Proposing an efficient implementation method for exponentiation in digital signature scheme on ring Z_n

Nguyen Dao Truong[1*], Le Van Tuan[2], Doan Thi Bich Ngoc[3], Dang Duc Trinh[4]

[1]Academy of Cryptography Technique;
[2]Military Science Academy;
[3]University of Information and Communication Technology, Thai Nguyen University;
[4]Department of Information-Mathematics, Vietnam Military Medical University.
[*]Corresponding author: truongnd-it@actvn.edu.vn
Received 08 Jun 2022; Revised 16 Aug 2022; Accepted 07 Nov 2022; Published 18 Nov 2022.
DOI: https://doi.org/10.54939/1859-1043.j.mst.83.2022.72-81

ABSTRACT

In this paper, we propose a design method for the signature scheme based on ring structure Zn. Our signature schemes are more secure, generate signatures at a faster rate than that of the ElGamal scheme and its variants. Moreover, our approaches also overcome some disadvantages of some similar signature schemes on ring Zn. For these advantages, they are fully applicable in practice.

Keywords: Digital Signature Scheme; Discrete logarithm problem; Hash function.

1. INTRODUCTION

Nowadays, a number of asymmetrical payment methods have been developed to enable mobile users to buy goods online by charging them their mobile phone bills. It has been recognized that these methods must be used in conjunction with the security services of authentication and non-repudiation of the origin of the request(s) sent from a mobile user so as to prevent fraudulent actions by any other entities. Therefore, proposing a design method for a fast and secure signature scheme plays a very important role.

ElGamal's digital signature scheme relies on the difficulty of computing the discrete logarithm problem in the multiplicative group Z_p^* [1] and a series of variants of it [2, 3]. In general, the order of the multiplicative group Z_p^* of the ElGamal scheme and its variants could not be kept secret. That leads to be insecure when the session key is revealed or be coincided. Furthermore, these signature schemes publicize the order of multiplicative group Z_p^*, which led to be insecurity when adversaries attacked them from the Pollard's Rho, the Pohlig Hellman, and the Index calculate algorithms [5]. In order to deal with these insecure situations, recently scientists have proposed some signature schemes on ring Z_n[4, 6-9,11]. However, their schemes have not used the Chinese Remainder Theorem (CRT) for faster computing exponentiation or faster computing inverses [10]. This paper presents a Design Method of Digital Signature Scheme in which the CRT was used for computing exponentiation and inverses. Our research results can be applied in practice. Some important contributions of this paper are as follows:

Firstly, proposing a design method for a digital signature scheme based on discrete logarithm problems. Secondly, based on the proposed design method, we propose a signature scheme that has overcome the disadvantage of the ElGamal digital signature scheme and its variants. The speed of signature generation of the proposed schemes is also faster than that of ElGamal scheme and its variants on ring Z_n.

Section 2 presents notations and terminologies. In section 3, we propose some digital signature schemes. In section 4, analysis of signature schemes and testing. Section 5 finally is the conclusion.

2. NOTATION AND TERMINOLOGY

In this section, we first introduce the notation and terminology, used in the article's following sections.

Element k is randomly chosen from the set X, $k \in_R X$. The concatenation operation of the string x with the string y, $x||y$. The bit-length of the binary representation of a, denoted L_a. The order of g in ring Z_n, denoted $Ord_n(g)$. The greatest common divisor of integers a and b, denoted $GCD(a, b)$. Discrete Logarithm Problem on ring Z_n, denoted DLP_n. If $n = p$ is prime, it is denoted DLP_p.

$Hash: \{0,1\}^{\infty} \to \{0,1\}^{H}$ where H is a integer number. Let $s \in \{0,1\}^{H}$, let $s = s_0 ... s_{H-2} s_{H-1}$ is a binary string. So $\bar{s} \in \mathbb{N}$, that is computed by the following formula: $\bar{s} = s_0 2^{H-1} + \cdots + s_{H-2} 2 + s_{H-1}$. Let $n \in \mathbb{N}$, $n = n_0 2^{k-1} + \cdots + n_{k-2} 2 + n_{k-1}$ where $n_j \in \{0,1\}$ $(j = 0..(k-1))$ and $n_0 \neq 0$. Let $H \in \mathbb{N}$, $n[H] = n_{k-H} n_{k-H-1} ... n_{k-1}$ if $H \leq k$. $n[H] = \underbrace{0 ... 0}_{H-k} n_0 n_1 ... n_{k-1}$ nếu $H > k$. $n[H]$ will return a H-bit string.

