Diffie-Hellman Type Key Exchange, ElGamal Like Encryption/Decryption and Proxy Re-encryption Using Circulant Matrices

Chitra Rajarama[1], Jagadeesha Narasimhamurthy Sugatoor[2], and T. Yerri Swamy[3]
(Corresponding author: Chitra Rajarama)
Information Science, Engineering, NIE Institute of Technology[1]
Mysuru-570018, Karnataka, India
Electronics, Communications Engineering, PES Institute of Technology, Management[2]
Shimoga-577204, Karnataka, India
Computer Science, Engineering, KLE Institute of Technology[3]
Hubli - 580030, Karnataka, India
(Email: chitramanuel@yahoo.co.in)
(Received Mar. 22, 2017; revised and accepted Nov. 11, 2017)

Abstract

New methods for Diffie-Hellman type key exchange, ElGamal like encryption decryption and proxy re-encryption, using circulant integer matrices as the private keys, are described. Arithmetic operations are carried out using modular arithmetic to provide secrecy as well as to limit the size of the elements of the key matrices. Here Bidirectional proxy re-encryption is realized using circulant matrices. In the proposed proxy re-encryption technique, we use only matrix multiplication and inversion. Here, the proxy re-encryptors can be easily cascaded. Our scheme is efficient and simple to implement.

Keywords: Circulant Integer Matrices; Diffie-Hellman Type Key Exchange; ElGamal-like Encryption Decryption; Proxy Re-encryption

1 Introduction

The most popular key exchange technique over an unsecure channel is Diffie-Hellman (DH) Key Agreement protocol [5, 11, 21]. In our scheme we use integer matrices as the parameters of the cryptosystem so that the effective size of the keys can be large with smaller sized integers as the elements of the key matrices. The elements of the matrices used belong to the finite field Z_p where in all the numbers are integers in the range 0 to (p–1) and all the arithmetic operations are carried out with respect to modulo p where p is a suitable prime number. In our scheme the private keys used are circulant matrices. ElGamal [6] encryption/decryption is a public key cryptosystem where the cipher text has two components. We use circulant matrices to realize the ElGamal scheme.

Proxy re-encryption [2, 3] basically delegates the decryption process to a third party by re-encrypting the ciphertext. Proxy re-encryption has become an important tool in digital rights management schemes in cloud computing. Our main contribution is the application of circulant matrices in a new way for bidirectional proxy re-encryption and decryption.It involves matrix multiplication and inversion in Z_p.

The paper is organized as follows. Section 2 contains brief information about previous work in this field. Section 3 gives preliminary symbols, notations and definitions. Section 4 describes the Diffie-Hellman key exchange using matrices. In Section 5, ElGamal encryption/decryption is given. Section 6 describes proxy re-encryption using matrices. In Section 7, we discuss matrix keys versus scalar keys in cryptography, vulnerabilities of circulant matrices and a brief comparison with other methods. Conclusion is presented in Section 8.

2 Previous Work

Hill Cipher [9] is the earliest work where a square nonsingular matrix is used as the symmetric key. Matrix multiplication is used for encryption and multiplication by the inverse of that matrix is used for decryption. All arithmetic operations are carried out in the finite field domain. Maximum Distance Separable matrices have been proposed for cryptography by a few authors [8,12]. Use of circulant matrices in cryptography is described and discussed in several research works [10, 15, 17, 18]. Products of commutative matrices as public keys are used for Diffie-Hellman type key exchange [19, 20]. Our work also uses product of matrices but in a different way as described

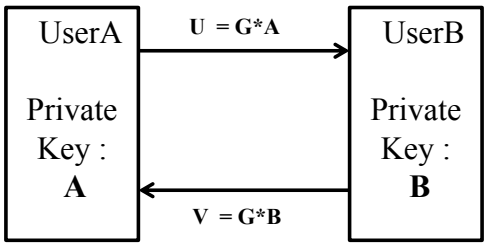

Figure 1: Diffie-Hellman type key exchange

later.

Several authors have based their proxy re-encryption algorithms on bilinear maps [1,2,14,22,23]. But we could not find any earlier work on the use of matrices for proxy re-encryption.

3 Symbols, Notations and Definitions

Consider a two user system as shown in Figure 1. The two users are designated as User A and User B. We assume a bidirectional communication link between User A and User B.

3.1 Circulant Matrix

A circulant matrix [4,13] is a square matrix where, given the first row, the successive rows are obtained by cyclically right shifting the present row by one element. Thus the i^{th} row of a circulant matrix of size (n x n) is obtained by cyclically right shifting the $(i-1)^{th}$ row by one position, for i = 2 to n, given the first row. Let the first row be the row vector, [$c(1)$, $c(2)$, ..., c(n–1), $c(n)$]. Then the circulant matrix **C** is obtained as

$$\mathbf{C} = \begin{bmatrix} c(1) & c(2) & \cdots & c(n) \\ c(n) & c(1) & \cdots & c(n-1) \\ \cdots & \cdots & \cdots & \cdots \\ c(2) & c(3) & \cdots & c(1) \end{bmatrix}$$

The elements of the first row are chosen such that gcd (elements of first row) = 1. This condition assures that the rank of the circulant matrix **C** is n. The most important property of circulant matrices is they are multiplicatively commutative. In our proposed method, we use circulant matrices which belong to the closed linear group GL(n, p) [3]. A Linear group GL(n, p) [16] represents non-singular matrices of size nxn over a finite field (Galois Field) GF(p) or Zp.

