Malaysian Journal of Mathematical Sciences 7(S): 131-143 (2013)
Special Issue: The 3rd International Conference on Cryptology & Computer Security 2012 (CRYPTOLOGY2012)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

Calculation Enhancement of Chebyshev Polynomial over Z_p

***Mohammed Benasser Algehawi, Azman Samsudin and Shahram Jahani**

School of Computer Sciences, Universiti Sains Malaysia, Penang, 11800, Malaysia

E-mail: benasser@cs.usm.my, azman@cs.usm.my and jahani2001@yahoo.com

*Corresponding author

ABSTRACT

It has been recommended that the safe size of the key space for any cryptosystem based on Chebyshev polynomial extended over the finite field $\mathbb{Z}_p$ must be chosen such that $p \geq 2^{256}$. For such size of p, the normal Chebyshev polynomial calculation speed will be slow and impractical. Thus, there is a need to improve the Chebyshev polynomial calculation before the polynomial can be used in mainstream cryptosystems. In this paper, two types of Chebyshev polynomial calculation models are being considered, the Matrix Algorithm and the Characteristic Polynomial Algorithm. This paper introduces new technique to improve both of these calculation models. Preliminary results show indications that the proposed technique is a reliable alternative for implementing Chebyshev polynomial calculation.

Keywords: Public-key cryptography, Chebyshev polynomial, and Chaos cryptography.

1. INTRODUCTION

Many cryptosystems have been proposed based on Chebyshev polynomial (Algehawi and Samsudin (2010), Wang and Zhao (2010), Xiao *et al.* (2007), Yoon and Yoo (2008)). The common issue on these cryptosystems is the security concern against attacks such as brute force attack and period distribution attack. Consider the equation of the extended Chebyshev polynomial over the finite field $\mathbb{Z}_p$ as follows:

$$T_n(x) = (a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0) \, (mod \, p) \tag{1}$$

To avoid brute force attack and attack based on period distribution on cryptosystems based on Chebyshev polynomial extended over the finite field $\mathbb{Z}_p$, the value of p and $n$must be large enough such that large period can be acquired. As claimed by many researchers (Liao et al. 2010; Li et al. 2011; Fee & Monagan 2004), p is a strong prime when both $p+1$ and $p-1$ have large factors, that is, larger that 2^{256}.For such size of p, the Chebyshev polynomial calculation speed is slow and impractical. There are numerous solutions that have been proposed in order to enhance the Chebyshev polynomial (over$\mathbb{Z}_p$) computational speed.

Two models of Chebyshev polynomial calculation are being considered in this paper. They are the Matrix Algorithm (MA) and the Characteristic Polynomial Algorithm (CPA), which were introduced by Fee and Monagan (2004). These two algorithms had been improved by Li et al. in 2011 and they named the improved version of the two algorithms as the Modified Matrix Algorithm (MMA) and the Modified Characteristic Polynomial Algorithm (MCPA), respectively.

2. EXISTING ALGORITHMS

In this section, the existing algorithms that have been used in speeding up the Chebyshev polynomial over $\mathbb{Z}_p$calculation are revised. The algorithms are MA, MMA, CPA, and MCPA.

2.1 Matrix Algorithm (MA)

The first matrix algorithm to calculate the Chebyshev polynomial, $T_n(x)$, was proposed by Fee and Monagan (2004). By considering Equation 1, this algorithm can be explained as follows(Li *et al.* (2011)):

Input: n, A, I
$If n = 0$ *then*
  $Return\ 1$
Else
  $If n > 0$ *then*
    $C \leftarrow I,\ S \leftarrow A$ // C represents the immediate result
    $For i = 0\ upto\ (n-1)\ by\ 1$
      $If b_i = 1\ then$
        $C \leftarrow C \times S$ //S holds the result of squaring the matrix A
        $S \leftarrow S \times S$

Output: $C_{11} + xC_{12}$ // C_{11},C_{12} are elements of C

In the matrix algorithm, MA, the user uses the square and multiply technique to calculate $C = A^n$. The algorithm starts after receiving the exponent value$n$, matrix A, and the matrix I. If the value of n is 0 then the program will return 1 as a result, else C will be initialized with the identity matrix I and S will be initialized with the matrix A.At each step i of the loop, the exponent$n$ is calculated as $n = \sum_{j=0}^{r-1} b_j 2^j$ and the intermediate matrix is calculated as$S = A^{2(i+1)}$. The end result is therefore a matrix equivalent to A^{2n}.