3. THE PROPOSED DIGITAL SIGNATURE SCHEME ON RING Z_N

3.1. The generalized digital signature scheme - BSS

This section will propose a Design Method for Digital Signature Scheme Based on a Discrete Logarithm Problem on ring $\mathbb{Z}_n$.

The base set of the signature scheme: consists of five set (M, S, k_s, k_b, K) in which: $M = \{0,1\}^{\infty}$is a finite set of messages. $S = \mathbb{Z}_{2^H}^* \times \mathbb{Z}_t^*$ is a finite set of signatures. k_s, $k_s = \mathbb{Z}_t^*$ is a finite set of secret keys. k_b, $k_b = \mathbb{Z}_n^*$ is a finite set of public keys. $K, K = \mathbb{Z}_t^*$ is a finite set of session keys.

General parameters of the system: The length size of modulo number n, denoted L_n; Hash function, denoted Hash: $\{0,1\}^{\infty} \to \{0,1\}^{H}$, the length size of $t = Ord_n(g)$, denoted L_t and $L_t \geq H + 2$.

The generalized formula: Computing key: let x is secret key, the public key, denoted y, $y = KeyGen(x) = g^x \bmod n$. The first component of signatures is computed by the following formula: $r = (g^k \bmod n) \bmod 2^H$ where $k \in_R [1, t-1]$. The second component of the signatures is computed by the following formula: $s = k.(f_1(m, r).x + f_2(m, r))^{-1} \bmod t$.

Algorithm 1. Parameter of scheme BSS

Input: $L_p = L_q = \frac{L_n}{2}$;

Output: $((p, q), c_n, g, (g_p, g_q), t, (p_1, q_1), c_t, x, y)$;

1. Generate two distinct odd primes p the lengthsize of it is L_p −bit and lengthsize of q $is L_q$ −bit. Then compute n, $n = p.q$ and compute $c_n = q.(q^{-1} \bmod p)$.

2. Generate two distinct odd primes p_1, q_1 they satisfy some following conditions:

✓ $p_1 \neq q_1, p_1|(p-1), q_1|(q-1)$

✓ $q_1 \nmid (p-1)$ and $p_1 \nmid (q-1)$, $L_{p_1} = L_{q_1} = \frac{H}{2} + 1$

3. Compute $t = p_1.q_1$, $c_t = q_1.(q_1^{-1} \bmod p_1)$,

4. Primitive element, denoted g of multiplicative group, denoted $\mathbb{Z}_t^*$ and its order is t with $t = \mathrm{Ord}_n(g)$, that is computed by the following formula: $g = \alpha^{\frac{\varphi(n)}{p1.q1}} \bmod n$with $\forall \alpha, (\alpha, n) = 1$

5. Compute $g_p = g \bmod p$ and $g_q = g \bmod q$.

6. Secret key is denoted by x which is chosen randomly in $\mathbb{Z}_t^*$. The public key is denoted by y and that is computed $y = \mathrm{KeyGen}(x) = g^x \bmod n$.

Set of parameters $\left((p,q), c_n, (g_p, g_q), t, (p_1, q_1), c_t, x\right)$ are used for signer and set of parametures (n, g, y) are used for verifier.

Algorithm 2. Signature Generation of scheme BSS

Input: Message m and parameters $\left((p,q), c_n, (g_p, g_q), t, (p_1, q_1), c_t, x\right)$.

Output: (r, s) are a signature of message m.

