ISSN:2231-0711

A New Set of Bilinear Pairings and Their Applications in Diffie-Hellman Type Key Exchange and Threshold Cryptography

Shobana T.S , Bhaskara Rao N

Dayananda Sagar College of Engineering
Bangalore, India

Abstract— **This paper presents a new set of bilinear pairings which provide Diffie-Hellman type key exchange and are also used to generate secret key shares for a (t, n) threshold cryptography. The threshold scheme uses Lagrange interpolation formula.**
Keywords— **Bilinear pairings, Diffie-Hellman type key exchange, Lagrange interpolation formula, threshold cryptography**.

I. INTRODUCTION

Bilinear pairings are used in several ways in cryptography [1], [2], [3]. Bilinear pairings can be formed in different ways [4]. We are presenting new set of bilinear pairings over Z_p and a few of its applications in Cryptography.

A. *Basic Definition*

Let Z_p be the modulo p integer set for a large prime number p. That is,

$$Z_p = \{ 0, 1, 2, \ldots, p-2, p-1\} \qquad (1)$$

Z_p is referred as the **set of residues** [5]**.** It is a finite field of order p.

We define the new bilinear pairing over Z_p as,

$$e(U, V) = 2^{U} \bmod p \qquad (2)$$

Here, U and V are the given integers in Z_p. That is, $U \in Z_p$ and $V \in Z_p$. Expression 2^{U} is calculated using the modular arithmetic such that 2^{U} also belongs to Z_p. Thus, $e(U, \in Z_p$. In Eq. (2), integer 2 is raised to the power (U*V) using mod p modular arithmetic.

In our method, $2^{U} \bmod p$ is calculated as,

$2^{U} \bmod p = (\bmod p)^V \bmod p$. Therefore the definition of e(U,V) becomes,

$$e(U,V) = (\bmod p)^V \bmod p \qquad (3)$$

We use left-to-right binary method for modular exponentiation in our calculation of e(U,V)..

B. *Properties of e(U,V)*

In the following expressions, U, V, a, b and R belong to Z_p.

[1] e(U, V) is commutative. That is,
e(U,V) = e(V, U).

This follows from the definition (2).

[2] $e(a*U, V) = e(U, a*V) = e(U, V)^a$ (4)

This follows from the property of indices as,

$$e(a*U, V) = 2^{a*U*V} = (2^{U*V})^a = e(U, V)^a$$

Similarly, it can be shown that,

$e(U, a*V) = e(U, V)^a$

[3] $e(a*U, b*V) = e(b*U, a*V) =$

$= e(U, V)^{a*b} = e(U, V)^{b*a}$

$= [e(U, V)^a]^b = [e(U, V)^b]^a$ (5)

This can be proved as follows.

By definition,

$$e(a*U, b*V) = 2^{(a*U)*(b*V)} = 2^{(b*U)*(a*V)}$$

By the theory of indices,

$$2^{(a*U)*(b*V)} = e(U, V)^{a*b} = [e(U, V)^a]^b =$$
$$= e(U, V)^{b*a} = [(2^{U*V})^b]^a$$

Therefore, Eq. (5) is proved.

$e(U, V)^{a*b}$ is evaluated as,

$$e(U, V)^{a*b} = (e(U, V)^a \bmod p)^b \bmod p$$

[4] $e(U+R, V) = e(U, V)*e(R, V)$ (6)

This follows from the theory of indices as,

$$2^{(U+R)*V} = 2^{U*V+R*V} = (2^{U*V}) * (2^{R*V})$$

II. DIFFIE-HELLMAN TYPE KEY EXCHANGE

Our new bilinear pairings are used for Diffie-Hellman (DH) type key exchange. The public keys of user A and user B are chosen as,

$$K_A = e(a, U) \qquad (7)$$
$$K_B = e(b, V) \qquad (8)$$

The private keys of A and B are,

$$R_A = \{ a , U\} \qquad (9)$$
$$R_B = \{ b , V\} \qquad (10)$$

A sends K_A to B through an unsecured channel and similarly, B sends K_B to A. The arrangement is shown in Fig.1. After receiving K_B from B, user A calculates the common key K_{BA} as,

$$K_{BA} = [(K_B)^a]^U \qquad (11)$$

Here, A uses his private keys m and U to get K_{BA}.

