and decoding a message [12].

This paper concentrates on the LUC_3 cryptosystem and its ability to increase the efficiency of computations. The new properties of the cubic version of the Lucas functions and previous experience of the fast LUC cryptosystem are used in Said and Loxton's LUC_3 cryptosystem to accelerate the encoding and decoding of messages. Section 2 gives a brief description of the LUC and LUC_3 cryptosystems. The proposed algorithm for a fast LUC_3 cryptosystem is described in Section 3 and the analysis of our algorithm and existing algorithms is given in Section 4.

2. Overview of Lucas functions based cryptosystems

Lucas functions are defined as one-way functions used in cryptography, and are seen as special linear recurrences. Suppose that $P_1, P_2, \ldots, P_m$ are integers. Lucas functions are solutions of the linear recurrence

$$T_n = P_1 T_{n-1} + P_2 T_{n-2} + \cdots + P_m T_{n-m}.$$

According to the degree of this recurrence relation, the LUC and LUC$_3$ cryptosystems have been proposed as two different cryptosystems.

2.1. LUC cryptosystem. Assume that P and Q are two rational integers. The standard Lucas sequences are given by two second-order linear recurrences:

$$V_0 = 2,\ V_1 = P,\ V_n = PV_{n-1} - QV_{n-2}, \quad \text{for } n \geq 2;$$
$$U_0 = 0,\ U_1 = 1,\ U_n = PU_{n-1} - QU_{n-2}, \quad \text{for } n \geq 2.$$

If $x^2 - Px + Q = 0$ is the characteristic equation of this recurrence relation, where α and β are the roots of this polynomial equation, then the two functions, V_n and U_n, are derived from solving the equations as follows:

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

Let P and N be two chosen numbers, with N the product of two prime numbers p and q; P is the message. The encryption converts plaintext P and gets ciphertext C by calculating $V_e(P, Q) \pmod N$ or $U_e(P, Q) \pmod N$, where V_e and U_e are the eth terms of the Lucas sequences. For the decryption of ciphertext C, the value of the private key d must be found so that $de \equiv 1 \pmod{S(N)}$, where $S(N) = \mathrm{lcm}(p - (D/p), q - (D/q))$, $D = C^2 - 4$, where (D/p) and (D/q) are Legendre symbols of D with respect to p and q, and lcm is the least common multiple function. After finding the private key d, the decryption process is done, and the plaintext is obtained by evaluating the dth term of the Lucas sequence in $V_d(P, Q) \pmod N$ or $U_d(P, Q) \pmod N$.

2.2. LUC$_3$ cryptosystem. The LUC$_3$ cryptosystem is a public key cryptosystem that uses the third-order linear recurrence of the Lucas function for encryption and decryption. Consider P, Q and R as three integer coefficients, so that

$$X_n = PX_{n-1} - QX_{n-2} + RX_{n-3}. \tag{2.1}$$

Let α, β and γ be the three roots of the cubic $f(x) = x^3 - Px^2 + Qx - R = 0$. The cubic Lucas sequences V_n, U_n and W_n are the three basic solutions of the above recurrence, namely,

$$V_n(P, Q, R) = \alpha^n + \beta^n + \gamma^n,$$
$$U_n(P, Q, R) = \alpha^n + \omega\beta^n + \omega^2\gamma^n,$$
$$W_n(P, Q, R) = \alpha^n + \omega^2\beta^n + \omega\gamma^n,$$

where $\omega = (-1 + \sqrt{-3})/2$ is a cube root of unity. The sequence V_n is primarily used in the LUC$_3$ cryptosystem. Suppose $R = 1$ and (P, Q) constitutes a message and N is the product of two prime numbers p and q. The encryption process uses public key e and computes the ciphertexts C_1 and C_2 by $C_1 = V_e(P, Q, 1) \pmod{N}$ and $C_2 = V_e(Q, P, 1) \pmod{N}$, where V_e is the eth term of (2.1) with initial values $V_0 = 3$, $V_1 = \alpha + \beta + \gamma = P$ and $V_2 = P^2 - 2Q$. So