1. $k \in_R [1, t-1]$; $k_p = k \bmod p_1$; $k_q = k \bmod q_1$;
2. if $(k_p = 0)$ *or* $(k_q = 0)$ then goto 1;
3. $r_p = g_p^{k_p} \bmod p$; $r_q = g_q^{k_q} \bmod q$; $r = \left(\left(c_n(r_p - r_q) + r_q\right) \bmod n\right) \bmod 2^H$; $f_1 = f_1(m, r)$;
4. if ($f_1 = 0$) then goto 1.
5. $f_2 = f_2(m, r)$; $w = (f_1.x + f_2)$; $w_p = w \bmod p_1$; $w_q = w \bmod q_1$;
6. if $(w_p = 0)$ or $(w_q = 0)$ then goto 1;
7. $z_p = w_p^{-1} \bmod p_1$; $z_q = w_q^{-1} \bmod q_1$; $z = \left(c_t.(z_p - z_q) + z_q\right) \bmod t$; $s = k.z \bmod t$;
8. return (r, s);

Algorithm 3. Signature verification of scheme BSS

Input: Message m's signature (r, s); set of parameters (n, g, y).

Output: Return "accept" or "reject".

1. if $(s = 0)$ then return "reject".
2. $f_1 = f_1(m, r)$; if ($f_1 = 0$) theo return "reject".
3. $a = y^{f_1} \bmod n$; $f_2 = f_2(m, r)$; $b = g^{f_2} \bmod n$; $c = a.b \bmod n$.
4. $w = (c^s \ (mod\ n)) \bmod 2^H$; if $(w = r)$ then return "accept" else "reject";

Proof of correctness:

The success probability of algorithm 2 depends on the end of three loops (step 2, step 4 and step 6). Lemma 3 proved that, the probability of these loop-end-events are approximately 1. So, the rest steps of algorithm 2 will terminate, obviously Algorithm 2 will success and return (r, s) of message m. The input of algorithm 3 include: the

signature of message m is denoted (r, s) and y (public key). Obviously, if (r, s) is a valid signature of message m then $s \neq 0$ and the value of w equals value r that will be proved as follows:

$s = k.w^{-1} \bmod t$; $s = k.\left(f_1(m,r).x + f_2(m,r)\right)^{-1} \bmod t$;

Then do following: $k = s.(f_1(m,r).x + f_2(m,r)) \bmod t$;

So $g^k mod\ n = g^{s.f_1(m,r).x}.g^{s.f_2(m,r)} \bmod n$.

Then do following

$$r = (g^k mod\ n)\ mod\ 2^H = g^{s.f_1(m,r).x}.g^{s.f_2(m,r)} \bmod n) \bmod 2^H \quad (1)$$

Value w is computed by algorithm 5 following:

$$w = \left(\left(y^{f_1(m,r)}.g^{f_2(m,r)}\right)^s\ mod\ n\right) mod\ 2^H \quad (2)$$

From formulas (1) and (2) that proved that $w = r$, algorithm 3 will terminate with success and to return "accept".■

3.2. Proposed signature scheme - SS01

The contents presented in this subsection are defining function $f_1(m, r)$ and $f_2(m, r)$ and proposing a signature scheme. The functions $f_1(m, r)$ and $f_2(m, r)$ are defined as follows. Functions $f_1(m, r) = 1$; $f_2(m, r) = \overline{Hash(m||r[H])}$.

We have the proposed signature scheme, denoted SS01, which consists of the signing algorithm, the signature verification algorithm. With the parameter generation algorithm that was inherited from BSS.

Algorithm 4. Generation signature of SS01

Input: Message m; Set of parameters $\left((p, q), c_n, (g_p, g_q), t, (p_1, q_1), c_t, x\right)$.

Output: (r, s) is the signature of message m.

1. $k \in_R [1, t-1]$; $k_p = k \bmod p_1$; $k_q = k \bmod q_1$;
2. if $(k_p = 0)$ *or* $(k_q = 0)$ then goto 1;
3. $r_p = g_p^{k_p} \bmod p$; $r_q = g_q^{k_q} \bmod q$; $r = \left((c_n(r_p - r_q) + r_q) \bmod n\right) \bmod 2^H$;
4. $f_2 = \overline{Hash(m||r[H])}$; $w = (x + f_2)$; $w_p = w \bmod p_1$; $w_q = w \bmod q_1$;
5. if $(w_p = 0)$ *or* $(w_q = 0)$ then goto 1;
6. $z_p = {w_p}^{-1} \bmod p_1$; $z_q = {w_q}^{-1} \bmod q_1$; $z = (c_t(z_p - z_q) + z_q) \bmod t$;
7. $s = k.z \bmod t$; return (r, s).