Similarly, user B calculates K_{AB} after receiving K_A a,

$$K_{AB} = [(K_A)^b]^V \qquad (12)$$

Substituting for K_B and K_A from Eqs. (8) and (7), in Eqs. (11) and (12), we get,

$$K_{BA} = [e(b, V)^a]^U \qquad (13)$$

$K_{AB} = [e(a, U)^b]^V$ (14)

From Eq.(5), we see that the RHS's of the above equations are equal. Therefore,

$K_{BA} = K_{AB}$ (15)

Hence common secret key is available for both A and B.

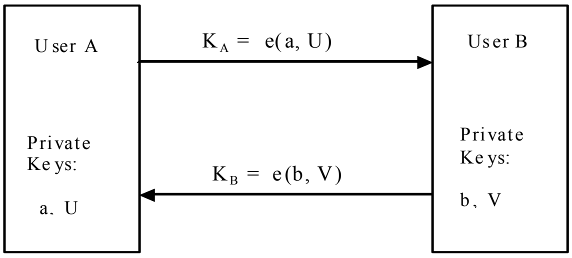

Fig. 1. Diffi-Hellman Type Key Exchange

A simple example is given to demonstrate the DH key exchange.

Example 1. The following parameters are chosen for this example.

Prime number p = 30577.

Private Keys of A: m = 1939, U = 2313.

Private Keys of B: n = 1799, V = 3111.

From Eqs. (7) and (8) K_A and K_B are found to be,

$K_A = 21771$ and $K_B = 5553$.

The intermediate values $(K_B)^a$ and $(K_A)^b$ are found to be,

$(K_B)^a = 3036$ and $(K_A)^b = 23397$

The final values K_{BA} and K_{AB} are,

$K_{BA} = K_{AB} = 9150$

III. Threshold Cryptography Using Bilinear Pairings

Bilinear pairings are well suited for threshold cryptography [6], [7]. In the (t, n) threshold scheme, a secret key is encoded in n shares which are then distributed to the corresponding n users. Any t (or more) shares out of n can be used to decode the secret. In our paper, the (t, n) encryption scheme is implemented using the Lagrange interpolation formula and the bilinear pairings. The arrangement is shown in Fig.2.

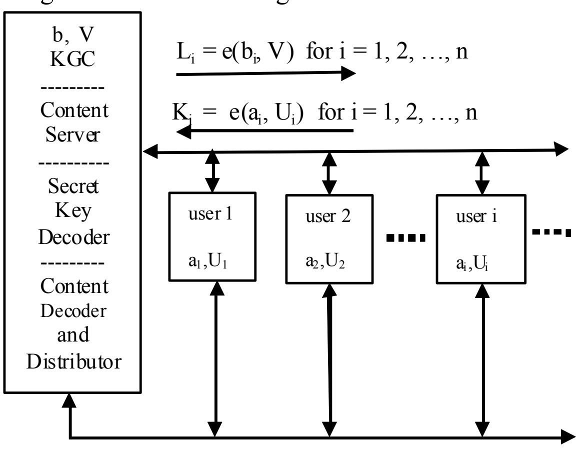

Fig.2. (t, n) threshold cryptography scheme

There are n users designated by 1,2,…,n. They can access the content server which also houses the Key Generation Centre (KGC) and the secret key decoder as shown in Fig.2. The working of the scheme is described as follows.

A. *Setup Phase*

User i selects (randomly) his/her two private keys a_i and U_i belonging to Z_p for i = 1, 2, …, n. The KGC selects its private key $V \in Z_p$ randomly. It also randomly chooses the coefficients b, $d_1, d_2, \ldots d_{t-1} \in Z_p$ and forms the polynomial over Z_p as,

$$F(x) = b + d_1*x + d_2*x^2 + \ldots + d_{t-1}*x^{t-1} \quad (16)$$

The degree of the polynomial is (t-1). The additional private keys of KGC are calculated as,

$$b = F(0); \quad (17)$$

$$b_1=F(1), b_2=F(2), \ldots, b_i=F(i), \text{for } i = 1,2,\ldots,n. \quad (18)$$

B. *Public keys generation and distribution*

User i generates his/her public key K_i as,

$$K_i = e(a_i, U_i) \text{ for } i=1,2,\ldots,n \quad (19)$$

and sends it to KGC. In turn, KGC generates the public keys

$$L_i = e(b_i, V) \text{ for } 1 \le i \le n \quad (20)$$

and sends them to the respective users. Therefore user i receives $L_i = e(b_i, V)$.