3.2 Members of The Cryptosystem

Private keys of User A and User B are **A** and **B** respectively which are circulant matrices of size (nxn). Matrices **A** and **B** belong to GL(n, p). The elements of the first rows of **A** and **B** are chosen so that the rank of both **A** and **B** is n. The generator matrix for this DH system is **G** which is a rectangular matrix of size (n–1) x n. The elements of **G** belongs to Zp. The elements of the generator matrix **G** are so chosen that the rank of **G** is (n–1). That is, rank(**G**) =(n–1). The public key of User A is denoted by matrix **U** and it is generated as

$$\mathbf{U} = \mathbf{G}*\mathbf{A} \tag{1}$$

The size of **U** is ((n–1) x n) x (n x n) = (n–1) x n. By knowing **U** and **G**, the private key **A** cannot be determined, because the left modular multiplicative inverse of **G** does not exist. In our scheme, **G** and **A** are so chosen that the rank of **U** = **G*****A** is (n–1). The public key of User B is denoted by matrix **V** and it is given by,

$$\mathbf{V} = \mathbf{G}*\mathbf{B} \tag{2}$$

The size of **V** is ((n–1) x n) x (n x n) = (n–1) x n. The matrix **B** is so chosen that the rank of **G*****B** is (n–1). Here also, **B** cannot be determined by knowing **V** and **G**. In our scheme, **U**, **V**, **G** and scalars n, p are in public domain while **A** and **B** are held private. All matrix multiplications are carried out in the finite field Zp.

4 Diffie-Hellman Type Key Exchange

User A sends matrix **U** to User B and User B sends matrix **V** to User A over the unsecured channel. User A calculates the common key $\mathbf{K_A}$ as

$$\mathbf{K_A} = \mathbf{V}*\mathbf{A} \tag{3}$$

The size of $\mathbf{K_A}$ is ((n–1) x n) x (n x n) =(n–1) x n. Similarly, User B calculates the common key $\mathbf{K_B}$ as

$$\mathbf{K_B} = \mathbf{U}*\mathbf{B} \tag{4}$$

The size of $\mathbf{K_B}$ is ((n–1) x n) x (n x n) = (n–1) x n. From Equations (2) and (3),

$$\mathbf{K_A} = \mathbf{G}*\mathbf{B}*\mathbf{A} \tag{5}$$

From Equations (1) and (4),

$$\mathbf{K_B} = \mathbf{G}*\mathbf{A}*\mathbf{B} \tag{6}$$

Since **A** and **B** are circulant matrices of size (n x n), they are multiplicatively commutative [13] as

$$\mathbf{G}*\mathbf{B}*\mathbf{A} = \mathbf{G}*\mathbf{A}*\mathbf{B} \tag{7}$$

From Equations (5), (6) and (7), the common keys of User A and User B are equal and the system common key **K** is

$$\mathbf{K} = \mathbf{K_A} = \mathbf{K_B} = \mathbf{G}*\mathbf{A}*\mathbf{B} = \mathbf{G}*\mathbf{B}*\mathbf{A} \tag{8}$$

The size of **K** is (n–1) x n. In Equations (3), (4), (5), (6) and (7), the results of the matrix multiplications are with respect to modulo p, even though the mod operation is not explicitly shown in those equations. Therefore, **K**, $\mathbf{K_A}$ and $\mathbf{K_B}$ also belong to Zp. Matrices **A**, **G** and **B** are so chosen that **K** given by Equation (8) has a rank of (n–1) so that its right modular inverse exists.

Example 1. *Let n = 4. The value of p is taken as 23. Matrices **G**, **A** and **B** are chosen as*

$$\boldsymbol{G} = \begin{bmatrix} 10 & 4 & 11 & 3 \\ 7 & 9 & 11 & 10 \\ 3 & 6 & 8 & 0 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 10 & 1 & 3 & 3 \\ 3 & 10 & 1 & 3 \\ 3 & 3 & 10 & 1 \\ 1 & 3 & 3 & 10 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 3 & 15 & 6 & 3 \\ 3 & 3 & 15 & 6 \\ 6 & 3 & 3 & 15 \\ 15 & 6 & 3 & 3 \end{bmatrix}$$

***U** and **V** are found to be*

$$\boldsymbol{U} = \begin{bmatrix} 10 & 0 & 15 & 14 \\ 2 & 22 & 9 & 21 \\ 3 & 18 & 3 & 2 \end{bmatrix} \quad \boldsymbol{V} = \begin{bmatrix} 15 & 6 & 1 & 21 \\ 11 & 18 & 10 & 17 \\ 6 & 18 & 17 & 4 \end{bmatrix}$$

*Without modulus operation, **V*A** and **U*B** are*

$$\boldsymbol{V*A} = \begin{bmatrix} 192 & 141 & 124 & 274 \\ 211 & 272 & 202 & 267 \\ 169 & 249 & 218 & 129 \end{bmatrix}$$

$$\boldsymbol{V*B} = \begin{bmatrix} 330 & 279 & 147 & 297 \\ 441 & 249 & 432 & 336 \\ 261 & 180 & 333 & 198 \end{bmatrix}$$

*With modulus p, mod(**V*A**, p) = mod(**U*B**, p) = $\boldsymbol{K_A}$ = $\boldsymbol{K_B}$ = **K** and with p=23, we get*

$$\boldsymbol{K} = \boldsymbol{K_A} = \boldsymbol{K_B} = \begin{bmatrix} 8 & 3 & 9 & 21 \\ 4 & 19 & 18 & 14 \\ 8 & 19 & 11 & 14 \end{bmatrix}$$

5 ElGamal Type Encryption and Decryption

ElGamal method [6] is an asymmetric key algorithm for public key cryptography. It is essentially based on Diffie-Hellman key exchange principle. ElGamal method that uses matrix keys is described in this section. All operations are with respect to mod p.

