Enhancing Channel Security By Error Detection And Correction During Quantum Key Distribution Using Hamming Code With Modified Elgamal

A.Beatrice Dorothy, Dr. S. Britto Ramesh Kumar

Abstract: In Cryptography, key distribution plays a vital role. The unique advantage of Quantum key distribution is that the users can identify the presence of an eavesdropper. During key sharing, error may occur due to eavesdropping. This results in the loss of security. To detect such errors many algorithms were proposed and used. Since key distribution is the major part of the encryption, it has to be shared without errors. Even though, the key is shared securely; if eavesdropping occurs, then alteration of the key happens which results in an error. To avoid this error, after key distribution using BB84 protocol, Hamming Code method is applied to detect and correct error in this paper. The result from the proposed HAmming Code with Modified ELgaMAl (HACMELMA) produces the shared secret key, which is used as a private key for modified Elgamal encryption. The proposed HAmming Code with Modified ELgaMAl (HACMELMA) detects and corrects an error which increases the security level. The method which is proposed is compared with existing Elgamal encryption and Modified RSA (MRSA).

Index Terms: Hamming Code, BB84 protocol, Elgamal, Modified Elgamal, Error detection.

1. INTRODUCTION

QUANTUM cryptography is well known for quantum key distribution. QKD offers a solution for key sharing issues in information security [4]. Quantum error correction (QEC) is used in quantum computing to protect quantum information from errors due to decoherence and other quantum noise. Quantum error correction is essential if one has to achieve the fault-tolerance. Quantum computation deals not only with noise on stored quantum information, but also with faulty quantum gates, faulty quantum preparation, and faulty measurements. Let us assume that noise errors are independent and occur with probability p; then it is most likely that the error is a single-bit error and the transmitted message is three ones. A double-bit error may occur and the transmitted message is equal to three zeros [5]. Peter Shor first discovered the method of quantum error correcting code. It stores the information of one qubit against an extremely entangled state of qubits. A quantum error correcting code protects quantum information against errors of a limited form. There are several methods proposed in the literature for quantum error detection and correction. But still finding the correct bit position of the error is difficult. The error-correcting codes like, Bit-Flip code, single flip code, the Shor code, Bosonic codes were already in the literature. All these methods attempted to detect and correct errors. The error occurrence depends on the possibility of the eavesdropping.

To detect the error and to correct it, this paper attempts the Hamming Code method. As in [11], the procedure followed is similar, for key sharing and to detect the eavesdropping. Once if the eavesdropping is detected, the error can be detected with the help of eavesdropping rate. This paper uses the BB84 protocol for key sharing. The public-key algorithm Elgamal is also taken into consideration for encryption and decryption. The novelty in this paper is the Shared Secret Key (SSK). SSK is retrieved after error detection and correction, which is taken as the private key for Modified Elgamal. So the Modified Elgamal already provides encryption by using the generator g and the ciphertext pairs. Thus this increases the level of security. Generally, the private key x is generated based on {p,g,y} in Elgamal algorithm. But here x is taken from the result of the Hamming Code. Finally, each bit of the plaintext is taken and encrypted using Modified Elgamal. The rest of the paper is organized as follows. In section 2, the works related to the detection of error during key sharing is discussed. The proposed methodology for error detection and correction in key sharing is shown in section 3. The encryption/decryption using the public-key algorithm with an example is conferred in section 4. The experimental results are shown in section 5. Section 6 is the conclusion.