Assume m representing the number of the 1's in the binary representation of the exponent n of matrix A. Then, the squaring $(S \times S)$ is performed in each loop step while the multiplication $(C \times S)$ is performed only $m + 1$ times with the first multiplication $A \times I$ where its running time is negligible. The binary digits of the exponent n are being checked one by one in right to left order.To calculate the running cost of MA, the following matrix operations should be considered:

- $C \times S = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} = \begin{pmatrix} C_{11}S_{11} + C_{12}S_{21} & C_{11}S_{12} + C_{12}S_{22} \\ C_{21}S_{11} + C_{22}S_{21} & C_{21}S_{12} + C_{22}S_{22} \end{pmatrix}$

  has the cost of$8t_m + 4t_a$ (where $t_m$indicates the calculation time for performing matrix multiplication and t_a indicates the calculation time for performing matrix addition).

- $C \times C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} {C_{11}}^2 + C_{12}C_{21} & C_{12}(C_{11} + C_{22}) \\ C_{21}(C_{11} + C_{22}) & C_{21}C_{12} + {C_{22}}^2 \end{pmatrix}$

  has the cost of$3t_m + 2t_s + 3t_a$ (where $t_s$indicates the calculation time for performing matrix squaring).

- $A \times C = C \times A = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix} = \begin{pmatrix} -C_{12} & C_{11} + 2xC_{12} \\ -C_{22} & C_{21} + 2xC_{22} \end{pmatrix}$

  has the cost of $2t_m + 2t_a$.

Subsequently, the computation time of the MA scheme can be illustrated as follows:

$$T_{MA} = (r-1)(3t_m + 2t_s + 3t_a) + m(8t_m + 4t_a) + 3(t_m t_a) \quad (2)$$

where r is the number of bits in the exponent n and m is the number of 1's in the exponent n.

2.2 Characteristic Polynomial Algorithm

The Characteristic Polynomial Algorithm, CPA, that was introduced to calculate the Chebyshev polynomial, $T_n(x)$, was presented in 2004 by Fee and Monagan. Based on Cayley-Hamilton theory, the matrix satisfies its own characteristic polynomial, i.e. if the characteristic polynomial of a square matrix A is $f(\lambda) = det\ (\lambda I - A)$ then $f(A) = 0$ where *det* is the determinant operation and I is the identity matrix. More precisely, since the entries of the matrix are (constant) polynomials in λ, the determinant is also a polynomial in λ. The Cayley–Hamilton theorem states that substituting the matrix A for λ in this polynomial will resulted in $f(A) = 0$. Thus, instead of calculating A^n, we just need to calculate λ^n modulo the characteristic polynomial by using the square and multiply technique. The characteristic polynomial is a quadratic polynomial and can be expressed as $f(\lambda) = a_1\lambda + a_0$, where the *n*-th power of the matrix Ahas the form of$a_1A + a_0$and therefore$T_n(x)$ can be expressed as follows (Li et al., 2011):

$$\begin{aligned} T_n(x)\ mod\ p &= A^n \begin{pmatrix} T_0(x) \\ T_1(x) \end{pmatrix} mod\ p \\ &= (a_1A + a_0) \begin{pmatrix} 1 \\ x \end{pmatrix} mod\ p \\ &= (a_1x + a_0)\ mod\ p \end{aligned} \quad (3)$$

By using Equation 3, $T_n(x)$ can be coded as follows:

