DIGITAL SIGNATURE SCHEMES BASED ON DISCRETE LOGARITHM OVER FINITE FIELDS

Melek D.Yücel*, Alpaslan Lorasdağı**
* EE Dept. of METU, TÜBİTAK-BİLTEN
** Karel Electronics Inc.

Abstract *– **Possible ElGamal like digital signature schemes are examined and required numbers of additions, multiplications, exponentiations and inverses are tabulated. Complexities of inverse and exponentiation algorithms are evaluated in terms of multiplication subblocks. Most effective schemes are selected and compared versus time duration criterion.***

1. INTRODUCTION

Today, most of the communications activities are done over open computer networks, since they provide opportunity of fast, effortless, and inexpensive communications between different regions of the world. However, maintaining secure communications over an open channel is an important problem.

There are at least three requirements of communications security: Privacy, authentication, and integrity. Privacy can be provided by encyption and decryption. In addition to encryption and decryption , "Digital Signature" concept is developed to provide authenticity and integrity features.

Depending on these requirements, two types of cryptographic systems have appeared. First one is the "conventional cryptographic system" which relies on the use of a single piece of private and necessarily secret information known as the key. So, conventional cryptography can also be called "single-key cryptography" or "secret-key cryptography". Second one is called "public-key cryptography", or "two-key cryptography". It differs from the conventional cryptography in that there is no longer a single secret key shared by two users. There exist two keys per entry. One key which is publicly revealed and called "public-key", is used for encryption; the other key which is kept in secret and called "private-key", is used for decryption. The public-key cryptosystems are more advantageous, since they do not require any key distribution over the insecure channels. But, the computation time required by single-key cryptosystems is less. So, some hybrid systems which combine both type of cryptosystems, are developed.

In 1984, Tahar ElGamal proposed a new signature scheme[1], together with an implementation of the Diffie-Hellman key distribution scheme that achieves a public key cryptosystem. The security of system relies on the difficulty of computing discrete logarithm over the finite fields. This scheme has become very popular and formed a basis for Digital Signature Standard (DSS) which was proposed by U.S. government for electronical verification of the integrity and source of the unclassified information.

After the proposal of ElGamal digital signature scheme, some modified versions have appeared. These schemes have implemented by interchanging the parameters of ElGamal digital signature scheme. In this paper, ElGamal like digital signature schemes [2] are worked.

2. ELGAMAL DIGITAL SIGNATURE SCHEME

The ElGamal digital signature algorithm [1] can be summarized as follows:

Let p be a large prime, at least of length 512 bits, and α be a primitive element of GF(p). Both of them are publicly known.

Let m be a document (or the hash of the document) to be signed and $0 \le m \le p-1$.

x is the private key and y is the public key so that:

$$y = \alpha^x \bmod p \qquad (1)$$

for each user.

If a user whose public key is y and private key is x, wants to sign a document m, a signature pair (r, s) is constructed such that:

$$\alpha^m \equiv y^r r^s \bmod p \qquad (2)$$

where $0 \le r,s \le p-1$, is satisfied.

The Signing Procedure:

1. Choose a random number k, $0 \le k \le p-1$, so that $\gcd(k , p-1) = 1$.
2. Compute the parameter r as follows:
$$r \equiv \alpha^k \bmod p. \qquad (3)$$
3. Compute the parameter s as follows:
$$s \equiv k^{-1}(m - xr) \bmod (p-1) \qquad (4)$$

Equation (4) has a unique solution for s, since k is chosen such that $\gcd(k , p-1) = 1$.

The Verification Procedure:

The message (or its hash) m and the digital signature (r , s) are known. So check whether equation (2) given as $\alpha^m \equiv y^r r^s \bmod p$, is satisfied or not. If it is satisfied then the digital signatureis verified.

If the message m is altered during transmission, or a public key y′ different from the actual public key y is used, equation (2) cannot be satisfied; therefore message integrity and user authenticity is guaranteed.