$$E(P, Q) = (V_e(P, Q, 1), V_e(Q, P, 1)) = (C_1, C_2) \pmod{N}.$$

In the deciphering process, the receiver must calculate $ed \equiv 1 \pmod{\phi(N)}$ for C_1 and C_2 to choose the proper private key, d. In the case of the third-order linear recurrence sequence, the Euler totient function is defined as $\phi(N) = \bar{p}_1\bar{p}_2$ and the value of $\bar{p}_i$ is given by

$$\bar{p}_i = \begin{cases} p_i^2 + p_i + 1 & \text{if } f(x) \text{ is of type } t[3] \pmod{p_i}, \\ p_i^2 - 1 & \text{if } f(x) \text{ is of type } t[2,1] \pmod{p_i}, \\ p_i - 1 & \text{if } f(x) \text{ is of type } t[1] \pmod{p_i}. \end{cases}$$

This notation follows Said and Loxton: $f(x)$ is of type $t[3]$ if it is irreducible, of type $t[2, 1]$ if it factors as an irreducible quadratic times a linear factor, and of type $t[1]$ if it factors into three linear factors.

Finally, the original messages P and Q are recovered by means of

$$D(C_1, C_2) = (V_d(C_1, C_2, 1), V_e(C_2, C_1, 1)) = (P, Q) \pmod{N}.$$

3. The fast LUC$_3$ cryptosystem

As with the LUC cryptosystem, the security of the LUC$_3$ cryptosystem is dependent on the difficulty of finding private key d from public key e and N. Therefore, choosing large integers for e and N increases the strength of the LUC$_3$ cryptosystem. However, it causes an increasing number of modular and multiplication operations that are time-consuming. The efficiency of the LUC$_3$ cryptosystem depends on the ability to compute the nth cubic Lucas function $V_n(P, Q, 1)$ in a reasonable amount of time. In this section, the existing algorithm for the fast LUC$_3$ is reviewed and the new algorithm for improving the cost of computation time is proposed.

3.1. Existing algorithm. Suppose that α, β and γ are the roots of the the cubic polynomial equation $f(x) = x^3 - Px^2 + Qx - R = 0$. The coefficients P, Q and R and the roots α, β and γ are related by $P = \alpha + \beta + \gamma$, $Q = \alpha\beta + \alpha\gamma + \beta\gamma$, $R = \alpha\beta\gamma$ and we assume that $R = 1$. Due to these relations, Said and Loxton mentioned some properties of the Lucas sequence $V_n = \alpha^n + \beta^n + \gamma^n$ as follows:

$$V_n(Q, P, 1) = (\alpha\beta)^n + (\alpha\gamma)^n + (\beta\gamma)^n,$$
$$V_{-n}(P, Q, 1) = \alpha^{-n} + \beta^{-n} + \gamma^{-n} = V_n(Q, P, 1),$$
$$V_{2n}(P, Q, 1) = \alpha^{2n} + \beta^{2n} + \gamma^{2n} = V_n^2(P, Q, 1) - 2V_{-n}(P, Q, 1),$$
$$V_{-2n}(P, Q, 1) = \alpha^{-2n} + \beta^{-2n} + \gamma^{-2n} = V_{-n}^2(P, Q, 1) - 2V_n(P, Q, 1).$$

For efficient computation with LUC$_3$, they also presented a binary method based on doubling rules to calculate the nth terms of Lucas sequences $V_n(P, Q, 1)$ and $V_{-n}(P, Q, 1)$ simultaneously. With their method, the signatures of $n \pmod N$ are defined as follows:

$$V_{-n-1},\ V_{-n},\ V_{-n+1},\ V_{n-1},\ V_n,\ V_{n+1} \pmod N.$$

All the above signatures are computed by applying doubling rules on the signatures of $m \pmod N$ where $m < n$:

$$V_{-2m-2},\ V_{-2m},\ V_{-2m+2},\ V_{2m-2},\ V_{2m},\ V_{2m+2} \pmod N.$$

The gaps between the above signatures are called centred signatures, which can be written by (2.1) as follows:

$$\begin{aligned}
V_{-2m-1} &= V_{-2m+2} - PV_{-2m+1} + QV_{-2m} \pmod N,\\
V_{-2m+1} &= PV_{-2m} - QV_{-2m-1} + V_{-2m-2} \pmod N,\\
V_{2m-1} &= V_{2m+2} - PV_{2m+1} + QV_{2m} \pmod N,\\
V_{2m+1} &= PV_{2m} - QV_{2m-1} + V_{2m-2} \pmod N.
\end{aligned}$$

