Hybrid-Key Agreement Protocol based on Chebyshev Polynomials

Ta Thi Kim Hue[1*], *Minh Duc Nguyen*[1], *Minh Hoang Vu*[1], *Hoang Manh Cuong*[2]
[1] *Hanoi University of Science and Technology - No. 1, Dai Co Viet, Hai Ba Trung, Hanoi, Viet Nam*
[2] *VNPT Technology - 124 Hoang Quoc Viet, Co Nhue, Cau Giay, Ha Noi, Viet Nam*
Received: August 10, 2018; Accepted: November 28, 2019

Abstract

This paper presented implementation of the Chebyshev permutation polynomials on hardware. The experimental results demonstrate that this is an efficient way to calculate the Chebyshev polynomials in a prime field. According to the hardware structure of the Chebyshev polynomial, a Hybrid-Key Agreement Protocol is proposed. The purpose of our protocol is to enable two end-users exchanging a secret session key using both the key distribution center and the Chebyshev-based public key encryption. Advantage of public-key encryption is authentic and confidential for delivering secret keys, the addition of KDCs serves a widely distributed set of users. The proposed key agreement protocol offers satisfactory security and can be implemented hardware efficiently suitable for the low resource utilization.

Keywords: Public key, Chebyshev, FPGA

1. Introduction

In cryptographic system, two or more parties can establish a session to share a key or be enable to the exchange of secret values by a key-agreement protocol. By this way, undesired third parties are not allowed to see the key, so the agreed key is not revealed to any eavesdropping party [1]. In general, there is one party in a key exchange system generating the key and then this key is distributed to other ones using for encryption [2]. Key distribution often consists of the master keys been lasting for long time but used infrequently, and session keys for temporary use between two parties. For those reasons, some sort of mechanism or protocol are proposed to deliver the secure session and master keys including a key distribution center and public-key infrastructure (PKI) [3]. In general, a public-key cryptosystem is applied to encrypt secret keys for distribution and the authenticity of the public key must be assured, several public key exchange schemes are commonly used for symmetric key agreement such as: RSA, Diffie-Hellman, Elliptic Curve Diffie-Hellman [4]. However, the key exchange based on public-key algorithms needs to the third party which is a certificate authority such as X.509 standard and each participant should have a public-key infrastructure. Consequently, public-key cryptographic systems inefficiently implement on low resource requirements or mobile devices. Because of the relatively high computational complexity of asymmetric key algorithm, secret keys are distributed by the public-key encryption leading to degrade overall system performance. Typically, the secret keys change frequently in each transaction, and then they are discarded. It means that a public-key distributed system is nearly ineffective in a wide-area distributed system because of a number of secret keys supplied dynamically. Therefore, the key distribution center is one of flexible ways to deliver the secret keys. A requirement for the use of KDCs is that KDCs be trusted and prevented from destruction [5]. In addition, the inverse problem of the discrete Chebyshev and the classical discrete logarithm are the computational complexity considered as equivalent. Authors in [6] showed that Chebyshev polynomials in finite fields fulfill cryptographic requirements and are also been applied to design a public-key encryption scheme in [7]–[9]. Moreover, Hue et al. in [10] presented a new signcryption scheme based on the Chebyshev chaotic map which is more efficient than elliptic curve-based scheme with respect to required hardware resources. The properties of Chebyshev polynomial in a finite field are considered to enhance security. For instances, authors proposed an efficient authentication protocol in [11], the key agreement protocol with Chebyshev polynomial sequences modulo a prime is introduced in [12].

In this paper, architecture hardware design of the discrete Chebyshev in the prime field is proposed. As a result, the proposed structure is either suitable for implementing on limited hardware resources or trade-

[*] Corresponding author: Tel.: (+84) 932.109.523
Email: hue.tathikim@hust.edu.vn

off between secure, performance and efficiency. According to the Chebyshev polynomial's hardware structure, we proposed a Hybrid-Key Agreement Protocol aimed to develop more efficient implementation on constrained devices. The proposed protocol retains both the KDC and public-key encryption based on the Chebyshev polynomial. Advantage of public-key encryption is authentic and confidential for delivering secret keys, the addition of KDCs serves a widely distributed set of users. By this way, distribution of the master key by the KDC is unique each time and then Chebyshev-based public-key encryption is used to update the session key between the end system users.