Algorithm 5. Signature verification of scheme SS01.

Input: (r, s) is the signature of message m; public key (n, g, y).

Output: Return "accept" or "reject".

1. if (s = 0) then return "reject";
2. $f_2 = \overline{Hash(m||r[H])}$. $w = (y.g^{f_2})^s \bmod n) \bmod 2^H$;
3. if (w = r) then return "accept" else "reject";

Proof of correctness: In subsection above, we showed that digital signature scheme, denoted BSS is correct so scheme SS01 is correct.■

3.3. Proposed signature scheme - SS02

The functions, denoted $f_1(m, r)$ and $f_2(m, r)$ that are defined as follows.

Function $f_1(m, r) = \overline{Hash(m||r[H])}$; $f_2(m, r) = 1$.

Algorithm 6. Signature generation of SS01

Input: Message m; Set of parameters $\left((p, q), c_n, (g_p, g_q), t, (p_1, q_1), c_t, x\right)$

Output: (r, s) is the signature of m.

1. $k \in_R [2, t-1]$; $k_p = k \bmod p_1$; $k_q = k \bmod q_1$;
2. if $(k_p = 0)$ *or* $(k_p = 0)$ then goto 1;
3. $r_p = g_p^{k_p} \bmod p$; $r_q = g_q^{k_q} \bmod q$; $r = \left((c_n(r_p - r_q) + r_q) \bmod n\right) \bmod 2^H$;
4. $f_1 = \overline{Hash(m||r[H])}$; if ($f_1 = 0$) then goto 1.
5. $f_2 = 1$; $w = (f_1.x + f_2) \bmod t$; $w_p = w \bmod p_1$; $w_q = w \bmod q_1$;
6. if $(w_p = 0)$ *or* $(w_q = 0)$ then goto 1;
7. $z_p = {w_p}^{-1} \bmod p_1$; $z_q = {w_q}^{-1} \bmod q_1$; $z = \left(c_t(z_p - z_q) + z_q\right) \bmod t$;
8. $s = k.z \bmod t$; return (r, s);

Algorithm 7. Signature verification of SS02

Input: The consists of parameters (n, g, y); The (r, s) is the signature of m

Output: Return "accept" or "reject".

1. if $(s = 0)$ then return "reject".
2. $f_1 = \overline{Hash(m||r[H])}$, if ($f_1 = 0$) then return "reject".
3. $a = y^{f_1} \bmod n$; $f_2 = 1$; $b = g^{f_2} \bmod n$; $c = a.b \bmod n$;
4. $w = (c^s \bmod n) \bmod 2^H$; if $(w = r)$then return "accept" else "reject";

Proof of correctness: In subsection above, we proved that the scheme BSS is correct so scheme SS02 is correct. ■

4. ANALYSIS OF THE SIGNATURE SCHEME AND TESTING

Suppose that, the bitlength of the modulus number (denoted n) of the proposed schemes and the RSA scheme equal, $n = p.q$, in which p and q are L −bit primes. Suppose $L_p = L_q = \frac{L_n}{2} = L$. Let the modulus number of ElGamal's scheme (denoted P) is a $2L$ −bit prime, $L_P = L_n = 2L$. Suppose that $H \geq 512$, $t = p_1.q_1$ and $L_t \geq 514$. The cost of computing of ElGamal scheme and RSA scheme [12] were reviewed in this paper.

4.1. Computational complexity

Let the number of bits in the binary representation of two positive integers a and b is L −bit, the number of bit operations of the modular multiplication of a and b is denoted M_l. Assume that the modular exponentiation operation $a^e \bmod n$ is denoted function $Pow(a, e, n)$.