Input: n, A, I
$Ifn = 0 \ \ then$
  $Return\ 1$
$Else$
  $Ifn > 0$
    $a_1\lambda + a_0 \leftarrow I, \quad s_1\lambda + s_0 \leftarrow A$
    $Fori = 0\ upto\ (n-1)\ by\ 1$
      $Ifb_i = 1\ then$
        $a_1\lambda + a_0 \leftarrow (a_1\lambda + a_0) \times (s_1\lambda + s_0)$
        $s_1\lambda + s_0 \leftarrow (s_1\lambda + s_0) \times (s_1\lambda + s_0)$//$(s_1\lambda + s_0) \times (s_1\lambda + s_0)$ uses // squaring operation
        $a_1\lambda + a_0 \leftarrow (a_1\lambda + a_0) \times (s_1\lambda + s_0)$//$(a_1\lambda + a_0) \times (s_1\lambda + s_0)$uses // multiplication operation

Output: $a_1x + a_0$

To calculate the running cost of the CPA, the following basic operations are considered:

- $(a_1\lambda + a_0) \times (s_1\lambda + s_0) = (2xs_1a_0 + s_1a_0 + s_0a_1)\lambda + (s_0a_0 - s_1a_1)$

  with the cost of$5t_m + 3t_a$.

- $(s_1\lambda + s_0) \times (s_1\lambda + s_0) = (2xs_1{}^2 + 2s_1s_0)\lambda + (s_0{}^2 + s_1{}^2)$

  with the cost of$2t_m + 2t_s + 3t_a$.

- $\lambda(s_1\lambda + s_0) = (2xs_1 + s_0)\lambda - s_1$

  with the cost of $t_m + t_a$.

Subsequently, the running costfor CPA can be calculated as follows:

$$T_{CPA} = (r-1)(2t_m + 2t_s + 3t_a) + (m-1)(5t_m + 3t_a) + 2(t_m + t_a) \quad (4)$$

The running time of the Modified Matrix Algorithm (MMA) and the Modified Characteristic Polynomial Algorithm (MCPA) (Li et al., 2011)can be calculated as follows:

$$T_{MMA} = (r-1)(3t_m + 2t_s + 3t_a) + m(2t_m + 2t_a) + (t_m + t_a) \quad (5)$$

$$T_{MCPA} = (r-1)(2t_m + 2t_s + 3t_a) + (m-1)(t_m + t_a) + (t_m + t_a) = (r-1)(2t_m + 2t_s + 3t_a) + m(t_m + t_a) \quad (6)$$

3. THE PROPOSED METHODS

3.1 Matrix Algorithm Based on Look-Up Table (LMA)

Following is the proposed method which utilizes look-up table while executing the MA. The bits representing the exponent n will be checked in reverse order (left-to-right) with group of b bits. In each loop, the result of the look-up table will be multiplied with the previous result which stored previous value in C,indicated as $C \leftarrow C \times LUT$in the pseudo-code. The look-up table LUT contains all powering results that can be accommodated by the b bits of the exponent n. Suppose that size of b is 3 bits, then the look-up table has8 entries. The LMA algorithm for the case of $b = 3$ is shown below:

Input:$n, A, I, t, b = 3$ // assume *n* is a multiple of 3
//Look-up table creation for the case of$b = 3$
$a[0]$ = null // not being used
$a[1] = A$
$a[2] = A^2 = A \times A$
$a[3] = A^3 = A^2 \times A$
$a[4] = A^4 = A^3 \times A$
$a[5] = A^5 = A^4 \times A$
$a[6] = A^6 = A^5 \times A$
$a[7] = A^7 = A^6 \times A$

$C \leftarrow I$
$For i = (n-1)\ down\ to\ 0\ by\ b$
  $t \leftarrow n_i n_{i-1} n_{i-2}$
  $If t \neq 0\ then$
    $C \leftarrow C \times a[t]$//multiplying the result with thelook-up table's value

$For\ j = 1\ up\ to\ b\ by\ 1$
$C \leftarrow C \times C$

Output: $xC_{11} + C_{12}$

The overall running time of the LMA can therefore be calculated as:

$$T_{LMA} = (r-1)(2t_m + 2t_s + 3t_a) + \frac{r-1}{b}(8t_m + 4t_a) + (t_m + t_a) + T_{LUT}, \quad (7)$$

where T_{LUT} is the time required to fetch a value from the look-up table.