Solving each pair of the above equations yields centred signatures. So, V_{-2m-1} and V_{2m+1} are found by (3.1) and (3.2). In both equations, the value of $(PQ-1)^{-1}$ is obtained by calculating the extended Euclidean algorithm of $(PQ-1)$ and N. By replacing these equations in the above relations, the equations for V_{-2m+1} and V_{2m-1} are computed:

$$V_{-2m-1} = ((P^2-Q)V_{-2m} + PV_{-2m-2} - V_{-2m+2})(PQ-1)^{-1} \pmod N \tag{3.1}$$

$$V_{2m+1} = ((Q^2-P)V_{2m} + QV_{2m+2} - V_{2m-2})(PQ-1)^{-1} \pmod N. \tag{3.2}$$

Now let $n = (n_0, n_1, \dots, n_{r-1})_2$ be the binary representation of n, where n_0 and n_{r-1} are the most and least significant bits, respectively. This array of bits could be called the Lucas binary chain. If the first k bits of a binary number produce the number m, then the first $k+1$ bits produce $2m$ or $2m+1$, depending on whether n_i is equal to 0 or 1. Therefore, Said and Loxton's fast algorithm is developed by Algorithm 1.

```
ALGORITHM 1: Said and Loxton's method
Input: $P, Q, p, q, n$
Output: $V_n(P, Q, 1), V_{-n}(P, Q, 1)$
Initialise $V_n = V_1$ and $V_{-n} = V_{-1}$;
For each $i$ from 1 to $r-1$ do
  If $n_i$ equal to 0 Then
    Compute signatures $V_{-2n-2}, V_{-2n}, V_{-2n+2}, V_{2n-2}, V_{2n}, V_{2n+2} \pmod N$;
    Compute centred signatures $V_{-2n-1}, V_{-2n+1}, V_{2n-1}, V_{2n+1} \pmod N$;
    Set $V_{-n+1} = V_{-2n+1}, V_{-n} = V_{-2n}, V_{-n-1} = V_{-2n-1}$;
    Set $V_{n-1} = V_{2n-1}, V_n = V_{2n}, V_{n+1} = V_{2n+1}$;
  Else
```

```
        Compute signatures $V_{-2n-2}, V_{-2n}, V_{-2n+2}, V_{2n-2}, V_{2n}, V_{2n+2} \pmod{N}$;
        Compute centred signatures $V_{-2n-1}, V_{-2n+1}, V_{2n-1}, V_{2n+1} \pmod{N}$;
        Set $V_{-n+1} = V_{-2n}, V_{-n} = V_{-2n-1}, V_{-n-1} = V_{-2n-2}$;
        Set $V_{n-1} = V_{2n}, V_n = V_{2n+1}, V_{n+1} = V_{2n+2}$;
    EndIf
EndFor
Return $(V_n(P, Q, 1), V_{-n}(P, Q, 1))$;
```

This algorithm requires $\lceil \log n \rceil$ steps. In each step, there are many modular multiplications running in $O(n^2)$. Thus, to economise oncomputation time, they suggest using Toom–Cook arithmetic, which can reduce the running time to $O(n^{1.4})$ in each step. The pseudo-code of the Toom–Cook three-way algorithm for $M = A \times B$ is shown in Algorithm 2 [3].