C. *Identification signatures by users*

User i generates his/her identification signature as,

$$G_i = L_i^{a_i*U_i} = e(b_i, V)^{a_i*U_i} \text{ for } 1 \le i \le n \quad (21)$$

D. *Verification signatures by KGC*

The KGC generates verification signatures from K_i's it has received from users as,

$$H_i = K_i^{b_i*V} = e(a_i, U_i)^{b_i*V} \text{ for } 1 \le i \le n \quad (22)$$

These H_i's are passed on to the secret key decoder unit of the content server for validating the users.

E. *Secret key of KGC*

The KGC generates its secret key K as,

$$K = e(q*b, V) \quad (23)$$

Here, q is a scale factor given by,

$$q = (n-1)\,! \quad (24)$$

q is the factorial(n-1). This scale factor is used to take care of fractional multipliers which may occur while calculating the coefficients in Lagrange formula which will be described later.

The content server uses this key K to encrypt its contents.

F. *Secret key decoding*

Let us represent all the users in the scheme by the set W as,W = {1, 2, …, n}. Here, user i is represented by the number i for $1 \le i \le$. n. Let the given t number of users involved in the current threshold decoding be represented by Y. Here Y is a subset of W For example if n =4 and t =3, the entire user set is W = {1, 2, 3, 4}. Then the threshold set Y could be one of the following sets. {1, 2, 3}, {1, 2 ,4}, {1, 3, 4) or {2, 3, 4}. In a given situation, let Y be a specific subset of t elements out of W. Then from the Lagrange interpolation formula [8],

$$b = F(0) = \sum_{i \in Y} b_i * c_i \quad (25)$$

where,

$$c_i = \prod_{j \in Y, j \neq i} \frac{j}{j-i} \quad \text{for } i \in Y \qquad (26)$$

for $1 \leq i \leq n$.
c_i's are called the Lagrange Coefficients.

Because of the term $\frac{j}{j-i}$ in Eq. (26), c_i can be a fraction instead of an integer. But in our cryptographic scheme, all variables should be integers and should belong to Z_p. Therefore, c_i's are scaled up by the factor q such that $q*c_i$'s become integers for all i's. The value of q is fixed as follows.
In Eq. (26), the denominator term is (j-i). The maximum value of (j-i) occurs when j = n (maximum of j) and i = 1 (minimum of i). Then max(j-i) is (n-1). The minimum of abs(j-i) cannot go to zero because of the condition $j \neq i$ in Eq. (26). Therefore the minimum of abs(j-i) is 1. Therefore the denominator term of Eq. (26) can take values in the range 1 to (n-1) depending on Y. Hence (n-1) ! is perfectly divisible by (j-i) for all possible combinations of j and i. Hence the scale up factor is,

$$q = (n-1)! = \text{factorial}(n-1) \qquad (27)$$

Therefore $q*c_i$ will be an integer for $1 \leq i \leq n$.
Now, multiplying both sides of Eq. (25) by q gives,

$$q * b = \sum_{i \in Y} b_i * q * c_i \qquad (28)$$

The secret key decoder knows set Y and it calculates c_i for $i \in Y$ according to Eq. (26). It has also received L_i's from KGC for $1 \leq i \leq N$. Then it will calculate R_i as,

$$R_i = (L_i)^{q*c_i} = e(b_i, V)^{q*c_i} \quad \text{for } i \in Y \qquad (29)$$

From the property of the bilinear pairings, $e(b_i, V)^{q*c_i}$ can be expressed as,

$$e(b_i, V)^{q*c_i} = e(1, V)^{b_i*q*c_i} \qquad (30)$$

In the light of Eq.(30), Eq. (29) can be rewritten as,

$$R_i = e(1, V)^{b_i*q*c_i} \qquad (31)$$

Now, consider the product,

$$S = \prod_{i \in Y} R_i \qquad (32)$$

$$\text{That is, } S = \prod_{i \in Y} R_i = \prod_{i \in Y} e(1, V)^{b_i*q*c_i} \qquad (33)$$