2. Implementation of permutation Chebyshev polynomial on hardware

The notations are denoted as in Table 1 throughout this paper

2.1. Properties of the Chebyshev permutation polynomials

The definition and characteristics of the Chebyshev polynomial are presented in articles [6], [7] and [13], in which application based on the Chebyshev polynomial in the field of cryptography or other potential applications are demonstrated in details.

Table 1. Summary of notations

p	a large prime
$GF(p)$	Galois Field of prime order
α_A	User A's secret key
α_B	User B's secret key
$Hash$	Hash function
E	Encryption algorithm
D	Decryption algorithm
$*$	Modular multiplication with p
$\ll$	Shift operator
ID_A	The idenfier of user A
ID_B	The idenfier of user B

The Chebyshev polynomials of the first kind is given as follows

$$\begin{cases} T_g(x) = 0 & if\ g = 0 \\ T_g(x) = 1 & if\ g = 1 \\ T_g(x) = 2xT_{g-1}(x) - T_{g-2}(x) & if\ g > 1 \end{cases} \quad (1)$$

which maps the interval $[-1, 1]$ with g times onto itself. Let g be a positive integer and x be a variable having a value over the interval $[-1, 1]$. The permutation polynomial (1) satisfies the semi-group properties:

1) $T_{nm}(x) = T_n(T_m(x))$
2) $T_n(T_m(x)) = T_m(T_n(x))$
3) $T_n\left(\frac{1}{2}(x + x^{-1})\right) = \frac{1}{2}(x^n + x^{-n})$

For our proposed applications, Chebyshev polynomials should be defined on a finite phase space, called a permutation polynomial. Let us consider the domain $R = F_p$, the finite field with p elements. The map $T_g: F_p \rightarrow F_p$ is defined by

$$y = T_g(x)\ mod\ p \quad (2)$$

where x is a positive integer, p is a large prime number and g is called an iterative coefficient. The following properties hold for the Chebyshev polynomials as $T_{nm}(x)\ mod\ p = T_n(T_m(x)\ mod\ p)$ (3)

In this paper, we considered the map $T_{g^\alpha}(x)$ with $T_g(x)$ iterated α times given as a formula below

$$T_{g^\alpha}(x)\ mod\ p = \underbrace{T_g(T_g(T_g \ldots T_g(x) mod\ p) \ldots)}_{\alpha\ iterationtimes} \quad (4)$$

From Eq.(4), multiplying two powers that have the same base, we have

$$T_{g^\alpha}\left(T_{g^\beta}(x) mod\ p\right) mod\ p = T_{g^{\alpha+\beta}}(x) mod\ p \quad (5)$$

Equation (1) gives

$$\begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix} \begin{pmatrix} T_{g-2}(x) \\ T_{g-1}(x) \end{pmatrix} = \begin{pmatrix} T_{g-1}(x) \\ T_g(x) \end{pmatrix} \quad (6)$$

and by induction

$$\begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix}^{g-1} \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} T_{g-1}(x) \\ T_g(x) \end{pmatrix} \quad (7)$$

Obviously, the computational complexity of the $T_g(x)$ from Eq.(8) is reduced to $O(log_2(g))$ [6] and the cost of the computation of $T_g(x)$ is 8 integer multiplications and 4 additions and 4 integer remainder operations for the matrix multiplication with individual elements reduced modulo p. Let us denote the recurrence relation matrix in Eq.(7) as a formula $A = \begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix}$. We have

$$A^{g-1} \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} T_{g-1}(x) \\ T_g(x) \end{pmatrix} \quad (8)$$