Let **M** be the message matrix whose elements are integers in the range 0 to (p–1). That is, the elements of **M** belong to Zp. The size of **M** is (n–1) x (n–1). User A encrypts **M** and sends it to User B. Matrices **A**, **B**, **G**, **U**, **V** and scalar p are same as described in Section 4. We assume that User A has already received **V** from User B. The encryption by User A is done by generating two crypto terms **U** (same as given by (1)) and **W** as

$$\begin{aligned} \mathbf{U} &= \mathbf{G*A} \\ \mathbf{W} &= \mathbf{M*V*A} \end{aligned} \tag{9}$$

From (2), (8) and (9),

$$\mathbf{W} = \mathbf{M*G*B*A} = \mathbf{M*G*A*B} \tag{10}$$

In the light of (8), Equation (10) can be rewritten as

$$\mathbf{W} = \mathbf{M*K_B} \tag{11}$$

The size of $\mathbf{K_B}$ is (n–1) x n. Since $\mathbf{K_B}$ is not a square matrix, it has no direct inverse. But, (11) can be solved for M using the pseudo inverse of $\mathbf{K_B}$ as

$$\mathbf{M} = \mathbf{W*(K_B)^\dagger} \tag{12}$$

Here, $\mathbf{(K_B)^\dagger}$ is the pseudo right modular inverse of $\mathbf{K_B}$ and is given by,

$$\mathbf{(K_B)^\dagger = K_B^T * (K_B * K_B^T)^{-1}} \tag{13}$$

We choose the crypto-parameters **G**, **A** and **B** such that $\mathbf{K_B}$ is a full rank matrix. Now, User A, the encrypter, sends the pair (**U**, **W**) to User B who is the intended decrypter.

5.1 Decryption at User B

On receiving (**U**, **W**), User B calculates $\mathbf{K_B}$ using (4). Then, he determines $\mathbf{(K_B)^\dagger}$ using (13) and consequently recovers **M** from (12). Results of all operations are calculated with respect to mod p.

5.2 Matrix Inverse in finite field Zp

Consider a square integer matrix **E** of size mxm and rank m, whose elements belong to Zp. We calculate the matrix **L**, the inverse of **E** with respect to mod p using the MatModInv(**E**, p) function [18] as

$$\mathbf{L} = \mathbf{MatModInv}(\mathbf{E}, p) \tag{14}$$

Here, **L** is found such that,

$$\mathbf{mod}(\mathbf{E*L}, p) = \mathbf{mod}(\mathbf{L*E}, p) = \mathbf{Im} = \mathbf{eye}(m) \tag{15}$$

where, $\mathbf{Im} = \mathbf{eye}(m)$ is the Identity Matrix of size mxm. Then **L** is the Matrix Inverse of **E** in the finite field Zp. The elements of **L** are integers in Zp.

Now, consider $\mathbf{W = M*K_B}$ as given by (11). The size of $\mathbf{K_B}$ is (n–1)xn. Since $\mathbf{K_B}$ is not a square matrix, it has no direct inverse. But,(11) is solved for M by post multiplying both sides of (11) by $\mathbf{(K_B)^T}$ which is the transpose of $\mathbf{K_B}$, to get

$$\mathbf{W * K_B^T = M * K_B * K_B^T} \tag{16}$$

The size of $(\mathbf{K_B} * \mathbf{K_B^T})$ is ((n–1)xn) x (nx(n–1)) which is equal to (n–1)x(n–1).

The elements $\mathbf{K_B}$ have to be so selected that the rank of $(\mathbf{K_B} * \mathbf{K_B^T})$ is (n–1) and that it has mod inverse. Then we have

$$(\mathbf{K_B} * \mathbf{K_B^T})^{-1} = \mathbf{MatModInv}((K_B * K_B^T), p) \quad (17)$$

Post multiplying (16) by $(\mathbf{K_B} * \mathbf{K_B^T})^{-1}$ gives

$$\mathbf{M} = \mathbf{W} * \mathbf{K_B^T} * (\mathbf{K_B} * \mathbf{K_B^T})^{-1} \quad (18)$$

The term $\mathbf{K_B^T} * (\mathbf{K_B} * \mathbf{K_B^T})^{-1}$ is the *right modular pseudo inverse* of $\mathbf{K_B}$ which is designated by $(\mathbf{K_B})^{\dagger}$ which has been specified in (13). From (13) and (18), Equation (12) follows. The size of $(\mathbf{K_B})^{\dagger}$ is nx(n–1).