- *A.Beatrice Dorothy is currently pursuing Ph.D inComputer Science in St. Joseph's College, Tiruchirappalli, Affiliated to Bharathidasan University, India. E-mail: adorothybrice@gmail.com*
- *Dr. S.Britto Ramesh Kumar is Assistant Professor in Dept. of Computer Science in St. Joseph's College, Tiruchirappalli, Affiliated to Bharathidasan University, India. E-mail: brittork@gmail.com*

brittork

2 RELATED WORK

In [6], Akihiro Yamamura and Hirokazu Ishizuka proposed practical methods for error detection and authentication in quantum key distribution. They also introduced several concepts about neighbourhood collision-free properties of Boolean functions, which are related to hash functions. The proposed methods based on neighbourhood collision-free functions and error correcting codes such as Reed-Solomon code. Finally, they examined widely used cryptographic hash functions SHA-1 and MD5 whether or not to satisfy the neighbourhood collision-free property by computation experiments. Xiangyu Wang, Yichen Zhang, Song Yu & Hong Guo [7], proposed an experimental demonstration of high-speed error correction with multi-edge type low-density parity-check (MET-LDPC) codes based on the graphics processing unit (GPU). The authors optimized the memory structure of the parity-check matrix. Also, the propagation decoding algorithm will reduce the computational complexity. In [8], Amr Goneid and et al. have enhanced the error correction phase so that the keys generated would be more efficient. They introduced an algorithm based on the use of a memory structure between rounds of the error correction phase. Hong Lai and et al. [9] have developed a matrix algorithm for high-capacity, simple and fast quantum cryptography. The scheme achieved secure private communication with fresh keys which are generated

from recursive Fibonacci and Lucas valued Orbital Angular Momentum (OAM) and to construct matrices. In [10], A. R. Dixon & H. Sato had reported details of equally high rate error correction, which is further adaptable to maximise the secure key rate under a range of different operating conditions. The error correction was implemented both in CPU and GPU using a bi-directional LDPC approach.

3 PROPOSED METHODOLOGY

In cryptography, encryption and decryption depend on the key. Quantum cryptography is used to distribute the key. The key is first shared by the sender and receiver. So it is important to secure the key during and after key distribution. Enhancing security by detecting and correcting the error is based on the proposed HAmming Code with Modified ELgaMAl (HACMELMA). The proposed method attempts to detect and correct errors during the key distribution phase. After that, the SSK retrieved is used as a private key for encryption using HAmming Code with Modified ELgaMAl (HACMELMA). As proposed in [11], ASCII encoding is used for text encoding. Here to detect and correct errors, Hamming Code method is used. That is after photon polarization is done by the sender, the receiver measures the polarized photons and retrieve the key. If the key mismatch from the original, then the error detecting mechanism is applied. The proposed HAmming Code with Modified ELgaMAl (HACMELMA) algorithm describes the error detecting and correcting mechanism in QKD.

The steps involved in proposed HACMELMA algorithm

1. For Sender
   a. Form the basis using BB84 protocol.
   b. Generate sender's random bit p based on MRSA.
   c. Polarize the photons and distribute the key.
   d. Polarized photon is sent to the receiver by sender.
2. For Receiver
   a. Perform random measuring basis based on MRSA.
   b. Measure the polarized photons and retrieve the key.
   c. The shared secret key is generated.
   d. The receiver discusses the shared secret key with the sender.
3. If the retrieved key>p bits (p is the overall bits), then the receiver finds that the key is eavesdropped. That is key=e. (e is the error in key).
4. Then the users detect and correct the error in the key using Hamming Code method.
5. After correcting the errors, the shared secret key is generated.
6. Encryption and Decryption
   a. s is taken as private key.
   b. Choose a large prime p.
   c. Choose a generator element g, g<p-1.
   d. Compute $y=g^k$ mod p.
   e. Use the shared secret key s as private key.
   f. Perform encryption using $a=g^x$ mod p.
   g. Perform encryption using $b= y^k$ M mod p.
   h. For public key (p,g,y), g must be a generator of p.
   i. For performing encryption of each bit in the plaintext, ASCII value of the text will be taken.
   j. Perform decryption using $M=b/a^x$ mod p.
7. Repeat the steps 4 and 5 until k≠e.