Lemma 1:

Recall that, consists of a, e, n are (L_a, L_e, L_n −bit) non-negative integers. The number of bit operations of the function $Pow(a, e, n) = a^e \bmod n$ is denoted $C_{Pow_{(L_a, L_e, L_n)}}$, it is estimated as follows:

$$C_{Pow_{(L_a, L_e, L_n)}} \approx 1{,}5.L_e.M_{L_n} \quad (3)$$

Proof:

Recall that L_n be the bitlength of the binary representation of n. Let L_e be the bitlength of the binary representation of e and let w(e) be the number of 1 is in representation of e. The using of efficient modular multiplication and exponentiation algorithms for computing function $Pow(a,e,n) = a^e \bmod n$ requires L_e of multiplication and w(e) of squaring in $\mathbb{Z}_n$. According to [5], $w(e) \approx \frac{L_e}{2}$. If squaring is approximate–as costly as an arbitrary multiplication then $C_{Pow_{(L_a,L_e,L_n)}}$ is roughly the following formula:

$$C_{Pow_{(L_a,L_e,L_n)}} \approx (L_e + \frac{L_e}{2}).M_{L_n} = 1{,}5.L_e.M_{L_n} \blacksquare$$

Lemma 2. Let the bit-length of value n, denoted $L_n = 2L$. Recall that the e and n of roughly equal bit-length. If modulus n is factored as follows: $n = p.q$ and $L_p = L_q = \frac{L_n}{2} = L$. Then computing $a^e \bmod n$ can execute be approximately 4 times faster and computing $a^{-1} \bmod n$ can execute be approximately 4 times faster

Proof:

(i) Suppose p and q are L-bit primes, and let n = p.q. if not having $n = p.q$ then according to Lemma 1 to compute $a^e \bmod n$ requires $1{,}5.2.L.M_{2L} \approx 12.L^3$ bit operations. On the other hand, having $n = p.q$, then according to [5] that compute $a^{e_p} \bmod p$ and $a^{e_q} \bmod q$ (in which $e_p = e \bmod (p-1)$ and $e_q = e \bmod (q-1)$ require $2.(\frac{3}{2}.L^3)$ bit operations. So the speed of computing $a^e \bmod n$ is about $\frac{12.L^3}{2.1{,}5.L^3} = 4$ times faster. ■

(ii) Suppose p and q are L-bit primes and let $n = p.q$. According to [5], if not having $n = p.q$, compute $a^{-1} \bmod n$ requires $\approx (2L)^2 = 4L^2$. On the other hand, if having $n = p.q$, then according to [5] that compute $x_p = a^{-1} \bmod p$ and $x_q = a^{-1} \bmod q$ require $2L^2$ bit operations. So, the speed of computing $a^{-1} \bmod n$ is about $\frac{4.L^2}{2.L^2} = 2$ times faster.■

Lemma 3. Suppose that g is a generator α of the unique cyclic group of order t in ring $\mathbb{Z}_n (t = Ord_n g)$. Let t_p, t_q are distinct primes, each roughly the same size and satisfying $t = t_p.t_q > 2^H$. The probability of the loop in proposed schemes is executed only one time be approximately 1 when $H \geq 512$.

Proof:

Apparently the probability of the loop in proposed schemes is executed only one time when consists of events $(k_p \neq 0)$ or $(k_q \neq 0)$ or $(w_p \neq 0)$ or $(w_q \neq 0)$ occur from the first loop. The consists of event $(k_p = 0)$, $(k_q = 0)$, $(w_p = 0)$, $(w_q \neq 0)$ just occur with the probability $\frac{1}{t}$ (when $k = p_1$ or $k = q_1$ or $w = p_1$ or $w = q_1$). Since $L_t = H + 2$, so $\frac{1}{t} = \frac{1}{2^{H+2}} \approx 0$ with $H \geq 512$. Furthermore, the hash function is based on the secure hash Algorithm, such as SHA-1, so $(f_1 = \overline{Hash(m||r[H])}) \neq 0$ with the probability 1■

4.2. Computational costs

What remains in this session are some analysis and comparison of the computational costs? We focus mainly on analyzing the number of bit operations of the modular multiplication, the modular exponentiation, and the modular inverses in consists of the