Algorithm 2: Toom–Cook three-way multiplication

```
Input: $0 < A$ and $B < \beta^n$ $A = a_0 + a_1x + a_2x^2$, $B = b_0 + b_1x + b_2x^2$; with $x = \beta^k$;
Output: $M = A \times B = c_0 + c_1\beta^k + c_2\beta^{2k} + c_3\beta^{3k} + c_4\beta^{4k}$; with $k = n/3$;
Compute $v_0 = \text{ToomCook3}(a_0, b_0)$;
Compute $v_1 = \text{ToomCook3}(a_{02} + a_1, b_{02} + b_1)$; where $a_{02} = a_0 + a_2$ and $b_{02} = b_0 + b_2$;
Compute $v_{-1} = \text{ToomCook3}(a_{02} - a_1, b_{02} - b_1)$;
Compute $v_2 = \text{ToomCook3}(a_0 + 2a_1 + 4a_2, b_0 + 2b_1 + 4b_2)$;
Compute $v_\infty = \text{ToomCook3}(a_2, b_2)$;
Compute $t_1 = (3v_0 + 2v_{-1} + v_2)/6 - 2v_\infty$ and $t_2 = (v_1 + v_{-1})/2$;
Set $c_0 = v_0, c_1 = v_1 - t_1, c_2 = t_2 - v_0 - v_\infty, c_3 = t_1 - t_2, c_4 = v_\infty$;
Substitute in $M = A \times B = c_0 + c_1\beta^k + c_2\beta^{2k} + c_3\beta^{3k} + c_4\beta^{4k}$;
Return $(M)$;
```

3.2. Proposed algorithm. The following algorithm is proposed to decrease the number of modular multiplications with respect to the existing algorithm in order to improve computation time. As mentioned, Said and Loxton's method needs to calculate six signatures by doubling rules and four centred signatures, which are found by these six signatures. Therefore, it not only increases the number of modular multiplications, but also avoids calculating all signatures of each step at the same time. The proposed algorithm is developed by new properties of the Lucas functions and uses addition–subtraction chains. In this sequential algorithm, the number of signatures required to calculate V_n and V_{-n} is reduced, and the centred signatures are found by the signatures of the previous step. Proposition 1 describes properties that can be used in our algorithm to calculate Lucas sequences V_n and V_{-n}.

Proposition 3.1. *Let n and m be two integers with $n > m$. The $(n + m)$th term of the cubic Lucas function V_{n+m} is obtained by (3.3):*

$$V_{n+m} = V_nV_m - V_{n-m}V_{-m} + V_{n-2m} \pmod{N}. \tag{3.3}$$

Proof. According to the relationships of the LUC$_3$ cryptosystem, (3.3) is proved as follows:

$$\begin{aligned}
V_{n+m} &= \alpha^{n+m} + \beta^{n+m} + \gamma^{n+m} \\
&= (\alpha^n + \beta^n + \gamma^n)(\alpha^m + \beta^m + \gamma^m) - \alpha^n\beta^m - \alpha^n\gamma^m - \alpha^m\beta^n - \beta^n\gamma^m - \alpha^m\gamma^n - \beta^m\gamma^n \\
&= V_nV_m - \alpha^m\beta^m(\alpha^{n-m} + \beta^{n-m}) - \alpha^m\gamma^m(\alpha^{n-m} + \gamma^{n-m}) - \beta^m\gamma^m(\beta^{n-m} + \gamma^{n-m}) \\
&= V_nV_m - \alpha^m\beta^m(V_{n-m} - \gamma^{n-m}) - \alpha^m\gamma^m(V_{n-m} - \beta^{n-m}) - \beta^m\gamma^m(V_{n-m} - \alpha^{n-m}) \\
&= V_nV_m - V_{n-m}(\alpha^m\beta^m + \alpha^m\gamma^m + \beta^m\gamma^m) + \alpha^m\beta^m\gamma^m\gamma^{n-2m} + \alpha^m\beta^m\gamma^m\beta^{n-2m} \\
&\qquad + \alpha^m\beta^m\gamma^m\alpha^{n-2m} \\
&= V_nV_m - V_{n-m}V_{-m} + V_{n-2m}.
\end{aligned}$$