Algorithm1: EC(QKD, be, HAC, pp, e, rb, SSK, k)

Begin{main}
1. form basis using QKD
2. read rb
3. pp←rb
4. measure pp
5. generate SSK
6. if (k=e) then
7. find be
8. compute HAC (be)
9. if (k≠e)
10. compute SSK

End{main}

3.1 Hamming Code for Error Detection and Correction

In any communication channel, there will be two beneficiaries to share the data, sender and receiver. To secure the data we use a choice of cryptography. But cryptographic algorithms are vulnerable to various attacks, which results in the evolution of quantum cryptography. As in [11], quantum cryptography is merely meant for secure distribution of the key. There are chances for an adversary to eavesdrop the shared key. If the key has eavesdropped, then it is altered and sent again to the receiver as the original key by the eavesdropper. This leads to an error in the actual key. If the eavesdropper retrieved any information of the polarized photon then it causes an error in the receiver's measurement. To detect the error, this paper uses the Hamming code method and corrects the error. The error correction is done after the pre-processing steps as proposed in MRSA [11]. The retrieved key is analyzed by the sender and receiver. If the key mismatches, that is, if the key ≠ e then the key is directly taken as the private key for encryption in HACMELMA. But if the key = e, then the error is corrected using the Hamming Code method. After the error correction, the generated errorless key is taken as SSK. The proposed HACMELMA detect and correct the error which results in improved security. Usually, if eavesdropping occurs during key distribution the sender and receiver discards the process and retransmits through another channel but here there is no need of retransmission. Because the error is detected and corrected at the stage once the user confirms that the key has eavesdropped. The Hamming code method is used to correct the errors. Hamming code is a type of binary codes. It is used to detect and correct the data transmission errors while transmitting the data from the sender to the receiver [1].

TABLE 2

BINARY TABLE

Decimal	Binary	P_4	P_3	P_2	P_1
1	00001	0	0	0	1
2	00010	0	0	1	0
3	00011	0	0	1	1
4	00100	0	1	0	0
5	00101	0	1	0	1
6	00110	0	1	1	0
7	00111	0	1	1	1
8	01000	1	0	0	0
9	01001	1	0	0	1
10	01010	1	0	1	0
11	01011	1	0	1	1
12	01100	1	1	0	0
Bit Position		8,9,10, 11,12	4,5,6, 7,12	2,3,6, 7,10,11	1,3,5, 7,9,11

The Hamming code is the concept which is invented by Richard. W. Hamming in the late 1940s. Hamming codes are the set of binary linear codes. For each integer r ≥ 2, there is a code with block length n = 2r – 1. The message length k = 2r − r − 1. The rate of Hamming codes is R = k / n = 1 − r / (2r − 1). The minimal number of bit changes needed to go from one code-word to any other code-word is three and the block length 2r − 1. The parity-check matrix of a Hamming code is constructed by listing all columns of length r that are non-zero, which means that the dual code of the Hamming code. The parity-check matrix has the property that any two columns are pair-wise linearly independent [2].

First using BB84 protocol as in [11], the photon polarization structure is created. Then the photon polarization basis is formed. The formed key is sent to the receiver. In the same basis, the receiver retrieves the key. During that, the sender and receiver discuss the key publically. If the receiver's generated random basis mismatches from the sender's, then they conclude that an eavesdropper eavesdrop the key. That is in many situations, there is a chance that the eavesdropper can alter the bit of actual key and send to the receiver as an actual key. So in this situation, there is a need for error detection. The receiver uses the Hamming code method to predict the bit which an eavesdropper altered. To do so, in Hamming Code the original message is taken and the Hamming Code is reconstructed using equation (1),

$$2^k \geq m+k+1 \qquad (1)$$

Here, m is the number of bits, k is the integer taken which is converted into binary. The total length of the message is calculated as length = m+k. These are used to find the range (r) of bits that is till which bit the process should continue. Then after forming the binary table, the parity bits are generated using the decimal and its corresponding binary value which is taken as the message. According to that, the parity bits are generated and in the parity bit, the even parity value is located in each decimal. After locating the parity bit, the bit position is generated and with that bit position, the whole message is computed with the parity bit.