3. ELGAMAL LIKE DIGITAL SIGNATURE SCHEMES

L. Harn and Y.Xu [2] have made a complete list of possible digital signature schemes modified from ElGamal algorithm. They have presented a parametric equation for all digital signature schemes of the type:

$$ax = (bk + c) \bmod (p-1) \quad (5)$$

where the parameters (a, b, c) can be some simple mathematical combinations of the values (m, r, s, 1) as given in Table I. So the verification for the above parametric equation becomes similar to equation (2):

$$y^a = r^b \alpha^c \bmod p \quad (6)$$

Harn and Xu omit other simple combinations of values (m, r, s, 1), since some restrictions are needed for the sake of security [2].

TABLE I
PROPOSED COMBINATIONS OF PARAMETERS (a, b, c)

Alg. No	a	b	c
1	r	s	m
2	m	s	r
3	r	m	s
4	m	r	s
5	s	r	m
6	s	m	r
7	r.m	1	s
8	1	r.m	s
9	s	1	r.m
10	1	s	r.m
11	r.m	s	1
12	s	r.m	1
13	r+m	1	s
14	s	1	r+m
15	1	r+m	s
16	1	s	r+m
17	r+m	s	1
18	s	r+m	1

Signing and verification equations for ElGamal like digital signature schemes can be found by substituting the proposed combinations of (a, b, c) parameters into equations (5) and (6).

We first summarize the notation:

p:the prime number ,

m:the sent message and $0 \le m \le p-1$,

r,s:signature pair and $0 \le (r,s) \le p-1$,

α:primitive element of GF(p),

x:the private key,

y:the public key equals to $\alpha^x \bmod p$,

k:the random number with $0 \le k \le p-1$ satisfying $\gcd(k, p-1) = 1$.

If the difference between +d and -d, and the difference between d and d^{-1}, where $d \in (x, k, m, r, s)$ are neglected, all the signing and verification operations for these schemes can be listed below:

1) $rx = (m + ks) \bmod (p-1)$: (ElGamal Scheme) [1]
Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(rx - m) \bmod (p-1)$
Verifying the signature:
Check whether $r^s \alpha^m \bmod p = y^r \bmod p$

2) $mx = (ks + r) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(mx - r) \bmod (p-1)$
Verifying the signature:
Check whether $y^m \bmod p = \alpha^r r^s \bmod p$

3) $xr = (mk + s) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = (rx - mk) \bmod (p-1)$
Verifying the signature:
Check whether $y^r \bmod p = \alpha^s r^m \bmod p$

4) $mx = (rk + s) \bmod (p-1)$: (Harn Scheme 1) [3]
Signing the message m:
$r = \alpha^k \bmod p$, $s = (mx - kr) \bmod (p-1)$
Verifying the signature:
Check whether $y^m \bmod p = r^r \alpha^s \bmod p$

5) $sx = (rk + m) \bmod (p-1)$: (AMV Scheme) [4]
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(rk + m) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = r^r \alpha^m \bmod p$

6) $sx = (mk + r) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(mk + r) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = \alpha^r r^m \bmod p$

7) $rmx = (k + s) \bmod (p-1)$: (Optimal Scheme) [5]
Signing the message m:
$r = \alpha^k \bmod p$, $s = (rmx - k) \bmod (p-1)$
Verifying the signature:
Check whether $y^{m.r} \bmod p = r\alpha^s \bmod p$

8) $x = (s + krm) \bmod (p-1)$: (Yen-Lain Scheme) [6]
Signing the message m:
$r = \alpha^k \bmod p$, $s = (x + krm) \bmod p$
Verifying the signature:
Check whether $r^{r.m} \alpha^s \bmod p = y \bmod p$

9) $sx = (k + mr) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(k + mr) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = r\alpha^{m.r} \bmod p$

10) $x = (sk + rm) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(x - rm) \bmod (p-1)$
Verifying the signature:
Check whether $y \bmod p = \alpha^{r.m} r^s \bmod p$

11) $mrx = (ks + 1) \bmod (p-1)$:

Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(mrx - 1) \bmod (p-1)$
Verifying the signature:
Check whether $y^{m.r} \bmod p = \alpha r^s \bmod p$

12) $sx = (rmk + 1) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(rmk + 1) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = \alpha r^{r.m} \bmod p$