Example 2. *The message matrix M, at User A, is taken as,*

$$M = \begin{bmatrix} 9 & 10 & 10 \\ 9 & 7 & 6 \\ 10 & 6 & 2 \end{bmatrix}$$

Other matrices and p are same as in Example 1. $\boldsymbol{W}$ *and* $(\mathbf{K_B} * \mathbf{K_B^T})$ *are calculated and found to be,*

$$\boldsymbol{W=M*V*A} = \begin{bmatrix} 8 & 16 & 3 & 9 \\ 10 & 21 & 20 & 3 \\ 5 & 21 & 13 & 0 \end{bmatrix}$$

$$\mathbf{K_B} * \mathbf{K_B^T} = \begin{bmatrix} 20 & 16 & 8 \\ 16 & 0 & 5 \\ 8 & 5 & 6 \end{bmatrix}$$

$(\mathbf{K_B} * \mathbf{K_B^T})^{-1}$ *and* $(\mathbf{K_B})^{\dagger}$ *as given by (13), are found to be,*

$$(\mathbf{K_B} * \mathbf{K_B^T})^{-1} = \begin{bmatrix} 7 & 12 & 19 \\ 12 & 11 & 17 \\ 19 & 17 & 22 \end{bmatrix}$$

$$(\mathbf{K_B})^{\dagger} = \begin{bmatrix} 3 & 0 & 5 \\ 12 & 16 & 16 \\ 5 & 10 & 6 \\ 6 & 0 & 2 \end{bmatrix}$$

All values are calculated with respect to mod p. Then, $\boldsymbol{M}$ *is recovered using (12) as* $\mathbf{M} = \mathbf{W} * (\mathbf{K_B})^{\dagger}$.

6 Proxy Re-encryption

Proxy re-encryption [4] is the process of re-encoding a given cipher text so that now, it can be decoded by another receiver other than the original one. The process is so designed that the re-encrypter itself cannot recover the plain text or it can not get hold of the private keys of the concerned parties.

Consider the model shown in Figure 2. Here, **A**, **B** and **C** are the private keys of User A, User B and User C respectively. Matrices **A**, **B**, **G**, **U**, **V**, $\mathbf{K_A}$, $\mathbf{K_B}$, **M** and **W** are same as described in Section 4. **C** is a circulant matrix of size nxn.

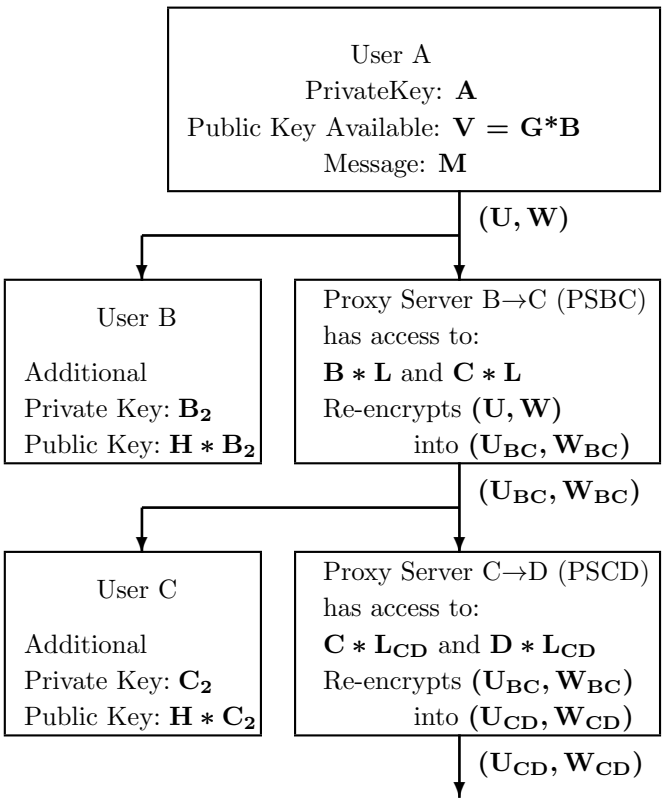

Figure 2: Proxy Re-encryption and decryption

6.1 Common Secret Key between User B and User C.

Here, the DH type common secret key between User B and User C is designed to have the size of nx(n+1) which is bigger compared to that of $\mathbf{K_A}$ or $\mathbf{K_B}$. The reason for using this bigger size is explained later in this section. This bigger common key is derived as follows.

We select one more public generator matrix **H**. The size of **H** is chosen as nx(n+1) and its elements belong to Zp. User B and User C select additional private circulant matrices (keys) $\mathbf{B_2}$ and $\mathbf{C_2}$ of size (n+1)x(n+1). The corresponding public keys are $\mathbf{H} * \mathbf{B_2}$ and $\mathbf{H} * \mathbf{C_2}$ respectively. By knowing **H** and $\mathbf{H} * \mathbf{B_2}$, we cannot determine $\mathbf{B_2}$ because **H** has no left inverse. Similarly, $\mathbf{C_2}$ cannot be determined by knowing **H** and $\mathbf{H*C_2}$.

On receiving $\mathbf{H} * \mathbf{B_2}$, User C calculates the secret common key between User C and User B as:

$$\mathbf{L_C} = (\mathbf{H} * \mathbf{B_2}) * \mathbf{C_2} \quad (19)$$

Similarly, User B calculates the secret common key between User B and User C as:

$$\mathbf{L_B} = (\mathbf{H} * \mathbf{C_2}) * \mathbf{B_2} \quad (20)$$

Since $\mathbf{B_2}$ and $\mathbf{C_2}$ are multiplicatively commutative $\mathbf{B_2} * \mathbf{C_2} = \mathbf{C_2} * \mathbf{B_2}$ and then, $\mathbf{L_B}$ and $\mathbf{L_C}$ are equal as:

$$\mathbf{L} = \mathbf{L_B} = \mathbf{L_C} = \mathbf{H} * \mathbf{C_2} * \mathbf{B_2} = \mathbf{H} * \mathbf{B_2} * \mathbf{CB_2} \quad (21)$$

The size of $\mathbf{L}$ is nx(n+1). The matrices $\mathbf{H}$, $\mathbf{B_2}$ and $\mathbf{C_2}$ are so chosen that the rank of $\mathbf{L}$ is n.