Therefore, $T_g(x)\ mod\ p \equiv A^e\ mod\ p$ with $e = g - 1$, by using the Chebyshev matrix powering algorithm, we can obtain $T_g(x)\ mod\ p$, but it takes more CPU time and a lot of resources because of its large integer multiplication [12]. To compute values of $T_g(x)\ mod\ p$ efficiently, we used the Cayley-Hamilton theorem [14], the characteristic polynomial of A is given by $f(\lambda) = \lambda^2 - 2x\lambda + 1$, if we defined $f(A) = A^2 - 2xA + I$ then $f(A) = 0$. The theorem gives $A^2 = 2xA - I$, observe $A^3 = AA^2 = A(2xA - I) = A(4x^2 + 1) + 2xI$. Likewise, $A^e = a(x)A + b(x)I$, where $a(x)$ and $b(x)$ are polynomials of x. We proposed a hardware structure to find the pair of $[a(x), b(x)]$ instead of powering the matrix A to obtain A^e modulo p. As a result, $T_g(x) = x(2xa(x) + b(x)) - a(x)$.

2.2. Hardware structure and performance analysis

According to the Cayley-Hamilton theorem, we have $A \equiv [A\ mod\ f(A)]\ mod\ p$. Supposed that the parameter e is presented by n bits, $e = 2^{n-1}e_{n-1} + 2^{n-2}e_{n-2} + \cdots + 2^2e_2 + 2e_1 + e_0$ with $e_i \in [0,1]$, thus

$$A^e = \prod_{i=0}^{n} (A^{2^i})^{e_i} \qquad (9)$$

hence,

$$A^e = \prod_{i=0}^{n} (\underbrace{A^2A^2 \dots A^2}_{i\ times})^{e_i} \qquad (10)$$

From Eq.(10), an efficient algorithm for computing $A^e\ mod\ p$ is described in Algorithm 1. By this way, we proposed the top-level implementation of the $T_g(x)\ mod\ p$, this architecture represents in Fig.1.

The hardware structure in Fig.1 has main components including the register file, the control unit, the shift register E_reg and modular operators (multiplication, addition and subtraction). In this design, the register file consists of temporary registers $a_0, a_1, r_0, r_1, x, 2x, t_1, t_2$. Let us assume that $A1$ and $A2$ are signals of the register address, $RD1$, $RD2$, $WD3$ and WE are data outputs and write handle signal, respectively. In the initialization state, a_1 = 1, a_0 = 0, r_1 = 0, r_0 = 1, $2x = (x \ll 1)$ and $t\_1 = t\_2 = 0$. x, g, p are registers contained input values. A ROM block is created to contain address of registers in the register file and synchronous signals are controlled by Counter and Control block.

Algorithm 1 Compute $A^e\ mod\ p$ by using the binary powering algorithm

```
1: procedure MODULO-POWER($A, e, p$)
2:     Initialization $R = 1$;
3:     $A = A\ mod\ p$;
4:     while ($e > 0$) do
5:         if ($e\ mod\ 2 == 1$) then $R = (R * A)\ mod\ p$;
6:         end if
7:         $e = e >> 1$;
8:         $A = (A * A)\ mod\ p$;
9:     end while
10:    Return $R$;
11: end procedure
```