13) $x(r+m) = (k + s) \bmod (p-1)$: (Harn Scheme 2) [7]
Signing the message m:
$r = \alpha^k \bmod p$, $s = (x(m + r) - k) \bmod (p-1)$
Verifying the signature:
Check whether $y^{m+r} \bmod p = r\alpha^s \bmod p$

TABLE II.
OPERATION NUMBERS OF DIGITAL SIGNATURE SCHEMES

Scenarios	*Signing or Verification*	*Num of add. & sub.*	*Num. of mult.*	*Num. of divi.*	*Num. of Inv.*	*Num. of Exp.*
Scheme 1 -ElGamal	**Sig.**	1	2	2	1	1
	Ver.		1	1		3
Scheme 2	**Sig.**	1	2	2	1	1
	Ver.		1	1		3
Scheme 3	**Sig.**	1	2	2		1
	Ver.		1	1		3
Scheme 4 - Harn1	**Sig.**	1	2	2		1
	Ver.		1	1		3
Scheme 5 - AMV	**Sig.**	1	2	2	1	1
	Ver.		1	1		3
Scheme 6	**Sig.**	1	2	2	1	1
	Ver.		1	1		3
Scheme 7 - Optimal	**Sig.**	1	2	2		1
	Ver.		2	1		2
Scheme 8 -Yen-Laın	**Sig.**	1	2	2		1
	Ver.		1	1		2
Scheme 9	**Sig.**	1	2	2	1	1
	Ver.		2	1		2
Scheme 10	**Sig.**	1	2	2	1	1
	Ver.		2	1		2
Scheme 11	**Sig.**	1	3	3	1	1
	Ver.		2	1		2
Scheme 12	**Sig.**	1	3	3	1	1
	Ver.		2	1		2
Scheme 13 - Harn 2	**Sig.**	2	1	1		1
	Ver.	1	1	1		2
Scheme 14	**Sig.**	2	1	1	1	1
	Ver.	1	1	1		2
Scheme 15	**Sig.**	2	1	1		1
	Ver.	1	1	1		2
Scheme 16	**Sig.**	2	1	1	1	1
	Ver.	1	1	1		2
Scheme 17	**Sig.**	2	2	2	1	1
	Ver.	1	1	1		2
Scheme 18	**Sig.**	2	2	2	1	1
	Ver.	1	1	1		2

14) $sx = (k + (m + r)) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(k + (m + r)) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = r\alpha^{m+r} \bmod p$

15) $x = (k(m + r) + s) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = (x - k(m + r)) \bmod (p-1)$
Verifying the signature:
Check whether $y \bmod p = r^{r+m}\alpha^s \bmod p$

16) $x = (sk + (r + m)) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(x - (m + r)) \bmod (p-1)$
Verifying the signature:
Check whether $y \bmod p = \alpha^{r+m}r^s \bmod p$

17) $x(m + r) = (ks + 1) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = k^{-1}(x(m + r) - 1) \bmod (p-1)$
Verifying the signature:
Check whether $y^{r+m} \bmod p = \alpha r^s \bmod p$

18) $sx = (k(m + r) + 1) \bmod (p-1)$:
Signing the message m:
$r = \alpha^k \bmod p$, $s = x^{-1}(k(m + r) + 1) \bmod (p-1)$
Verifying the signature:
Check whether $y^s \bmod p = \alpha r^{m+r} \bmod p$

The numbers of fundamental arithmetic operations for each scheme are calculated to make a comparison in terms of time duration and shown in table II.

If N is the length of binary strings that are multiplied or divided, complexity of this operation is given in big-O notation as $O(N^2)$ [8].

The time duration of addition or subtraction operation is much smaller than that of multiplication or division; hence, addition is negligible with respect to multiplication. On the other hand, division takes approximately the same time with multiplication, whereas inverse and exponentiation operations are more complex. Multiplication operation is a basic block in inverse and exponentiation operations. In the next section, time durations of inverse and exponentiation operations are expressed in terms of time durations of multiplication and division operations.

4. COMPLEXITIES OF EXPONENTIATION AND INVERSE ALGORITHMS

The worst case analysis will be employed. We assume that all numbers which are going to be operated, are of bit length N, where N is smaller than or equal to the bit length of prime number p.

All the algorithms are taken from Knuth [9].