6.2 Encryption by User A

Encryption by User A is same as described in Section 4. User A generates the two matrices $\mathbf{U}$ and $\mathbf{W}$ (See (1) and (10)) as repeated here.

$$\begin{aligned} \mathbf{U} &= \mathbf{G} * \mathbf{A} \\ \mathbf{W} &= \mathbf{M} * \mathbf{V} * \mathbf{A} = \mathbf{M} * \mathbf{G} * \mathbf{B} * \mathbf{A} \\ &= \mathbf{M} * \mathbf{G} * \mathbf{A} * \mathbf{B}. \end{aligned}$$

User A sends the encrypted data $\mathbf{(U, W)}$ to User B and to the proxy server B→C. From (4) and (12),

$$\mathbf{M} = \mathbf{W} * (\mathbf{U} * \mathbf{B})^{\dagger} \tag{22}$$

User B can decrypt cipher data $\mathbf{(U, W)}$ as given by Equation (22).

6.3 Re-encryption at Proxy Server B→C

Now the Proxy Server B→C (PSBC) is requested by User A (or by User B) to send the same data $\mathbf{M}$ to User C with proper re-encryption so that User C can decode it correctly. Here, PSBC has to translate the cipher text meant for User B to a new format so that the translated cipher text can be decoded by User C. Basically the PSBC accepts $\mathbf{(U, W)}$ as the input and re-encrypts it to generate ($\mathbf{U_{BC}}$, $\mathbf{W_{BC}}$) which is sent to User C. The re-encrypted term $\mathbf{W_{BC}}$ is so constructed that it can be decoded by User C only. Also, PSBC itself should be incapable of recovering $\mathbf{M}$, $\mathbf{B}$ or $\mathbf{C}$. During initialization, User B sends $\mathbf{(B * L)}$ to PSBC and similarly User C sends $\mathbf{(C * L)}$ to PSBC. The size of $\mathbf{(B * L)}$ as well as $\mathbf{(C * L)}$ is nx(n+1).

6.4 Formulation of $\mathbf{U_{BC}}$ and $\mathbf{W_{BC}}$ at PSBC

For the purpose of re-encryption, PSBC formulates $\mathbf{U_{BC}}$ and $\mathbf{W_{BC}}$ from $\mathbf{U}$ and $\mathbf{W}$ as,

$$\mathbf{U_{BC}} = \mathbf{U} = \mathbf{G} * \mathbf{A} \tag{23}$$

$$\mathbf{W_{BC}} = \mathbf{W} * (\mathbf{C} * \mathbf{L}) * (\mathbf{B} * \mathbf{L})^{\dagger}. \tag{24}$$

Here, $\mathbf{(B * L)}^{\dagger}$ is the right modular inverse of $\mathbf{(B * L)}$. The size of $\mathbf{(B * L)}$ is nx(n+1). By definition,

$$(\mathbf{B} * \mathbf{L})^{\dagger} = (\mathbf{B} * \mathbf{L})^{\mathbf{T}} * ((\mathbf{B} * \mathbf{L}) * (\mathbf{B} * \mathbf{L})^{\mathbf{T}})^{-1} \tag{25}$$

Reducing the parantheses on the RHS of (25), we get,

$$(\mathbf{B} * \mathbf{L})^{\dagger} = \mathbf{L}^{\mathbf{T}} * \mathbf{B}^{\mathbf{T}} * (\mathbf{B} * \mathbf{L} * \mathbf{L}^{\mathbf{T}} * \mathbf{B}^{\mathbf{T}})^{-1} \tag{26}$$

Taking the inverse operator on the RHS of (26) inside the paranthesis we get,

$$(\mathbf{B} * \mathbf{L})^{\dagger} = \mathbf{L}^{\mathbf{T}} * \mathbf{B}^{\mathbf{T}} * (\mathbf{B}^{\mathbf{T}})^{-1} * (\mathbf{L} * \mathbf{L}^{\mathbf{T}})^{-1} * \mathbf{B}^{-1} \tag{27}$$

Now, the RHS of Equation (27) is simplified as,

$$(\mathbf{B} * \mathbf{L})^{\dagger} = \mathbf{L}^{\mathbf{T}} * (\mathbf{L} * \mathbf{L}^{\mathbf{T}})^{-1} * \mathbf{B}^{-1} \tag{28}$$

By definition,

$$\mathbf{L}^{\mathbf{T}} * (\mathbf{L} * \mathbf{L}^{\mathbf{T}})^{-1} = \mathbf{L}^{\dagger} \tag{29}$$

from (28) and (29),

$$(\mathbf{B} * \mathbf{L})^{\dagger} = \mathbf{L}^{\dagger} * \mathbf{B}^{-1} \tag{30}$$

substituting this in (24), we get,

$$\mathbf{W_{BC}} = \mathbf{W} * (\mathbf{C} * \mathbf{L}) * \mathbf{L}^{\dagger} * \mathbf{B}^{-1} \tag{31}$$