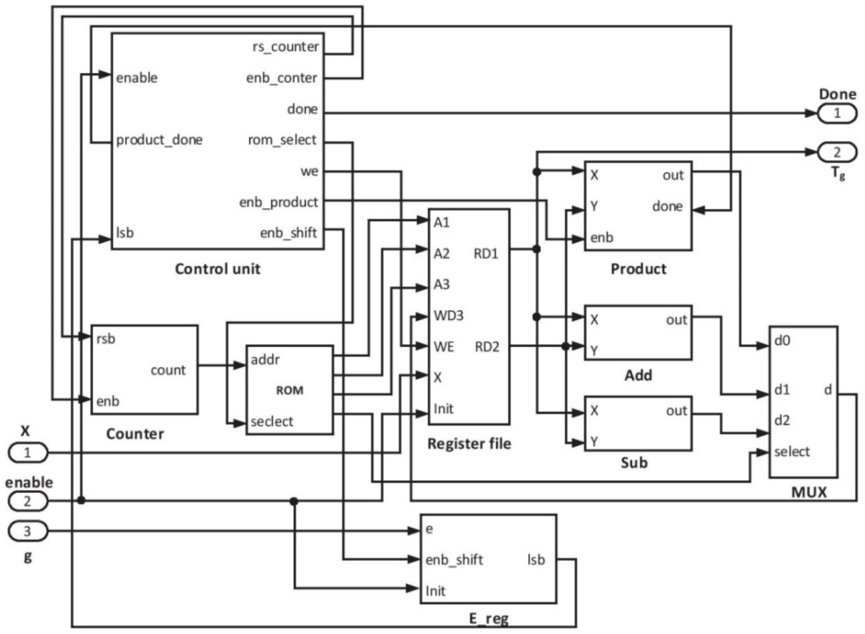

Fig. 1. Hardware Architecture of T_g

A shift register E_reg with $E = g - 1$, e is a LSB of E. Let us consider that $A_{vect} = [a_1\, a_0]$ and $R_{vect} = [r_1\, r_0]$ are the coefficient vectors corresponding with respectively. According to Algorithm 1, the equation the polynomial $A^e = a_1 A + a_0 \mathrm{I}$ and $R = r_1 A + r_0 I$, $R = R * A$ in step 5 is equivalent to $(a_1 A + a_0 I) * (r_1 A + r_0 I) = a_1 r_1 A^2 + A(a_1 r_0 + a_0 * r_1) + a_0 r_0 I$, since $\mathrm{A}^2 = 2xA - 1 = 2x(a_1 A + a_0 I)$, we have $\mathrm{R} = \mathrm{R}*\mathrm{A} = (r_0 * a_1 + r_1 * a_0 + 2x * r_1 * a_1)A + (r_0 * a_0 - r_1 * a_1)I$. The expression is executed as following steps:

1) $E = E >> 1$, check the $LSB\ e = [0, 1]$
2) $t_1 = r_1 * a_1$, $t_2 = a_0 * r_0 - t_1$
3) $r_1 = r_0 * a_1 + r_1 * a_0 + 2x * t_1$, $r_0 = t_2$
4) Update R = R ∗ A and A = A ∗ A
5) E = 0, we obtain $T_g(x) = x(2xr_1 + r_0) - r_1$

where registers t_1, t_2 contain temporary values of multiplication and addition.

In order to maximum security, p and x should be a large prime and a large integer, respectively [6]. In this design, registers have the length from 64 to 256 bits which storage the values of p and x, thus both p and x are chosen in ranges $[0, 2^{64} - 1]$ and $[0, 2^{256} - 1]$. Authors in [12] indicated that the larger the iterative coefficient g is, the more the storage space of $T_g(x)$ is. Using the hardware platform on ASIC, Fig.2 shows the area of ASIC implemention $T_g(x)$ with the bit length of p is corresponding with 64, 80, 128, 192 and 256 bits.

Assumed that $g' = g^{\alpha}$, we referred to the calculation problem $T_{g'}(x)\, mod\, p$. However, the value of g' increases rapidly according to α, so Algorithm 1 will be ineffective, it should take more C.P.U time. We proposed the hardware architecture of $T_{g^{\alpha}}$ in Fig.3, this is an efficient way to calculate the Chebyshev polynomials $T_{g^{\alpha}}(x)$ mod p accurately. This design is based on properties of the permutation polynomial over the finite field.

The period of $T_{g'}(x)\, mod\, p$ is $p - 1$ or $p + 1$ depending on the roots of the characteristic polynomial $f(\lambda) = \lambda^2 - 2x\lambda + 1$, the period p' is $p - 1$ if the roots are are in GF(p), otherwise, $p + 1$ when the roots are in GF(p^2) [15]. By this way, if $T_{p-1}(x) = T_0(x) = 1$ then $p' = p - 1$, else $p' = p + 1$. On the other hand, $T_{g'}$ mod p is equivalent to $T_{(g^{\alpha}\, mod\, p')}\, mod\, p$, instead of calculating g^{α}, we determine $g' = g^{\alpha}\, mod\, p'$.