The Exponentiation Algorithm: It is also known as the binary left to right exponentiation and computes M^e=C mod n, as follows:

E1. Let $e = e_{N-1}e_{N-2}.....e_1e_0$.

E2. Set the variable C to 1.

E3. Repeat steps E3a and E3b for i = N-1, N-2,.....1, 0:

E3a. Set $C \equiv C^2$ mod n

E3b. If e_i=1, then set $C \equiv (C\ M)$ mod n

E4. Terminate the algorithm. C is the e^{th} power of M modulo n.

It can be seen that the number of times that the program runs into the E3 loop equals to the number of digits (N) of the number e. If e_i is 1 then there are 2 multiplications and 2 divisions; if e_i is 0 then there are 1 multiplication and 1 division. So any time in the loop, there are at most 2 multiplication operations and 2 division operations. In the worst case in which all e_i are 1, there are 2N multiplication and 2N division operations totally. Let T_{exp}, T_{mul}, T_{div} be the time duration of exponentiation, multiplication, division operation respectively, then maximum value of T_{exp} is:

$$T_{exp} = 2N(T_{mul} + T_{div}) \quad (7)$$

Also T_{div} equals approximately to T_{mul}, so

$$T_{exp} = 4NT_{mul} \quad (8)$$

Since the complexity of multiplication operation is given as $O(N^2)$, the complexity of exponentiation is given as $O(N^3)$ by (8).

The Inverse Algorithm: Given two nonnegative integers u and v, this algorithm determines a vector (u_1, u_2, u_3) such that $uu_1 + vu_2 = u_3$ and u_3 is the gcd (u, v). The computation makes use of auxiliary vectors (v_1, v_2, v_3) and (t_1, t_2, t_3). All vectors are manipulated in such a way that the relations $ut_1 + vt_2 = t_3$, $uu_1 + vu_2 = u_3$ and $uv_1 + vv_2 = v_3$ hold throughout the calculations. If this algorithm is used to find the inverse of any number a $\in$ GF(p), the inputs of the algorithm are u=p, v=a. Since a and p are relatively prime, the algorithm finds gcd (a, p) = 1; and the output $u_2 = a^{-1}$ is the desired inverse.

I1. *[Initialize.]* u_1=1, u_2=0, u_3=u ; v_1=0, v_2=1, v_3=v.

I2. *[v_3 = 0 ?]* If v_3 = 0 the algorithm terminates.

I3. *[Divide, subtract.]*

Set $q = \lfloor u_3 / v_3 \rfloor$ (9)

$t_1 = u_1 - qv_1$ (10)

$t_2 = u_2 - qv_2$ (11)

$t_3 = u_3 - qv_3$ (12)

$u_1=v_1$, $u_2=v_2$, $u_3=v_3$; (13)

$v_1=t_1$, $v_2=t_2$, $v_3=t_3$ (14)

I4. Go to step I2.

Notice that if equation (9) is substituted into equation (12), one obtains

$$t_3 = u_3 \bmod v_3 \quad (15)$$

Since we are interested in worst case analysis, we try to find the maximum number of times that (I2, I3, I4) loop is repeated. We concentrate on values of v_3, since termination condition is v_3=0. From step I3 of the algorithm, a relation between previous and current values of u_3 and v_3 can be given using equation (13) as:

$$u_k = v_{k-1} \quad (16)$$

and using equations (14) and (15) as:

$$v_k = u_{k-1} - \lfloor u_{k-1} / v_{k-1} \rfloor . v_{k-1} = u_{k-1} \bmod v_{k-1} \quad (17)$$

From (17) one can see that v_k=0 whenever v_{k-1}=1.