This completes the proof. □

By substituting n and m with various values in (3.3), the new equations obtained are as follows. Replacing m and n by $-n$ and $-n+1$ respectively yields

$$V_{-2n+1} = V_{-n+1}V_{-n} - PV_n + V_{n+1} \pmod{N}. \tag{3.4}$$

Replacing m by $n-1$ yields

$$V_{2n-1} = V_nV_{n-1} - PV_{-n+1} + V_{-n+2} \pmod{N}, \tag{3.5}$$

and replacing m by n and n by $n+1$ yields

$$V_{2n+1} = V_{n+1}V_n - PV_{-n} + V_{-n+1} \pmod{N}. \tag{3.6}$$

The method of the proposed algorithm is such that first the Lucas addition–subtraction chain for a positive integer n is computed and then the terms of the Lucas sequences for all elements of this chain are evaluated to obtain V_n and V_{-n}.

Definition 3.2. The Lucas addition–subtraction chain for a positive integer n is a set of integers $L(n) = \{a_1 = 1, a_2, a_3, \ldots, a_r = n\}$, with chain length r, such that each element a_k of the chain can be written as a sum, difference or successor of preceding elements of the set. Thus, for all $k \in \{1, \ldots, r\}$, there exist $i, j \in \mathbb{N}$, $0 \le j, i < k$, such that

$$a_k = |a_i - a_j| \quad \text{or} \quad a_k = a_i + a_j \quad \text{or} \quad a_k = a_i + 1.$$

From the many methods of generating addition–subtraction chains, we choose the binary strategy for our proposed algorithm [15]. Let $L(n)$ denote the Lucas addition–subtraction chain. The method of calculating $L(n)$ is as follows:

$$L(n) = \begin{cases} n-1 & \text{if } n \text{ is odd,} \\ n/2 & \text{if } n \text{ is even.} \end{cases}$$

For example the Lucas addition–subtraction chain for V_{198} with $n = 198$ is $L(n) = \{1, 2, 3, 6, 12, 24, 48, 49, 98, 99, 198\}$. To compute V_{198}, the values of $V_1, \ldots, V_{99}, V_{198}$ need to be calculated at each step.

After generating $L(n)$, define its length x as the number of elements of the chain and define the array $k[x]$ of length x and with elements 0 or 1 according as the corresponding elements of $L(n)$ are even or odd. Algorithm 3 describes the proposed fast LUC$_3$ algorithm executed with an array of size x.

ALGORITHM 3: Proposed algorithm

```
Input: $n, P, Q, p, q, x, k[x]$
Output: $V_n(P, Q, 1)$ and $V_{-n}(P, Q, 1)$
Initialise $V_n = V_1$ and $V_{-n} = V_{-1}$;
For each $i$ from 1 to $x - 1$
  If $k[i]$ equal to 0 Then
    Compute $V_{-2n+2}, V_{-2n+1}, V_{-2n}, V_{2n-1}, V_{2n}, V_{2n+1} \pmod{N}$;
    Set $V_{-n+1} = V_{-2n+1}, V_{-n} = V_{-2n}, V_{-n+2} = V_{-2n+2}$;
    Set $V_{n-1} = V_{2n-1}, V_n = V_{2n}, V_{n+1} = V_{2n+1}$,
  Else
    Compute $V_{n+2}, V_{-n-1} \pmod{N}$;
    Set $V_{-n} = V_{-n-1}, V_{-n+1} = V_{-2n}, V_{-n+2} = V_{-2n+1}$;
    Set $V_{n-1} = V_{2n}, V_n = V_{2n+1}, V_{n+1} = V_{n+2}$;
  EndIf
EndFor
Return $(V_n(P, Q, 1), V_{-n}(P, Q, 1))$;
```

In this algorithm, there are three signatures found by doubling rules, three centred signatures V_{-2n+1}, V_{2n-1} and V_{2n+1} which are computed by (3.4)–(3.6), and two other signatures V_{n+2} and V_{-n-1} are acquired from the defining equation of the Lucas sequence as follows:

$$V_{n+2} = PV_{n+1} - QV_n + V_{n-1} \pmod{N},$$
$$V_{-n-1} = V_{-n+2} - PV_{-n+1} + QV_{-n} \pmod{N}.$$