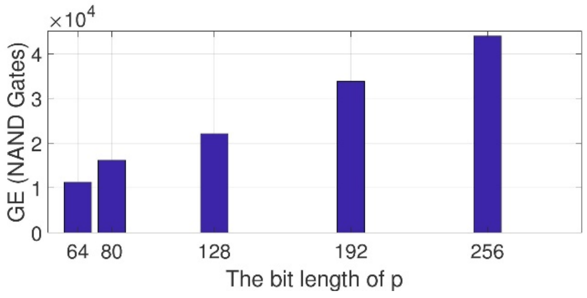

Fig. 2. Area of ASIC implementation $T_g(x)$ mod p

As can be seen in Fig.3, the mod-exp block undertakes calculating g^{α} mod p', the finite state machine (FSM) block is used to control the operation of others. All benchmarks were executed on a kit FPGA Kintex KC705. Table 2 showed the performance of $T_{g^{\alpha}}$ mod p with several values of x, p, g and α. It is clear that the more the bit length of x and p is, the slower processing speed and the more hardware resources are required.

3. Hybrid-key Agreement Protocol

In this section, we proposed a Hybrid-Key Agreement Protocol using the Chebyshev-based public-key encryption, called HKAChev. A hybrid approach is both the use of a the chebyshev-based public-key encryption and the key distribution center (KDC) to distribute the secret session keys between users. The proposed scheme is illustrated in Fig.4. Two elements including a security service and a Chebyshev-based key generation are embedded on each user's devices. The first element, a security service buffers packets and transmits a connection-request packet. The second one, a Chebyshev based key generation is created by the Chebyshev $T_{g^{\alpha}}$ module mentioned in Section 2. In the hybrid-key protocol, the session key is considered as a temporary key and used for the communication between end-user's devices in a certain duration, and then discarded. Each session key is transmitted in encryption form by Chebyshev-based public key scheme, using a master key shared by the KDC.

3.1. Key Agreement Protocol based on Chebyshev map $T_{g^{\alpha}}$

Figure 4 shows our proposed protocol that retains KDC to share the stream of parameters containing a master key. A Chebyshev-based public key scheme is applied to distribute the session key.

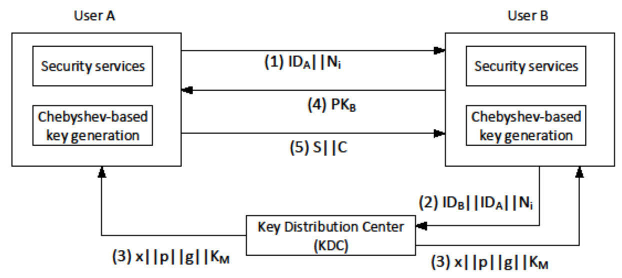

Fig. 4. Hybrid-Key Agreement Protocol based on Chebyshev polynomials

Let us suppose that User A wishes to establish a connection with User B and encrypt messages by a one-time session key on that connection. User A can issues a request with its identifier ID_A and a nonce N_i which is given as a time stamp to identify this transaction uniquely. User B sets up a transaction to KDC and sends the identifier of User B ID_B and $ID_A||N_I$. The KDC responds with the values of x, p, g and $K_M = hash\,\{ID_B||ID_A||N_i\}$ to both A and B. Then the following procedures are employed.

1) User B gets (x, p, g, K_M), calculating $K'_M = hash$ $\{ID_B||ID_A||N_i\}$ and checking that if $K'_M = K_M$ then choosing a secret key α_B and calculating the public key $PK_B = T_{g^{\alpha_B}}(x)$ mod p. User B transmits PK_B to A.

2) User A selects a random number α_A as a secret key and receives PK_B. User A can generate a one-time session key and send that to User B by the steps below

   - Generating $K_{SS} = T_{g^{\alpha_A - K_M}}(PK_B)$ mod p, hence $K_{SS} = T_{g^{\alpha_A+\alpha_B-K_M}}(x)$ mod p.
   - Calculating $C = E(K_{SS}, K_M)$ and $S = T_{g^{\alpha_A - K_M - c}}(x)$ mod p.

3) User B obtains (S, C), the following steps occur

   - Recovering $K'_{SS} = T_{g^{\alpha_B + c}}(S)$ mod p, hence $K'_{SS} = T_{g^{\alpha_A+\alpha_B-K_M}}(x)$ mod p
   - Recovering $K'_M = D(K'_{SS}, C)$ and if $K'_M = K_M$ then indicating $K'_{SS} = K_{SS}$ as the one-time session key.