In that, the numbers generated in the range computed from equation (1) is taken as parity bits and the original message (m) is taken as the remaining till m bits. The HAmming Code (HAC) is generated from this and the bit position is named from right to left. Change in any of this bit is detected using Hamming Code. To detect the error bit position, the coded message is taken and the parity positions are computed and the resulting bits are combined and the error is detected finally. After detecting the bit position of the error, it is then corrected and used as a shared secret key.

3.2 Error Detection and Correction using HAC method- An Example

In this paper, the BB84 protocol is taken from [11] to generate a random basis and to polarize the photons. Then the error is detected using Hamming Code and it is corrected. After error detection and correction, the shared secret key is generated. Finally, it is used as a private key in the proposed HACMELMA. The key is shared through the quantum channel and eavesdropping is noted which is shown in Table.1. The error is detected and corrected which is shown in Table.2 and Table.3.From the above table, the Hamming Code is generated and the error is detected in the 11th and 12th columns and that is highlighted and shown. The 11th column is the original message the 12th highlighted column is the erroneous message. For that, the integer 2789 is taken and is converted into binary $(2789)_2$ = 101011100101. Hence the number of bits m=12. Using equation (1), $2^5 \geq 12+5+1$, that is 2^0, 2^1, 2^2, 2^3, 2^4, 2^{5-1}. The range r is {1,2,4,8,16}. The binary

TABLE 1

QKD, GENERATION OF SSK AND ERROR DETECTION

Sender's rb	1	0	1	0	1	1	1	0	0	1	1	0	1
Sender's random sending basis	\	/	\	/	+	\	+	\	+	/	/	×	+
PP Sender sends	↓	←	↘	←	↗	↘	→	↓	↗	↙	↙	↑	↗
Eve's random measuring basis	×	+	×	×	+	+	+	/	+	/	/	/	+
Polarization Eve measures and sends	↑	↗	↖	↖	→	→	→	←	→	↙	←	←	→
Receiver's random measuring basis	×	/	\	/	/	\	/	+	×	/	/	\	+
PP Receiver measures	↑	←	↓	←	←	↘	↙	↗	↖	↙	←	↓	↗
SSK		0		0		1					0		1
Errors in key		✘		✘		✘					✔		✘

*The highlighted column is the error made by the eavesdropper

table 2 is generated as follows.

Bit Position is calculated and from that the Hamming Code is generated. The bit position here is {1,2,3,4,5,6,7,8,9,10,11,12}. This is represented as shown in Table.3.

Hamming Code=10010101001110001. From this it is noted that the error is in this message. To check the bit position of the error,

TABLE 4

ENCRYPTION AND DECRYPTION FOR M USING PROPOSED HACMELMA

S.No	Plain Text (M)	ASCII (M)	Encryption Cipher text C=(a,b)		Decryption $M=b/a^x \bmod p$	Plain Text
			$a=g^k \bmod p$	$b=y^k M \bmod p$		
1	Q	81	3	37	81	Q
2	U	85	3	63	85	U
3	A	65	3	22	65	A
4	N	78	3	62	78	N
5	T	84	3	12	84	T
6	U	85	3	63	85	U
7	M	77	3	11	77	M

C1= 8, 9,10,11,12 = 00100 = Even Parity=1
C2= 4, 5, 6, 7, 12 = 01110 = Even Parity=1
C3= 2, 3, 6, 7, 10, 11 = 011110 = Odd Parity=0
C3= 1, 3, 5, 7, 9, 11 = 111100 = Odd Parity=0

TABLE 3

PARITY BIT GENERATION

1	2	4	8	16	3	5	6	7	9	10	11	12	13	14	15	16
P_1	P_2	P_3	P_4	P_5	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
				0	1	0	1	0	0	1	1	1	0	1	0	1
			1		1	0	1	0	0	1	1	1	0	1	0	1
		0			1	0	1	0	0	1	1	1	0	1	0	1
	0				1	0	1	0	0	1	1	1	0	1	0	1
1					1	0	1	0	0	1	1	1	0	1	0	1

The erroneous bit position is identified as 0011 that is 3. Like this whenever error occurs, the Hamming Code attempts to detect and correct it using the above steps.