Another technique found effective at speeding up computations is decreasing the computation time of modular multiplication. The calculation of modular multiplication has a running time of $O(n^2)$. To reduce this cost, the fast Fourier transform multiplication (FFTM) method is used, combined with Barrette modular reduction to optimise the multiplication and modular operations. Given $M = A \times B$ and $R = M \pmod{N}$, the calculation of both arithmetic operations is described in Algorithms 4 and 5. More details about FFT, inverse FFT functions and Barrett modular reduction can be found in [4, 10].

ALGORITHM 4: Fast Fourier transform multiplication

```
Input: $A(x) = \sum_{i=0}^{(s/2)-1} a_i x^i$, $B(x) = \sum_{i=0}^{(s/2)-1} b_i x^i$;
Output: $M = A \times B$;
Compute $a = FFT(A(x))$ and $b = FFT(B(x))$;
Compute pointwise multiplication of $a$ and $b$; $m_i = a_i \times b_i$;
Compute inverse FFT of $m_i$ to obtain a vector of coefficients; $IFFT(m_i)$;
```

```
Convert the coefficients to integers according to their radix, $M$.
Return ($M$);
```

```
ALGORITHM 5: Barrett modular reduction method
Input: $M = (m_1, m_2, \ldots, m_n)_s$ and $N = (n_1, n_2, \ldots, n_n)_s$;
Output: $R = M \mod N$;
Compute $q = ((M \text{ div } b^{k+1})\mu) \text{ div } b^{k+1}$; where $k = \log_s N$ and $\mu = b^{2k} \text{ div } N$;
Compute $R = M \mod b^{k+1} - (q \times N) \mod b^{k+1}$;
If ($R < 0$) Then
  $R = R + b^{k+1}$;
EndIf
While ($R \geq N$) do
  $R = R - N$;
EndWhile
Return ($R$);
```

4. Results and discussion

This section discusses the computational performance of the LUC$_3$ cryptosystem using our proposed algorithm. After implementing both binary and addition–subtraction methods in the C# programming language, the methods were tested with different large public key sizes and different N values. Finally, the computation times required in both algorithms were compared using mathematical analysis.

4.1. Implementation. In Lucas-based cryptosystems, reducing the number and computation time of modular multiplication operations can increase efficiency. To evaluate the performance of both algorithms, the algorithms were tested with two cases. In the first case, the value of N is considered constant, and the size of the public key was increased; then the Lucas number was calculated. In the second case, the value of the Lucas number is computed with a public key with a constant value, and the value of N is increased. The computation times in both cases are shown in Tables 1 and 2. In both tables, the encryption and decryption times using the addition–subtraction chain method are shown by T_{eA} and T_{dA} and for the binary method, they are represented as T_{eB} and T_{dB}.

Regarding both proposed and existing methods, encryption times T_{eA} and T_{eB} are obtained by computing $E(P, Q) = (V_e(P, Q, 1), V_e(Q, P, 1)) = (C_1, C_2) \pmod N$, and decryption times are found by $D(C_1, C_2) = (V_d(C_1, C_2, 1), V_e(C_2, C_1, 1)) = (P, Q) \pmod N$, where P and Q are messages and C_1 and C_2 are ciphertexts. Comparing the computation times of both algorithms in Tables 1 and 2 shows a high efficiency of the proposed algorithm. To make a comparison, the performance of both algorithms was tested using numbers that were not overly large, because the binary method requires huge computation time and needs a powerful system with high memory and CPU. As displayed in Table 2, increasing the value of N results in increased encryption and decryption computation times; in some cases, our system could not complete the computations because of lack of memory. Thus, we continued the evaluation of the proposed algorithm for large sizes of parameters, and the results are shown in Table 3.

TABLE 1. Encryption and decryption times for different public key sizes, e.