proposed schemes, ElGamal's Scheme and the RSA scheme. The Inverses in $\mathbb{Z}_t$ can be computed by using the extended Euclidean algorithm. According to [5], computing an inverse in $\mathbb{Z}_t$ is as approximately costly as modular multiplication. In Lemma 3, It proved that the loop event executes once with the probability ≈1, so we suppose that the algorithm of the proposed schemes hasn't the loop. Algorithm 4, if excluding addition, subtraction, modulus reduction then the computational costs of algorithm as follows: two exponentiations in $\mathbb{Z}_p$ these require $2(1{,}5.\frac{L_t}{2}M_L)$ bit-operations; one multiplication in $\mathbb{Z}_p$, and requires M_{2L} bit operations; two inverses in $\mathbb{Z}_{p_1}$and $\mathbb{Z}_{q_1}$that require $2M_{\frac{L_t}{2}}$ bit operations; two multiplications in $\mathbb{Z}_t$ and require $2M_t$ bit operations. So, the total cost of algorithm 4 are $3L_t.M_L + M_{2L} + 2M_{\frac{L_t}{2}} + 2M_t$ bit operations. In algorithm 5, if excluding the modulus reduction and the checking ($s = 0$) then the computational costs of algorithm 5 is: two exponentiations (L_s-bit exponents and $L_{f_2 s}$-bit exponents) and one multiplication in $\mathbb{Z}_n$. So, the total of computational cost of algorithm 5 is $1{,}5.\ L_s M_{2L} + 1{,}5.L_{f_2 s}.M_{2L} + M_{2L}$ bit operations.

Algorithm 6, if excluding addition, subtraction, modulus reduction then the computational costs of algorithm following: $3L_t.M_L + M_{2L} + 2M_{\frac{L_t}{2}} + 3M_t$bit operations. In algorithm 7, if excluding the modulus reduction and the checking ($s = 0$) then the computational costs of algorithm 7 is as approximately costly as the computational costs of algorithm 5, then the expected amount of bit operations is about $1{,}5.L_{f_2 s}.M_{2L} + M_{2L}$.

4.3. Storage space costs

Each signature is generated by the proposed schemes that require storage $2L_t$ −bit, it is much smaller than the storage space consists of ElGamal's signature, RSA's signature. If we review the proposed schemes (SS01 scheme, SS02 scheme) in the generalization case, each member of the system own a set of individual parameters (consists of 11 components require $7L + 5L_t$ −bit), and their storage space requirement depends on the number of members in the system. Recall that, the proposed signature scheme has K members, the proposed schemes required storage space be $C_K^2(7L + 5L_t)$ −bit, in which C_K^2 is the convolution instances k of n elements. Therefore, the proposed schemes require a storage space that are much larger than the storage space of ElGamal's scheme and its variant schemes on the field $\mathbb{Z}_p$. The results of the estimation of the computational cost and storage space of the proposed signature scheme are shown in table 1.

Table 1. *Number of bit operations and storage spaces.*

Scheme	Signature generation (number of bit operations)	Signature Verification number of bit operations	Signature spaces	Parameter spaces
RSA	$\approx 3L.M_{2L}$	$\approx 128.M_{2L}$	$\approx 2L$	$\approx 6.L$
ElGamal	$\approx 3L.M_{2L} + 3.M_{2L}$	$\approx 9.L.M_{2L} + 2.M_{2L}$	$\approx 4.L$	$\approx 6.L$
SS01	$\approx 3L_t.M_L + M_{2L} + 2M_{\frac{L_t}{2}} + 2M_t$	$\approx 1.5.L_{f_2 s}.M_{2L} + M_{2L}$	$\approx 2.L_t$	$\approx C_K^2.(7L + 5L_t)$
SS02	$\approx 3L_t.M_L + M_{2L} + 2M_{\frac{L_t}{2}} + 3M_t$	$\approx 1.5.L_{f_1 s}.M_{2L} + M_{2L}$	$\approx 2.L_t$	$\approx C_K^2.(7L + 5L_t)$

4.4. Security analysis

Proposition 1. The proposed schemes are secure from the situations of coinciding or revealing of session key

Proof: Ideally, a session key is an ephemeral secret, that is generated in each transaction. According to the "birthday paradox", the possibility of key revealing still occurs quite high. However, the order of primitive element (denoted $Ord_g = t$) of the proposed signature schemes was kept secret, so it is secure against attacks of revealing or coinciding session key:

(i) Review algorithm 4 (scheme SS01) and algorithm 6 (scheme SS02). Recall that, a session key (denoted k) was revealed, because t is kept secret, so adversaries cannot determine w from the following formula: $w = (f_1.x + f_2) \bmod t$. Since it is impossible to determine w, so it can't determine x and our proposed scheme can't be forged by adversaries.