3.2 Characteristic Polynomial Algorithm Based on Look-up Table (LCPA)

Similar to the LMA, the proposed LCPA drew the results of each loop computation from a pre-calculated look-up table. The bits representation of the exponent $n$is scanned in reverse order (left-to-right) with group of b bits. Following is an example of the LCPA pseudo-code for the case of b = *3*.

Input:n, A, I, t, b // assuming *n* is a multiple of 3
//Look-up table creation
$a[b_0]$ = null // not being used
$a[b_1]$= $(S_1\lambda + S_0)$
$a[b_2]$ =$(S_1\lambda + S_0) \times \lambda$
$a[b_3]$ =$(S_1\lambda^2 + S_0) \times \lambda$
$a[b_4]$ = $(S_1\lambda^3 + S_0) \times \lambda$
$a[b_5]$ = $(S_1\lambda^4 + S_0) \times \lambda$
$a[b_6]$ = $(S_1\lambda^5 + S_0) \times \lambda$
$a[b_7]$ = $(S_1\lambda^6 + S_0) \times \lambda$

$S_1\lambda + S_0 \leftarrow I$
$Fori = (n-1) down\ to\ 0\ by\ b$
$t \leftarrow n_i n_{i-1} n_{i-2}$
$If t \neq 0 then$
$S_1\lambda + S_0 \leftarrow (S_1\lambda + S_0) \times a[t]$

$For\ j = 1\ up\ to\ b\ by\ 1$
$S_1\lambda + S_0 \leftarrow (S_1\lambda + S_0) \times (S_1\lambda + S_0)$

Output:$xS_1 + S_0$

Subsequently, the running time of the LCPA can be calculated as follows:

$$T_{LCPA} = (r-1)(2t_m + 2t_s + 3t_a) + \frac{r-1}{b}(5t_m + 3t_a) + T_{LUT} \qquad (8)$$

4. ANALYSIS OF THE PROPOSED ALGORITHMS

Based on Equations 2, 5 and 7, the new LMA can be mathematicallycompared to the originalMA as follows

$$\begin{aligned}
&\mathrm{T_{MA} - T_{LMA}} \\
&\quad = \big((\mathrm{r}-1)(3\mathrm{t_m} + 2\mathrm{t_s} + 3\mathrm{t_a}) + \mathrm{m}(8\mathrm{t_m} + 4\mathrm{t_a}) \\
&\quad\quad + 3(\mathrm{t_m} + \mathrm{t_a})\big) \\
&\quad\quad - \left(\begin{array}{c}(\mathrm{r}-1)(3\mathrm{t_m} + 2\mathrm{t_s} + 3\mathrm{t_a}) + \dfrac{\mathrm{r}-1}{\mathrm{b}}(8\mathrm{t_m} + 4\mathrm{t_a}) \\ +(\mathrm{t_m} + \mathrm{t_a}) + \mathrm{T_{LUT}}\end{array}\right) \\
&\quad = \big(\mathrm{m}(8\mathrm{t_m} + 4\mathrm{t_a}) + 3(\mathrm{t_m} + \mathrm{t_a})\big) \\
&\quad\quad - \big(\tfrac{\mathrm{r}-1}{\mathrm{b}}(8\mathrm{t_m} + 4\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a}) + 2^{\mathrm{b}}(2\mathrm{t_m} + 2\mathrm{t_a})\big).
\end{aligned}$$