On cancelling $\mathbf{L} * \mathbf{L}^{\dagger}$ in (31), we have,

$$\mathbf{W_{BC}} = \mathbf{W} * \mathbf{C} * \mathbf{B}^{-1} \tag{32}$$

Since C and $\mathbf{B}^{-1}$ are circulant matrices, they are multiplicatively commutative. Therefore,

$$\mathbf{C} * \mathbf{B}^{-1} = \mathbf{B}^{-1} * \mathbf{C} \tag{33}$$

From (32) and (33),

$$\mathbf{W_{BC}} = \mathbf{W} * \mathbf{B}^{-1} * \mathbf{C} \tag{34}$$

On substituting for $\mathbf{W}$ from (10) in (34), we get,

$$\mathbf{W_{BC}} = (\mathbf{M} * \mathbf{G} * \mathbf{A} * \mathbf{B}) * \mathbf{B}^{-1} * \mathbf{C} \tag{35}$$

On cancelling $\mathbf{B} * \mathbf{B}^{-1}$ in (35) we have,

$$\mathbf{W_{BC}} = \mathbf{M} * \mathbf{G} * \mathbf{A} * \mathbf{C} \tag{36}$$

In this way the private key $\mathbf{B}$ of User B is eliminated from $\mathbf{W_{BC}}$ and the private key $\mathbf{C}$ is inserted in its place. Now $\mathbf{W_{BC}}$ is ready for decryption by User C. The Proxy Server PSBC sends($\mathbf{U_{BC}}$, $\mathbf{W_{BC}}$) pair to User C.

6.5 Decryption at User C

Once, User C receives $\mathbf{U_{BC}}$ and $\mathbf{W_{BC}}$, $\mathbf{M}$ is recovered from $\mathbf{W_{BC}}$, based on (36). From (36) and (23),

$$\mathbf{W_{BC}} = \mathbf{M} * (\mathbf{U_{BC}}) * \mathbf{C} \tag{37}$$

Therefore $\mathbf{M}$ is recovered by User C as,

$$\mathbf{M} = \mathbf{W_{BC}} * (\mathbf{U_{BC}} * \mathbf{C})^{\dagger} \tag{38}$$

Here, $\mathbf{(U_{BC} * C)}^{\dagger}$ is the right modular inverse of $\mathbf{(U_{BC} * C)}$. It can be seen that (43) is similar to (22). A numerical example of proxy re-encryption is given below.

Example 3. *The values of* ***G, A, B, U, V, M, W*** *and* p *are same as in Example 2. The values of* ***C, H, B2,*** *and* ***C2*** *are taken as,*

$$\boldsymbol{C} = \begin{bmatrix} 9 & 15 & 9 & 6 \\ 6 & 9 & 15 & 9 \\ 9 & 6 & 9 & 15 \\ 15 & 9 & 6 & 9 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} 2 & 9 & 7 & 8 & 18 \\ 15 & 9 & 14 & 11 & 19 \\ 2 & 16 & 5 & 4 & 9 \\ 14 & 7 & 22 & 1 & 15 \end{bmatrix}$$

$$\boldsymbol{B_2} = \begin{bmatrix} 13 & 21 & 3 & 5 & 17 \\ 17 & 13 & 21 & 3 & 5 \\ 5 & 17 & 13 & 21 & 3 \\ 3 & 5 & 17 & 13 & 21 \\ 21 & 3 & 5 & 17 & 13 \end{bmatrix}$$

$$\boldsymbol{C_2} = \begin{bmatrix} 9 & 12 & 8 & 1 & 2 \\ 2 & 9 & 12 & 8 & 1 \\ 1 & 2 & 9 & 12 & 8 \\ 8 & 1 & 2 & 9 & 12 \\ 12 & 8 & 1 & 2 & 9 \end{bmatrix}$$

$\boldsymbol{L}$ *is calculated from (21) as,*

$$\boldsymbol{L} = \begin{bmatrix} 4 & 21 & 4 & 9 & 4 \\ 14 & 20 & 9 & 17 & 7 \\ 1 & 20 & 9 & 15 & 4 \\ 9 & 22 & 1 & 10 & 7 \end{bmatrix}$$

$\mathbf{B} * \mathbf{L}$ *is found to be,*

$$\mathbf{B} * \mathbf{L} = \begin{bmatrix} 2 & 20 & 20 & 11 & 1 \\ 8 & 3 & 19 & 18 & 20 \\ 20 & 1 & 1 & 1 & 1 \\ 13 & 9 & 6 & 13 & 20 \end{bmatrix}$$

$(\mathbf{B} * \mathbf{L})^{\dagger}$ *and* $\mathbf{C} * \mathbf{L}$ *are found to be,*

$$(\mathbf{B} * \mathbf{L})^{\dagger} = \begin{bmatrix} 14 & 11 & 8 & 8 \\ 18 & 19 & 11 & 13 \\ 11 & 16 & 20 & 12 \\ 7 & 20 & 6 & 6 \\ 6 & 1 & 11 & 16 \end{bmatrix}$$

$$\mathbf{C} * \mathbf{L} = \begin{bmatrix} 10 & 19 & 5 & 2 & 12 \\ 16 & 22 & 19 & 16 & 3 \\ 11 & 14 & 2 & 8 & 12 \\ 20 & 8 & 20 & 8 & 3 \end{bmatrix}$$