(ii) Review algorithm 4 (scheme SS01) and algorithm 6 (scheme SS02). Recall that, a session key (denoted k) coincided, because t is kept secret, so adversaries can't determine w from the following formula: $w = (f_1.x + f_2) \bmod t$. So it can't determine x from $x = {f_1}^{-1}.(w - f_2) \bmod t$. Therefore, our proposed scheme can't be forged by adversaries.

Proposition 2: The proposed signature schemes avoid the attacking by Pohlig Hellman algorithm, Pollard's Rho algorithm and Index calculate algorithm.

Proof: It can be seen that the proposed signature schemes are secure as an integrity mechanism, since the inputs of all the above algorithms must have the order of the generator element g, (denoted $t, t = Ord_g(n)$). Because t is kept secret, so adversaries can't determine x by using the Pohlig Hellman algorithm, the Pollard's Rho algorithm and the Index calculate algorithm. ■

Safety threshold: Similar to the digital signature schemes being applied in practice, in order to apply the two proposed signature schemes SS01 and SS02 in practice, the security threshold must first be determined[11] and its safety parameter standards[4] at application time. The determination of safety thresholds and safety standards will be presented by the authors in the next articles.

4.5. Testing simulation

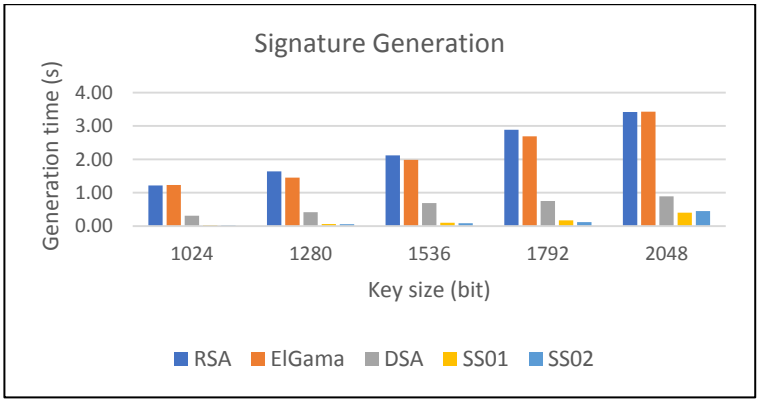

Figure 1. *Signature generation times of different schemes.*

Recall that, $L_P = L_n = 2L$ =1024, 1280, 1536, 1792, 2048 (bit). The message's size of the testing is 25.87 MB. The Computer's configuration is CPU Intel(R) Core(TM)2/3.00GHz, the physical memory 2 Gbyte. The hash function used for testing be the SHA 512. The results of testing are shown in figure 1 and figure 2.

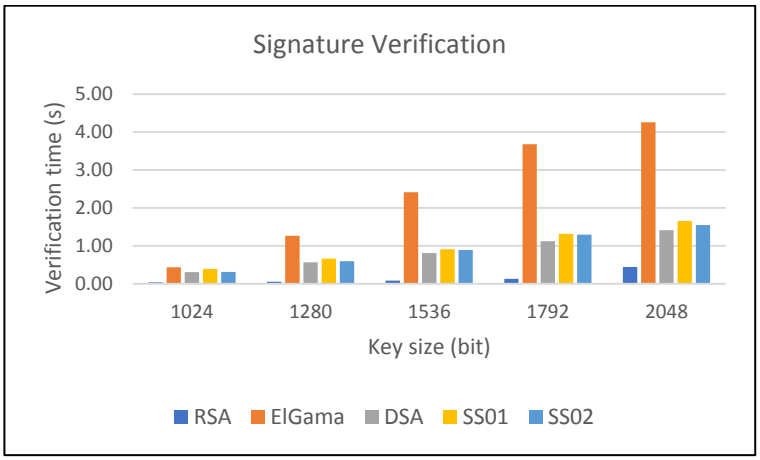

Figure 2. *Signature verification times of different schemes.*

Figures 1 and 2 show that the signing speed of our two proposed schemes is approximately 50 times faster than it of RSA and ElGamal. In addition, our two proposed schemes also have 10 times faster signing speed than DSA's signature scheme. However, for signature verification, the verification speed of our two proposed schemes is slower than that of other schemes. This is because our verification technique is not optimal. In the next research time, we will improve this for better.