4) The result is that both A and B know K_{SS}, therefore the session key K_{SS} can be used for securely communicating between A and B. Our proposed scheme provides either confidentiality or authentication for exchanging the secret key. At the next session, both A and B discard K_{SS} and make deal with each other to exchange a new session key.

In Table 3, time required for the generation of a single key pair 128-bit symmetric with different algorithms such as RSA, Diffie-Hellman (DH) and Elliptic curve Diffie-Hellman (EC) is shown in [1]. The keys generated in HKAChev protocol are 64, 80, 128, 192 and 256 bit width.

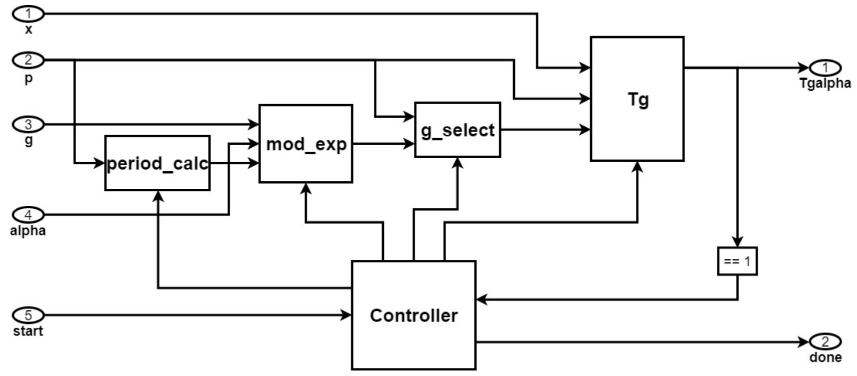

Fig. 3. Hardware Architecture of T_g^{α}

Table 2. Hardware Resource

Bit length of x, p, g	Bit length of α	Fmax(MHz)	Max Latency (cycle count)	Max delay time (ms)	Flip-flops
64	256	217	79236	0.365	1488
80	256	193	122084	0.563	1792
128	256	150	305924	1.409	2704
192	256	141	680068	3.134	3920
256	256	136	1201668	5.538	5136

3.2. Security analysis

The Hybrid-Key Agreement protocol (HKAChev) depicted in Figure 4 ensures against an attacker who can control the intervening communication between User A and B. In this case, an adversary, E, wants to compromise the communication channel without being detected and desires the session key. By eavesdropping, E can acquire a set of parameters involved $(S||C)$ and knows (x, g, p). E has seen $S = T_{g^{\alpha_A - K_M - c}}(x)\ mod\ p$ without known α_A. One way to break our proposed protocol, E should be first to exploit α_A and $\alpha_B$by solving both of the equations $\beta = T_\alpha(x)\ mod\ p$ and $\alpha = g^\gamma$. If $\beta = T_\alpha(x)\ mod\ p$ then $\alpha = log_{x+\sqrt{x^2+1}}(\beta + \sqrt{\beta^2 - 1})$ satisfied that $\sqrt{x^2 - 1}$ and $\sqrt{\beta^2 - 1}$ must be in the field $GF(p)$ or $GF(p^2)$. To solve a square root in $GF(p)$ or $GF(p^2)$, it takes an expected running time of $O\left(\exp\left((\varsigma + O(1))(\ln(n))^{\frac{1}{3}}(\ln(\ln(n)))^{\frac{2}{3}}\right)\right)$ where $\varsigma \cong 1.92$ in [6]. Besides, the assumption that E can get the value of α, the discrete logarithm $\gamma = log_g(\alpha)$ has no efficient solution. It means that it is infeasible to find α from β. Consequently, it is impossible to recover the value of α_A from S; In other words, no efficient method for recovering the value of K_{SS}. The shared session key is against the man-in-middle attack. For connection-oriented protocols, session keys are exchanged frequently. Each nonce N_i , we have only one K_M, assumed that A wants to change K_{SS} and B is unaware of changes. A new session key is created by randomly generating the value of α_A. User B received and authenticated the new session key by itself without going through KDC. As a result, an efficient key exchange protocol is established by the HKAChev one.