Similarly in the case of the MMA and LMA, the comparison is as follows:

$$\begin{aligned}
&\mathrm{T_{LMA} - T_{MMA}} \\
&= \left(\begin{array}{c}(\mathrm{r}-1)(3\mathrm{t_m} + 2\mathrm{t_s} + 3\mathrm{t_a}) + \dfrac{\mathrm{r}-1}{\mathrm{b}}(8\mathrm{t_m} + 4\mathrm{t_a}) + \\ (\mathrm{t_m} + \mathrm{t_a}) + 2^{\mathrm{b}}(2\mathrm{t_m} + 2\mathrm{t_a})\end{array}\right) \\
&\quad - \big((\mathrm{r}-1)(3\mathrm{t_m} + 2\mathrm{t_s} + 3\mathrm{t_a}) + \mathrm{m}(2\mathrm{t_m} + 2\mathrm{t_a}) \\
&\quad\quad + (\mathrm{t_m} + \mathrm{t_a})\big) \\
&\quad = \left(\begin{array}{c}\dfrac{\mathrm{r}-1}{\mathrm{b}}(8\mathrm{t_m} + 4\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a}) \\ + 2^{\mathrm{b}}(2\mathrm{t_m} + 2\mathrm{t_a})\end{array}\right) \\
&\quad - \big(\mathrm{m}(2\mathrm{t_m} + 2\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a})\big)
\end{aligned}$$

Figure 1, shows a theoretical comparison between LMA, MA, and MMA considering that the average of bit"1" in the exponent$n$ is at 50%. LMA is tested with $b = 7$. The comparison has shown that LMA has the less number of operations compared to the MA and almost equal in performance with MMA, with preference on MMA over LMA.

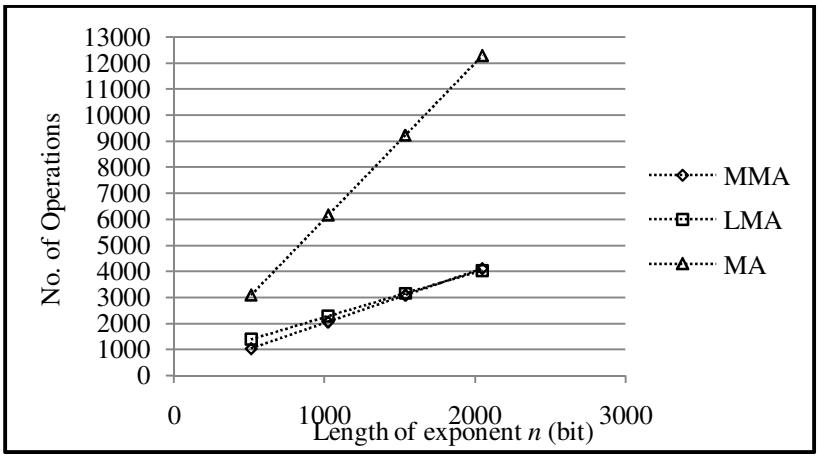

Figure 1: Theoretical comparison between MA, MMA, and LMA

Based on Equations4, 6 and 8, the different in the running cost among CPA, MCPA, and LCPA can be calculated. In the case of the CPA against LCPA, the mathematical comparison is as follows:

$$\begin{aligned} T_{CPA} - T_{LCPA} &= \big((r-1)(2t_m + 2t_s + 3t_a) + (m)(5t_m + 3t_a) \\ &\quad + 2(t_m + t_a)\big) \\ &\quad - \left(\begin{array}{c}(r-1)(2t_m + 2t_s + 3t_a) + \\ \frac{r-1}{b}(5t_m + 3t_a) + (t_m + t_a) + T_{LUT}\end{array}\right) \\ &= \big((m-1)(5t_m + 3t_a) + 2(t_m + t_a)\big) \\ &\quad - \left(\frac{r-1}{b}(5t_m + 3t_a) + (t_m + t_a) + 2^b(t_m + t_a)\right) \end{aligned}$$