5. CONCLUSIONS

In this paper, we proposed a Design Method for Digital Signature Scheme Based on Discrete Logarithm Problem. We proposed a method for designing digital signature schemes based on the discrete logarithm problem in this article. The proposed scheme's security is based on the discrete logarithm problem on ring Z_n, (denoted DLP_n) in which n is a product of two primes. With this approach, in our scheme, the order of primitive elements (denoted $ord_n(g)$) can be kept secret, so it is secure against not only session key revealing or session key coinciding situations but also using algorithms solving such as Pollard's Rho algorithm, the Pohlig Hellman algorithm or the Index algorithm calculate algorithm. Even the CRT was applied for computing exponentiation and inverses in our proposed schemes, so its signature generation speed is faster than that of similar schemes on ring Z_n. Therefore, they are suitable for smart cards. Admittedly, our proposed scheme's storage space costs are much larger than the similar scheme's storage space costs.

REFERENCES

[1]. Dimitrios Poulakis and Robert Rolland. *"A Digital Signature Scheme based on two hard problems."* https://eprint.iacr.org/2012.

[2]. Ng, Tiong-Sik, Syh-Yuan Tan, and Ji-Jian Chin. *"A variant of Schnorr signature scheme with tight security reduction."* 2017 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, (2017).

[3]. Morita, Hiraku, et al. *"On the security of the schnorr signature scheme and DSA against related-key attacks."* ICISC 2015. Springer, Cham, (2015).

[4]. Tuan Le Van, *"Developing and constructing parameters for digital signature scheme on discrete logarithmic problem by composite modulus"* Military Technical Academy, PhD thesis, Ha Noi, (2019).

[5]. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *"Handbook Applied Cryptography"*, Webster Professor of Electrical Engineering and Computer Science Massachusetts Institute of Technology June (1996).

[6]. Duy Ho Ngoc, Van Vu Long,Tuan Nguyen Kim, Thuy nguyen Thi Thu, *"A Solution to improve security for digital signature scheme"*, SOIS Ho Chi Minh City, No 2, pp.13-16, (2017).

[7]. Berezin, A. N., N. A. Moldovyan, and V. A. Shcherbacov. *"Cryptoschemes based on difficulty of simultaneous solving two different difficult problems."* Computer Science 21.2: 62 (2013).

[8]. Meshram, Chandrashekhar. *"Discrete Logarithm and Integer Factorization using ID-based Encryption."* Bulletin of Electrical Engineering and Informatics 4.2: 160-168 (2015).

[9]. Tripathi, Shailendra Kumar, and Bhupendra Gupta. *"An efficient digital signature scheme by using integer factorization and discrete logarithm problem."* 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, (2017).

[10]. *"Cryptographic Mechanisms: Recommendations and Key Lengths"*, TR-02102-1 v2020-01, BSI, (03/ 2020).

[11]. Lê Văn Tuấn, Tạ Minh Thanh và Bùi Thế Truyền, *"Phát triển lược đồ chữ ký số Elgamal trên vành Zn ngăn ngừa tấn công dựa vào tình huống lộ khóa phiên hoặc trùng khóa phiên"* , Tạp chí ITC, số 13 (6-2019), (in Vietnamese).

TÓM TẮT

Đề xuất một phương pháp thiết kế lược đồ chữ ký số dựa trên bài toán logarit rời rạc trên vành Zn

Trong bài báo này, chúng tôi đề xuất một phương pháp thiết kế lược đồ chữ ký số dựa trên độ khó của bài toán logarit rời rạc trên vành Zn. Những lược đồ đề xuất của chúng tôi an toàn hơn, việc tạo chữ ký được thực hiện nhanh hơn so với lược đồ ElGamal và các biến thể của nó. Ngoài ra, phương pháp thiết kế đề xuất của chúng tôi cũng có các chi phí tốt hơn so với các lược cùng loại trên vành Zn. Lược đồ đề xuất của chúng tôi có thể được áp dụng trong thực tế về sau.

Từ khoá: Digital Signature Scheme; Discrete logarithm problem; Hash Function.