Public key e (digits)	Message (digits)	N (digits)	T_{eA} (seconds)	T_{dA} (seconds)	T_{eB} (seconds)	T_{dB} (seconds)
10	50	50	0.442	3.309	9.151	64.598
20	50	50	0.983	3.471	18.646	66.947
30	50	50	1.670	3.814	29.604	70.637
40	50	50	2.441	4.174	41.672	76.589

TABLE 2. Encryption and decryption times for different prime numbers; $N = p \times q$.

Public key e (digits)	Message (digits)	N (digits)	T_{eA} (seconds)	T_{dA} (seconds)	T_{eB} (seconds)	T_{dB} (seconds)
50	50	60	3.637	6.215	64.322	138.675
50	50	70	4.136	7.779	83.442	212.997
50	50	80	4.297	8.613	99.932	∞
50	50	90	5.009	11.462	122.824	∞

TABLE 3. Encryption and decryption times for different sizes of e and N, by proposed algorithm.

Public key e (digits)	Message (digits)	N (digits)	T_{eA} (seconds)	T_{dA} (seconds)
100	200	150	14.634	26.344
200	200	250	50.207	81.042
300	200	350	106.893	141.88
400	200	450	195.218	207.839

4.2. Analyses and discussion. Analysing both algorithms for fast LUC$_3$ cryptosystems shows that both algorithms generate an array of bits by Lucas numbers, then scan each bit sequentially to compute V_n. To scan each bit, the binary method needs six modular multiplications for signatures, found using doubling rules and 14 modular multiplications for computing centred signatures. The addition–subtraction method depends on the value of bit 0 or 1, uses nine or four modular multiplications for scanning each bit. The time needed to create an array of Lucas numbers for V_n in the binary method is $\lceil \log_2 n \rceil$. In the addition–subtraction method, the worst case is when the elements of an array are 1 and 0 one by one, which is found in $2 \times \lfloor \log_2 n \rfloor$ steps. The best case in this method is $\lfloor \log_2 n \rfloor$, and this occurs when all the elements are 0. Therefore, if T_A and T_B are the times required to calculate V_n in the addition–subtraction and binary methods, multiplying the length of an array by the number of modular multiplications for each method produces the ratio of the expected time to compute V_n by the proposed algorithm to that by the binary method. The worst and best ratios, r_{worst} and r_{best}, are respectively

$$r_{\text{worst}} = \lim_{n\to\infty} \frac{13 \times \lfloor \log_2 n \rfloor}{20 \times \lceil \log_2 n \rceil} \cong 0.65,$$
$$r_{\text{best}} = \lim_{n\to\infty} \frac{9 \times \lfloor \log_2 n \rfloor}{20 \times \lceil \log_2 n \rceil} \cong 0.45.$$

Without using a method to reduce the computation time of modular multiplication, our method is an average of 55% better than the binary method. This ratio increases remarkably using FFTM with Barrett modular reduction. Using FFTM with Barrett modular reduction in each signature computation will reduce the modular multiplication time to $O(n \log n)$. Therefore, the proposed algorithm can be used to improve the computational performance of the LUC_3 cryptosystem.

5. Conclusion

A new fast algorithm is proposed for improving the computation performance of LUC_3 cryptosystems. Using this method decreases the computation cost of modular multiplication with respect to the existing algorithm, and results in a reduction in the computation time needed for encryption and decryption in resource-constrained environments. Furthermore, in contrast to the binary method, all resulting signatures in each step are completely dependent on the signatures of previous steps. This causes the simultaneous calculation of signatures during the process of scanning the bits, and opens a path to the development of a parallel algorithm to make the performance even more efficient.

References

[1] D. Bleichenbacher, W. Bosma and A. K. Lenstra, ‘Some remarks on Lucas-based cryptosystem’, in: *Advances in Cryptology, CRYPTO 95 Proceeding of the 15th Annual International Cryptology Conference* (ed. Don Coppersmith) (Springer, Berlin, 1995), 386–396.