$\mathbf{W_{BC}}$ *and* $(\mathbf{U_{BC}} * \mathbf{C})$ *are found to be,*

$$\mathbf{W_{BC}} = \begin{bmatrix} 7 & 18 & 9 & 18 \\ 14 & 10 & 3 & 5 \\ 7 & 22 & 13 & 22 \end{bmatrix}$$

$$\mathbf{U_{BC}} * \mathbf{C} = \begin{bmatrix} 21 & 21 & 10 & 20 \\ 17 & 11 & 3 & 5 \\ 20 & 11 & 5 & 11 \end{bmatrix}$$

Matrix $(\mathbf{U_{BC}} * \mathbf{C})^{\dagger}$ *is found to be,*

$$(\mathbf{U_{BC}} * \mathbf{C})^{\dagger} = \begin{bmatrix} 2 & 9 & 18 \\ 8 & 13 & 9 \\ 10 & 9 & 14 \\ 11 & 0 & 21 \end{bmatrix}$$

Finally, $\mathbf{W_{BC}} * (\mathbf{U_{BC}} * \mathbf{C})^{\dagger}$ *is calculated to get* $\boldsymbol{M}$ *as,*

$$\mathbf{W_{BC}} * (\mathbf{U_{BC}} * \mathbf{C})^{\dagger} = \mathbf{M} = \begin{bmatrix} 9 & 10 & 10 \\ 9 & 7 & 6 \\ 10 & 6 & 2 \end{bmatrix}$$

6.6 Re-encryption from User C to User D

The Proxy Server Now the Proxy Server C→D (PSCD) accepts ($\mathbf{U_{BC}}$, $\mathbf{W_{BC}}$) and generates ($\mathbf{U_{CD}}$, $\mathbf{W_{CD}}$) using the similar process as described in Section 6.4. The proxy server should have access to $\mathbf{(C * L_{CD})}$ and $\mathbf{(D * L_{CD})}$ from User C and User D respectively. ($\mathbf{L_{CD}}$ is the common key between User C and User D similar to as given in (21). The decryption at User D would be similar as described in Section 6.5. The above chaining action can be continued further.

7 Discussion

7.1 Matrix Keys Versus Scalar Keys in Cryptography

When scalar keys are used their lengths have to be relatively very large, like 1024 bits, 2048 bits etc.as in RSA. The exponentiation and related arithmetic operations, using these long keys, are more difficult to implement in the processors used in wireless sensor nodes. The processors within the sensor nodes here have limited register sizes and less memory compared to the conventional processors. Therefore, long scalar keys are not convenient for cryptography involving sensor nodes. When integer matrices are used as keys, the length of the individual elements can be kept as low as 8 bits. The large number of elements in a matrix key make hacking very difficult and this very large number will compensate the short length of the key elements. The effective key length of a matrix of size nxn is nxnxm where m is the length of the individual elements of the matrix in bits. The Arithmetic operations on these small sized integers of the matrix keys are easy to implement in the processors of the sensor nodes. Therefore, matrix keys are well suited for the cryptographic operations involving sensor nodes.

7.2 Vulnerabilities of Circulant Matrices in Cryptography

In a circulant matrix, only the first row is chosen independently. The remaining rows are obtained by the circular shift of the preceding rows. Thus a hacker has to break only one row of the circulant key matrix to discover the entire matrix. Thus the effective key length of a circulant matrix used as a key is, nxm where n is number of columns of the matrix and m is the length of the individual elements in bits. Therefore the size of the circulant matrix itself has to be large to provide a large effective

key length. One major advantage is, while storing a circulant matrix, it is enough if we store only the first row. Thus the memory space is saved.

7.3 Comparison with Existing Methods

The closest method to ours is by Keith R Slavin [20], US patent US 7346162 B2. In that work also, the author uses closed Linear Group of matrices **GL**(n, p)'s. But multiplicatively commutative matrices are obtained in a different way other than using the circulant matrices. Our method is quite different from that of [20]. Our scheme is substantially better because, while generating public keys or the shared secret key, we use only one matrix multiplication compared to two as in [20]. The method used in [2], overcomes the deficiency of Cayley-Purser Algorithm [7] that uses even values for n, the size of private keys. But in our case, whether n is even or odd, the security of the cryptosystem will not be compromised. In [21], each user has to store two private keys which are mutually commutative and the number of matrix multiplications is more compared to our method. As far as our knowledge, no earlier work was found on proxy re-encryption using commutative matrices.

8 Conclusions

A new method of Diffie-Hellman type key exchange using circulant matrices as private and public keys is presented. Matrices as keys provide a large number of smaller sized integer elements which are easy to manipulate than a few very large sized integers. Circulant matrices are also used as keys to provide multi-stage proxy re-encryption. Since we use modular multiplication of matrices rather than modular exponentiation, our method is faster and less complex.

References

[1] H. Abdalla, X. Hu, A. Wahaballa, *et al.*, "Integrating the functional encryption and proxy re-cryptography to secure DRM scheme," *International Journal of Network Security*, vol. 19, pp. 27–38, 2017.

[2] G. Ateniese, K. Fu, M. Green and S.Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC'06)*, vol. 9, no. 1, pp. 1–30, 2006.

[3] M. Blaze, G. Bleumer and Martin Strauss, "Divertible protocols and atomic proxy cryptography," *EUROCRYPT*, vol. 1403, no. 1, pp. 127–144, 1998.

[4] P. J. Davis, *Circulant Matrices (2ed)*, New York: Chelsea Publishing, 2012.