Table 3. Time required for the generation of a single key pair 128-bit symmetric

RSA	DH	EC	HKAChev
900(ms)	51(ms)	0.53(ms)	0.7045(ms)

As a replay attack, the attacker E may intercept $PK_B$when it is sent by User B in Step 1 and then will masquerade as User B next time. Since the attacker does not know the secret key of each user neither α_A or α_B, E can not compute a correct $S||C$, E must try all possible values of α_A. In this case, the α_A or α_B can be run with maximum 256 bits long. Thus, the computational complexity to brute-force all possible α_A combinations is 2^{256}. It implies that the HKAChev protocol also withstands the replay attack.

4. Conclusion

This paper has presented a Hybrid-Key Agreement protocol (HKAChev) which possessed both security attributes of public key based on the Chebyshev polynomial and efficient exchanging key scheme of KDC. The proposed protocol achieved desirable security levels resisting man-in-the middle, replay and brute-force attacks.

A model-based hardware design of the discrete Chebyshev polynomial in the prime field is illustrated. These results show that the proposed structure hardware requires a small resource and has more effective performance. Thus, it is potential for embedding in encryption applications. As an exemplar, this hardware structure is applied to create a Chebyshev-based key generation in HKAChev protocol. Both the theoretical analysis and experimental results show that the proposed key agreement protocol has a good security and potential for implementing on limited hardware resources.

Acknowledgments

This research is funded by Ministry of Science and Technology (MOST) under grant number 10/2018/TCT-KC.01/16-20.

References

[1]. W. Stallings, Cryptography and Network Security: Principles and Practice, Upper Saddle River, NJ, USA: Prentice Hall Press, 6th ed., 2013.

[2]. Dong Hwi Seo and P. Sweeney, Simple authenticated key agreement algorithm, Electronics Letters, vol. 35, pp. 1073–1074, June 1999.

[3]. P. E. Abi-Char, A. Mhamed, and B. El-Hassan, A secure authenticated key agreement protocol based on elliptic curve cryptography, in Third International Symposium on Information Assurance and Security, pp. 89–94, Aug 2007.

[4]. C. Boyd and A. Mathuria, Key Agreement Protocols, pp. 137–199. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[5]. C. Adams, Kerberos Authentication Protocol, pp. 674–675. Boston, MA: Springer US, 2011.

[6]. G. Maze, Algebraic methods for constructing one-way trapdoor functions. PhD thesis, University of Notre Dame Notre Dame, 2003.

[7]. L. Kocarev and S. Lian, Chaos-based Cryptography. Springer, 2011.

[8]. S. Vairachilai, M. K. Kavithadevi, and R. Gnanajeyaraman, Public key cryptosystems using chebyshev polynomials based on edge information, in

2014 World Congress on Computing and Communication Technologies, pp. 243–245, Feb 2014.

[9]. P. Bergamo, P. D'Arco, A. De Santis, and L. Kocarev, Security of public-key cryptosystems based on chebyshev polynomials, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 52, pp. 1382–1393, July 2005.

[10]. T. T. K. Hue, T. M. Hoang, and A. Braeken, Lightweight signcryption scheme based on discrete chebyshev maps, in 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 43–47, Dec 2017.

[11]. A. Braeken, P. Kumar, M. Liyanage, and T. T. K. Hue, An efficient anonymous authentication protocol in multiple server communication networks (eaam), The Journal of Supercomputing, vol. 74, pp. 1695– 1714, Apr 2018.

[12]. G. J. Fee and M. B. Monagan, Cryptography using chebyshev polynomials, in Laurier University, Waterloo, pp. 1–15, 2004.

[13]. J. Liu, D. Yang, H. Zhou, and S. Chen, A digital image encryption algorithm based on bit-planes and an improved logistic map, Multimedia Tools and Applications, p. 1, Nov. 2017.

[14]. R. Bronson and G. B. Costa, 7 - matrix calculus, in Matrix Methods (Third Edition) (R. Bronson and G. B. Costa, eds.), pp. 213 – 255, Boston: Academic Press, third edition ed., 2009.

[15]. D. Yoshioka, Properties of chebyshev polynomials modulo p^k, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, pp. 386–390, March 2018.