Similarly in the case of the LCPA and MCPA, the mathematical comparison is as follows:

$$
\begin{aligned}
&T_{LCPA} - T_{MCPA} \\
&= \left(\begin{array}{c} (r-1)(2t_m + 2t_s + 3t_a) + \dfrac{r-1}{b}(5t_m + 3t_a) \\ +(t_m + t_a) + T_{LUT} \end{array} \right) \\
&\quad -\big((r-1)(2t_m + 2t_s + 3t_a) + (m \\
&\quad -1)(t_m + t_a) + (t_m + t_a)\big) \\
&= \left(\frac{r-1}{b}(5t_m + 3t_a) + 2^b(t_m + t_a)\right) \\
&\quad -\big((m-1)(t_m + t_a)\big)
\end{aligned}
$$

Thus, the LCPA has the extra of 541 multiplications and 135 additions than the MCPA.

Figure 2, shows a theoretical comparison between LCPA, CPA, and MCPA assuming 50% of bit "1" on the exponent n. LCPA is tested with $b = 7$. The comparison has shown that LCPA has less number of operations compared to the CPA and very close to MCPA with preference for MCPA over LCPA.

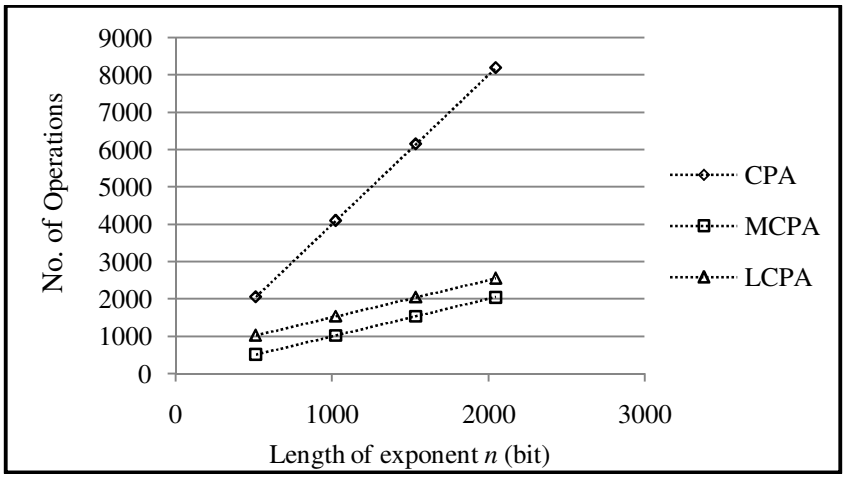

Figure 2: Theoretical comparison between CPA, MCPA, and LCPA

5. PRACTICAL APPLICATION OF THE PROPOSED SOLUTION

The propose technique has an advantage if the higher-level protocol that uses the technique has a fix base but varied exponent values. One of such instances is the key exchange protocol. In any key exchange protocol, the key server has to generate the common public values and publish them so that the respective communicating parties are able to generate their corresponding public keys and shared keys. In the case of the key exchange protocols based on the Chebyshev polynomial, the generation of all the keys needs to be based on the public value x as indicated in Equation 1.

The public value x is generated and published off-line. Consequently the value $x$is used to generate the look-up table, which can be done once and off-line as well. Since the look-up table is generated off-line, the cost of producing the table look-up can then be taken out from the cost calculation. As the result, this will help to reduce the execution time of calculating the Chebyshev polynomials when the user calculates the corresponding user's public and shared key. As a case study, the following example shows the calculation of the estimated execution time of a key exchange protocol that is based on Chebyshev polynomials if the overhead of preparing the look-up table is removed. Let assume the following:

- Size for the look-up table is$b = 15$
- The number of the exponent bits is$r = 1024$
- The number of1's inr is$m$which is 50% of r (Note that, on average, half of r bits are 1's)

If we exclude the overhead calculation needed in generating the look-up table, then we have the following:

1. For the MMA algorithm compared against the LMA algorithm, we have the following:

$$\begin{aligned} \mathrm{T_{MMA}} - \mathrm{T_{LMA}} &= \left(512(2\mathrm{t_m} + 2\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a})\right) \\ &\quad - \left(\frac{1024}{15}(8\mathrm{t_m} + 4\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a})\right) \\ &= (1024\mathrm{t_m} + 1024\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a}) \\ &\quad - \left((552\mathrm{t_m} + 276\mathrm{t_a}) + (\mathrm{t_m} + \mathrm{t_a})\right) \\ &= (1025\mathrm{t_m} + 1025\mathrm{t_a}) - (553\mathrm{t_m} + 277\mathrm{t_a}) \\ &= 472\mathrm{t_m} + (798\mathrm{t_a}). \end{aligned}$$

In this example, MMA requires 472 extra multiplications and 141 extra additions compared to LMA. Thus under this assumption, the LMA is faster than the MMA.

2. For the MCPA algorithm compared against the LCPA algorithm, we have the following:

$$\begin{aligned} T_{MCPA} - T_{LCPA} &= 512(t_m + t_a) - \left(\frac{1024}{15}(5t_m + 3t_a)\right) \\ &= (512t_m + 512t_a) - (345t_m + 207t_a). \\ &= (167t_m + 305t_a) \end{aligned}$$

Similarly, the existing algorithm, MCPA, hasan extra of 167 multiplications and 305 additions compared to the propose LCPA. Therefore, with the overhead of generating the look-up table being removed, LCPA will perform better than MCPA.

6. CONCLUSION

This paper has presented the most common algorithms that are being used in calculating the extended Chebyshev polynomial over the finite fields. These algorithms are the Matrix Algorithm (MA), Modified Matrix Algorithm (MMA), Characteristic Polynomial Algorithm (CPA) and the Modified Characteristic Polynomial Algorithm (MCPA). This paper had also introduced a technique (LMA and LCPA) to improve the MA and CPA calculation by using look-up table. The analysisindicate that the propose technique registered significant improvement over the MA and CPA but slightly under performance when compared against the MMA and MCPA. However, if the cost of seting-up the look-up table can be removed, which is the case for some protocols such as key exchange, the propose technique should then performed better than the existing methods. That is, LMA is better than MA and MMA, while LCPA is better than CPA and MCPA.

REFERENCES

Algehawi, M. B. and Samsudin, A. 2010. A new Identity Based Encryption (IBE) Scheme using Extended Chebyshev Polynomial Over the Finite Fields Zp. *Physics Letters A*. **374**: 4670-4674.

Fee, G.J. and Monagan, M.B. 2004. Cryptography using Chebyshev Polynomials. In *Proceedings of the Maple Summer Workshop MSW'04*. Burnaby, Canada, pp. 1-15.

Li, Z.-hui, Cui, Y.-dong and Xu, H.-min. 2011. Fast Algorithms of Public Key Cryptosystem Based on Chebyshev Polynomials Over Finite Field. *The Journal of China Universities of Posts and Telecommunications*. **18**(2): 86-93. Available at: http://linkinghub. elsevier.com/retrieve/pii/S1005888510600490 [Accessed October 12, 2011].

Liao, X., Chen, F. and Wong, K.-wo. 2010. On the Security of Public-Key Algorithms Based on Chebyshev Polynomials over the Finite Field Z_N. *IEEE Transactions on Computers*. **59**(10): 1392-1401. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm? arnumber=5487511.

Wang, X. and Zhao, J. 2010. An Improved Key Agreement Protocol Based on Chaos. *Communications in Nonlinear Science and Numerical Simulation*. **15**: 4052-4057.

Xiao, D., Liao, X. and Deng, S. 2007. A novel Key Agreement Protocol Based on Chaotic Maps. *Information Science*. 177: 1136-1142.

Yoon, E.-jun and Yoo, K.-young. 2008. A New Key Agreement Protocol Based on Chaotic Maps. In *Agent and Multi-Agent Systems: Technologies and Applications - KESAMSTA*. Berlin, Heidelberg: Springer-Verlag, pp. 897-906.