4 ENCRYPTION AND DECRYPTION USING HACMELMA

The major part of encryption is key distribution. In this paper, the key is distributed using BB84 protocol and the error in the key is detected and corrected using the HAC method. At the same time encryption and decryption is performed using the Modified Elgamal algorithm. In this proposed HACMELMA method, the security is enhanced by using a generator that is for encryption a generator element is used. The private key is generated from the distributed SSK and the
encryption/decryption is performed. For text encryption, the plaintext message is taken and its ASCII value is found for each character. Then the ASCII value of each character is taken as the public key and the encryption is performed. This provides an additional layer of security [11].

TABLE 5

ENCRYPTION AND DECRYPTION OF EXISTING ELGAMAL

File Size (MB)	Encryption	Decryption	Security (in %)
1	1176	1301	97
2	2901	3163	95
3	4943	5012	97
4	6578	7102	95
5	8401	8573	96

4.1 HAMMING CODE WITH MODIFIED ELGAMAL (HACMELMA)

Elgamal is a public-key encryption scheme. It depends on the difficulty of finding discrete logarithm. Hence encryption and decryption are highly complex. First, a key pair is generated here. For that first, two random numbers g, x and prime p are chosen. Here, g is a generator and x is the private key in which g and x must be less than p that is g,x<p. After this, public key y is generated using y=g x mod p. For encrypting a message M, a random key k is chosen first in which k is relatively prime to p-1. Then the pair of ciphertext a and b are computed. This enhances the security hence by using pairs of ciphertext, it is hard to predict. Even if one ciphertext is eavesdropped, finding the other is difficult here because the generator g is difficult to compute. Also the private is used here is not generated directly it is taken from the result of key distribution. The ciphertext pairs are computed as, a=gk mod p and b=ykM mod p. For one plaintext, two ciphertexts are computed. For decryption, compute M=b/ax mod p.

4.2 HACMELMA - An Example

In this section, an example is illustrated in order to understand the relevance of the work. The private key x is generated from the BB84 protocol after error detection and correction. Thus (g,p,y) is chosen based on x. Let p=89 and g is generated as g<p-1 so g=23. As x is generated earlier, x=7, k=17. Thus the private key is {7, 89}. Then y is computed as y=gx mod p. Here, y=237 mod 89, y=62. So the public-key is {89, 23, 62}. By this, both the public and private keys are used for encryption and decryption. To encrypt a message, M = “QUANTUM”, the ASCII value of each character is taken. Thus by computing “QUANTUM” {81, 85, 65, 78, 84, 85, 77}, the M is encrypted by a pair of ciphertexts. That is for each character, a pair of ciphertexts is generated. That is encrypting the character ‘Q’, the ASCII of ‘Q’ is 81. Compute a=gk mod p. Hence, a=2317 mod 89. So a=3. Then, b=yk M mod p = 6217 81 mod 89, b=37. Thus M is encrypted as C{c1,c2}={3,37}. The one ciphertext pair ‘a’ is same for all the characters but b varies. Then to decrypt the message, compute M=b/ax mod p. M=34/37 mod 89. First, compute 37 mod 89=51. Then the inverse of 51 is computed. That is M= 34×51-1 mod 89=7. So M=37×7 mod 89=81. The plaintext is recovered. Likewise, each and every character of M is computed in the same way. Table.4 shows the encryption and decryption for M using proposed HACMELMA.