Substituting (16) into (17) one obtains:

$$v_k = v_{k-2} \bmod v_{k-1} \quad (18)$$

All integers $v_{k-2} \geq 2$ satisfy (18) with v_k=0 and v_{k-1}=1. To find the maximum value of iteration numbers corresponding to worst case, we choose the smallest v_{k-2}=2. Substituting v_k=1 and v_{k-1}=2 into (18), we evaluate the smallest value of v_{k-2} as 3. Continuing this procedure, the set (v_k, v_{k-1}, v_{k-2}) corresponding to worst case is found as (0, 1, 2), (1, 2, 3), (2, 3, 5), and (3, 5, 8) successively. Hence the difference equation for the worst case is found as:

$$v_{k-2} = v_k + v_{k-1} \quad (19)$$

If this linear difference equation is solved with v_k=1 and v_{k-1}=2, one can find a relation between the maximum number of iterations corresponding to the bit length of v_{k-2}. It is found that bit length of v_{k-2} is increased by 2 bits for approximately 3 iteration steps. Hence in order to find the inverse of an N bit number a, the algorithm requires 3N/2 iteration steps at most.

In the loop, there are 1 division operation and 3 multiplication operations. Since inverse algorithm requires at most 3N/2 steps for finding the inverse of an N bit number ;

$$T_{inv} = 3N/2\ (3T_{mul} + T_{div}) = 6NT_{mul} \quad (20)$$

Hence complexity of the inverse algorithm is also given as $O(N^3)$ by (20).

Let us summarize the time complexities of the multi-precision algorithms used in digital dignature signing and verification processes:

Addition & Subtraction : O(N)

Multiplication & Division: $O(N^2)$

Modular Inverse : $O(N^3)$

Modular Exponentiation : $O(N^3)$

5. CONCLUSION

The most time-consuming operation is the exponentiation algorithm. So efficient schemes must have as small number of exponentiation operations as possible. From table II, all schemes have 1 exponentiation operation in signing procedure. However for verification procedure, some schemes have 2, and others, which can be omitted, have 3 exponentiation operations.

Among the algorithms which use two exponentiations, the second time-consuming operation is the inverse algorithm. Some schemes have one inverse operation, others do not have any (as in schemes 7, 8, 13, and 15). We could ignore the schemes with one inverse operation; however, if the algorithm finds the inverse of the private key (as in schemes 9, 12, 14, and 18), inverse operation is not time-consuming since the inverse of the private key can be precalculated and stored.

From the above results, in addition to scheme 7 which is announced as optimal by Nyberg and Rueppel [5], the other schemes 8, 9, 12, 13, 14, 15 and 18 also seem to be more efficient than the others.

REFERENCES

[1] ElGamal, T. "A Public-Key Cryptosystem Based On Discrete Logarithms", *IEEE Trans. Inform. Theory IT-31, No.4,* July 1985, 469-472.

[2] Harn, L., Xu, Y. "Design Of Generalised ElGamal Type Digital Signature Schemes Based On Discrete Logarithm", *Electronic Letters,* 1994, 2025.

[3] Harn, L. "Group-Oriented (t,n) threshold signature and multisignature", *IEE Proc. E,* 1994.

[4] Agnew, G.B., Mullin, R.C., Vanstone, S.A., "Improved digital signature scheme based on discrete exponentiation", *Electronic Letters,* 1990 ,26 ,(14) , 2025.

[5] Nyberg, K., Rueppel, R.A., "Message recovery for signature schemes based on the discrete logarithm problem", Pre-proc. of Eurocryp '94, May 1994, 175-190

[6] Yen, S. –M., Lain, C., -S. "New Digital Signature Scheme Based On Discrete Logarithm", *Electronic Letters,* 1993, 1120.

[7] Harn, L. "New Digital Signature Scheme Based On Discrete Logarithm", *Electronic Letters,* 1994, 396.

[8] Harel, D. *ALGORITHMICS; The Spirit Of Computing,* Addison-Wesley, 1987.

[9] Knuth, D. E. *The Art Of Computer Programming, Vol 2: Seminumerical Algorithms,* Addison-Wesley, 1969.

[10] Diffie, W., Hellman. M. "New Directions in Cryptography". *IEEE Trans. Inform. Theory IT-22, No.6,* November 1976, 644-654.

[11] Kernighan, Brain W. *The C (ANSI C) Programming Language,* Prentice Hall Software Series, 1988.

[12] Schneier, B. *Applied Crytography,* Wiley, 1994.

[13] Stallings, William *Data and Computer Communications,* Prentice-Hall International Editions, Fourth Edition, 1994.

[14] Pfleger, Charles P. *Security in Computing,* Prentice-Hall International Editions,1989.