[2] G. Brandner, ‘RSA, Dickson, LUC and Williams: a study on four polynomial-type public-key cryptosystems’, *Appl. Algebra Engrg. Comm. Comput.* **24** (2013), 17–36.

[3] R. Brent and P. Zimmermann, *Modern Computer Arithmetic* (Cambridge University Press, New York, 2010).

[4] J. P. David, K. Kalach and N. Tittley, ‘Hardware complexity of modular multiplication and exponentiation’, *IEEE Trans. Comput.* **56** (2007), 1308–1319.

[5] P. Dehornoy, ‘Braid-based cryptography’, in: *Group Theory, Statistics, and Cryptography*, Contemporary Mathematics, 360 (American Mathematical Society, Providence, RI, 2004), 5–22.

[6] J. Hoffstein, J. Pipher and J. Silverman, ‘NTRU: A ring-based public key cryptosystem’, in: *ANTS*, Lecture Notes in Computer Science, 1423 (ed. J. P. Buhler) (Springer, Berlin, 1998), 267–288.

[7] R. N. Majid, E. Sundararajan and Z. M. Ali, ‘A new fast method of evaluating Lucas sequence for improving encryption and decryption times in Lucas cryptosystem’, *Int. J. Inf. Process. Manage.* **4** (2013), 11–18.

[8] A. J. Menezes, P. C. Van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, Boca Raton, FL, 1997).

[9] M. Othman, E. M. Abulhirat, Z. M. Ali, R. Johari and M. R. M. Said, ‘A new computation algorithm for a cryptosystem based on Lucas functions’, *J. Comput. Sci.* **4** (2008), 1056–1060.

[10] D. S. Phatak and T. Goff, ‘Fast modular reduction for large wordlengths via one linear and one cyclic convolution’, in: *17th IEEE Symposium on Computer Arithmetic (ARITH’05), 2005* (2005), 179–186.
[11] R. L. Rivest, A. Shamir and L. Adelman, ‘A method for obtaining digital signatures and public-key cryptosystems’, *Comm. ACM* **21** (1978), 120–126.
[12] M. R. M. Said and J. Loxton, ‘A cubic analogue of the RSA cryptosystem’, *Bull. Aust. Math. Soc.* **68** (2003), 21–38.
[13] N. A. Saqib, F. Rodriguez-Henriquez and A. Diaz-Perez, ‘A parallel architecture for fast computation of elliptic curve scalar multiplication over GF(2m)’, *18th International Parallel and Distributed Processing Symposium* **2** (2004), 493–497.
[14] P. Smith and M. Lennon, *LUC: A new public key system*, Computer Security, Proc. IFIP TC11 9th Int. Conf. Information Security, IFIP/Sec’93, Toronto, Canada, 1993 (North-Holland, Amsterdam, 1993), 103–117.
[15] A. Tall and A. Y. Sanghare, ‘Eficient computation of addition subtraction chains using generalized continued fractions’, *Int. J. Appl. Math. Res.* **2** (2013), 76–83.
[16] C. T. Wang and C. C. Chang, ‘A method for computing Lucas sequences’, *Int. J. Comput. Math. Appl.* **38** (1999), 187–196.
[17] B. Wang, Y. Wei and Y. Hu, ‘Fast public-key encryption scheme based on Chinese remainder theorem’, *Front. Electr. Electron. Eng.* **4** (2009), 181–185.
[18] S. Yen and C. Laih, ‘Fast algorithm for LUC digital signature computation’, *IEEE Proc. Comput. Digit. Tech.* **142** (1995), 165–169.

REZA NAGHIZADEH MAJID, Computer Science Program,
Faculty of Information Science and Technology,
University Kebangsaan, Malaysia
e-mail: r.naghizade@gmail.com

ELANKOVAN SUNDARARAJAN, Information Technology Program,
Faculty of Information Science and Technology,
University Kebangsaan, Malaysia
e-mail: elankovan.sundararajan@gmail.com

ZULKARNAIN MD ALI, Computer Science Program,
Faculty of Information Science and Technology,
University Kebangsaan, Malaysia
e-mail: zulkarnainmdali@gmail.com