Using the identity, $x^m.x^n = x^{(m+n)}$, Eq. (33) can be expressed as,

$$S = e(1, V)^{\sum b_i*q*c_i} \quad \text{summation over } i \in Y \qquad (34)$$

But from Eq. (28), $\sum b_i * q * c_i$ over $i \in Y$ is q*b. Hence Eq. (34) can be expressed as,

$$S = e(1, V)^{q*b} \qquad (35)$$

Then from the property of the bilinear pairings,

$$S = e(q*b, V) \qquad (36)$$

From Eqs. (23) and (36), we see that the key recovered by Eq. (36) is same as given by Eq.(23). Hence K can be extracted by calculating S at the key decoding centre as follows.

G. *Signature verification and secret key extraction*

User i submits his L_i along with G_i to the secret key decoder. The key decoder verifies the validity of the user by comparing G_i with H_i. If they are not equal the submission L_i from user i is rejected by the decoder and there is no content decryption. If G_i and H_i are found equal, the decoder accepts L_i and further processing takes place.

The key decoder accepts L_i's submitted by users for $i \in Y$. Here Y is a specific threshold combination of t users. The decoder calculates R_i's for $i \in Y$ using Eq. (29).

Then using Eq. (32), it calculates S which is same as K. The content decoder uses S to decrypt the encrypted data from the content server. The decrypted data can be viewed by the interested party at the content decoder's location

IV. TEST RESULTS

Public and private key calculations and the secret key extraction using the threshold decryption is simulated using matlab. Example 2 demonstrates the resulting numerical values.

Example 2. The following parameters are used in this example.
Number of users, n = 4. For Z_p, p=30577.
Threshold level t =3.
Private key parameters a_i and U_i of the users in the order are,
a_i = [179 1993 2163 291] for i = 1 to 4
U_i = [235 111 173 2537] for i = 1 to 4.
The Lagrange polynomial is chosen as,
$F(x) = 193+111*x+171*x^2$
Private keys of KGC are, V = 13113 and b = 193. The values of b_i's are,
b_i = [475 1099 2065 3373] for i = 1to 4.

The public keys generated by the users and sent to KGC are,
K_i = [24531 23680 20447 9420] for i = 1to 4.

The public keys generated by KGC and sent to the users are,
L_i = [3269 15713 17050 16505] for i = 1to 4.

User generated identification signatures are,
G_i = [12111 2954 22014 16508] for i = 1to 4.

KGC generated H_i's are found to be same as G_i's.
The threshold set Y = [1 2 4]. The corresponding Lagrange coefficients are,
[c_1 c_2 c_4] = [8/3 –2 1/3]. Here q=factorial(n-1) = 6.

Therefore, q*[c_1 c_2 c_4] = [16 –12 2].
The values of R_i's are,
[R_1 R_2 R_4] = [194 14252 4532].

The product term S =R1*R2*R4 is found to be 17816. This is same as the secret key generated by KGC and used by the content decrypter. That is

$$K = e(q*b, V) = e(6*193, 13113) = 17816.$$

V. CONCLUSION

A new set of bilinear pairings are defined. A new method is proposed for (t, n) threshold cryptography using these bilinear pairings. Even though the key generation process and the bilinear calculations are computationally expensive, it is easy to implement them using the modern high speed machines.

The decoding and recovery of the secret can take place at different distributed locations. The length of the secret key, given by k can be easily scaled up for improved security.

REFERENCES

[1] An effiecint signature scheme from bilinear pairing and its applications,F Zhang, R Safavi-Naini, W Susilo 2004 – Springer.

[2] Boneh and M. Franklin, Identity-based encryption from the Weil pairing, Ad-vances in Cryptology-Crypto 2001, LNCS 2139, pp.213-229, Springer-Verlag, 2001.

[3] Efficient ID-based blind signature and proxy signature from bilinear Pairings. F Zhang, K Kim - Information Security and Privacy, 2003 – Springer

[4] A Promenade through the New Cryptography of Bilinear Pairings robotics.stanford.edu

[5] William Stallings. Cryptography and Network security, principles and Practices. Fourth Edition. Pearson Education. 2006. Page 105.

[6] Wei Gao et al. “Efficient identity-based threshold decryption scheme from bilinear pairings”.

[7] User-Oriented Key Management Scheme for Content protection in OPMD environment.