[5] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[6] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[7] S. Flannery and D. Flannery, *In code*, New York: Workman publishing, 2001.

[8] K. C. Gupta and I. G. Ray, "On constructions of involutory MDS matrices," in *International Conference on Cryptology in Africa (AFRICACRYPT'13)*, pp. 43–60, June 2013.

[9] L. S. Hill, "Cryptography in an algebraic alphabet," *The American Mathematical Monthly*, vol. 36, no. 6, pp. 306–312, 1929.

[10] S. Inam and R. Ali, "A new ElGamal-like cryptosystem based on matrices over groupring," *Neural Computing and Applications*, pp. 1–5, 2016.

[11] C. H. Ling, S. M. Chen, and M. S. Hwang, "Cryptanalysis of Tseng-Wu group key exchange protocol," *International Journal of Network Security*, vol. 18, no. 3, pp. 590-593, 2016.

[12] J. Nakahara Jr. and E. Abrahao, "A new involutory MDS matrix for the AES," *International Journal of Network Security*, vol. 9, no. 2, pp. 109–116, 2009.

[13] I. Kra and S. R. Simanca, "On circulant matrices, notes," *American Mathematical Society*, vol. 59, no. 3, pp. 368–377, 2012.

[14] C. Lan, H. Li, S. Yin and L. Teng, "A new security cloud storage data encryption scheme based on identity proxy re-encryption," *International Journal of Network Security*, vol. 19, no. 5, pp. 804–810, 2017.

[15] A. Mahalanobis, "The discrete logarithm problem in the group of non-singular circulant matrices," *Groups Complexity Cryptology*, vol. 2, pp. 83–89, 2010.

[16] J. Overbey, W. Traves and J. Wojdylo, "On the keyspace of the hill cipher," *Cryptologia*, vol. 29, pp. 59–72, 2005.

[17] A. V. Ramakrishna and T. V. N. Prasanna, "Symmetric circulant matrices and publickey cryptography," *International Journal of Contemporary Maths and Sciences*, vol. 8, no. 12, pp. 589–593, 2013.

[18] K. A. Reddy, B. Vishnuvardhan, Madhuviswanatham and A. V. N. Krishna, "A modified hill cipher based on circulant matrices," *Procedia Technology*, vol. 4, pp. 114–118, 2002.

[19] M. K. Singh, "Public key cryptography with matrices," *Groups Complexity Cryptology*, vol. 2, pp. 83–89, 2010.

[20] K. R. Slavin, *Public key cryptography using matrices*, US Patent 7346162 B2, 2008.

[21] W. Stallings, *Cryptography and Network Security (4ed)*, India: Pearson Education, 2011.

[22] Y. Wang, D. Yan, F. Li and H. Xiong, "A key-insulated proxy re-encryption scheme for data sharing in a cloud environment," *International Journal of Network Security*, vol. 19, no. 4, pp. 623–630, 2017.

[23] T. Yoshida and M. Shirase, "A digital content sharing model using proxy re-encryption without server access," in *IEEE International Conference*

on Consumer Electronics - Taiwan (ICCE-TW'17), pp. 243–244, 2017.

Biography

Chitra Rajarama received her B.E., in Computer Science and Engineering., and M.Tech., in Computer Science and Engineering, from Visvesvaraya Technological University, Belgaum, Karnataka, India. She is currently pursuing PhD under Visvesvaraya Technological University, Belgaum, Karnataka, India. Her area of interest is in the field of wireless networks. She has guided many undergraduate projects. She has attended many national/international conferences and published several papers in international journals. At present she is Associate Professor in the department of Information Science and Engineering, NIE Institute of Technology, (affiliated to Visvesvaraya Technological University) Mysuru, Karnataka, India.

S. N. Jagadeesha received his B.E.,in Electronics and Communication Engineering, from University B. D. T. College of Engineering., Davangere affiliated to Mysore University, Karnataka, India in 1979, M.E. from Indian Institute of Science (IISC), Bangalore, India specializing in Electrical Communication Engineering., in 1987 and Ph.D. in Electronics and Computer Engineering., from University of Roorkee, Roorkee, India in 1996. He is an IEEE member. His research interest includes Array Signal Processing, Wireless Sensor Networks and Mobile Communications. He has published and presented many papers on Adaptive Array Signal Processing and Direction-of-Arrival estimation. Currently he is professor and head in the department of Computer Science and Engineering, PES Institute of Technology and Management,. (Affiliated to Visvesvaraya Technological University), Shimoga, Karnataka, India.

T. Yerri Swamy received his B.E.,in Electronics and Comm-unication Engineering, from Gulbarga Univerisity,Gulbarga, Karnataka , India in 2000. MTech in Network and Internet Engineering, from Visve-svaraya Technological University, Belgaum, Karnataka. at J. N. N. College of Engineering, Shimoga, Karnataka in 2005. and PhD in the Faculty of Computer and Information Sciences from Visvesvaraya Technological niversity, Belgaum, Karnataka in the year 2013. He is an ISTE member. His research interest includes Antenna Array Signal Processing, Statistical Signal Processing, Detection and Estimation, Cognitive radio comm-unications ,LTE/MIMO. He has published and presented number of papers in national/international conferences and journals. Currently he is Professor and Head, in the department of Computer Science and Engineering, KLE Institute of Technology, (Affiliated to Visvesvaraya Technological University), Hubli, Karnataka, India.