5 RESULTS AND DISCUSSION

The proposed methodology is implemented in VC++ with version 6.0. The encryption, decryption time and security are calculated and it is compared with existing ElGamal, MRSA and proposed HACMELMA. Table.5 shows the experimental results of existing ElGamal. Table.6 shows the encryption, decryption and security of proposed HACMELMA method. Fig. 1 shows the Comparison graph for Encryption and Fig. 2 shows Decryption time for MRSA, existing ElGamal and proposed MRSA and Fig. 3 show the comparison result of security.

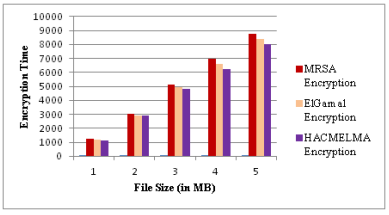

Fig. 1*. Comparison result of Encryption time for MRSA, existing ElGamal and proposed*

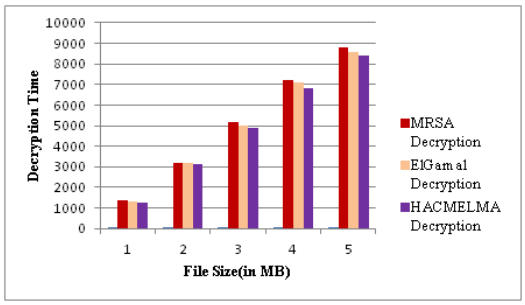

***Fig. 2**. Comparison result of Decryption time for MRSA, existing ElGamal and proposed HACMELMA.*

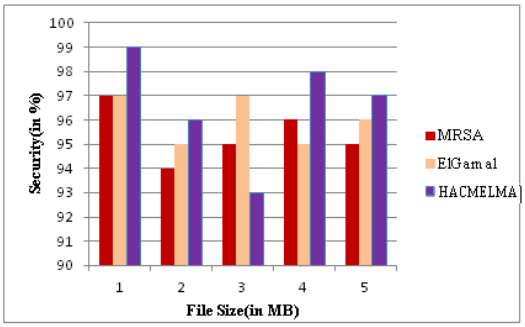

***Fig. 3**. Security Comparison graph for Security level for MRSA, existing ElGamal and proposed HACMELMA.*

6 CONCLUSION

In previous work, the eavesdropping was noted and the process is discarded and the communication is transmitted through another channel. But in this proposed HACMELMA the eavesdropping is noted and the error is detected and corrected after sharing the key. The corrected key is then taken as SSK which is used as a private key in HACMELMA. Thus a modified method for encrypting the message is proposed. In this method, security is enhanced by using the SSK as a private key and also the ciphertext is taken as pair which in turn makes the decryption difficult. The proposed HACMELMA is compared with MRSA, existing ElGamal. This can be further extended by incorporating the same in the IoT environment for a secured communication channel.

TABLE 6

ENCRYPTION AND DECRYPTION OF PROPOSED HACMELMA

File Size (MB)	Encryption	Decryption	Security (in %)
1	1135	1286	99
2	2879	3115	96
3	4789	4863	93
4	6205	6804	98
5	8017	8376	97

REFERENCES

[1] https://www.geeksforgeeks.org/computer-network-hamming-code
[2] https://en.wikipedia.org/wiki/Hamming_code
[3] https://math.asu.edu/sites/default/files/elgamal.pdf
[4] https://en.wikipedia.org/wiki/Quantum_cryptography
[5] https://en.wikipedia.org/wiki/Quantum_error_correction
[6] Akihiro Yamamura and Hirokazu Ishizuka, "Error Detection and Authentication in Quantum Key Distribution", Springer-Verlag, pp. 260-273, 2001.
[7] Xiangyu Wang, Yichen Zhang, Song Yu & Hong Guo, "High speed error correction for continuous-variable quantum key distribution with multi-edge type LDPC code", Scientific Reports, pp. 1-7, 2018.
[8] A. Goneid, S. El-Kassas, M. El-Ashmawy and A. Abbas" Enhancement of Error Correction in Quantum Cryptography BB84 Protocol", Egyptian Computer Science Journal, pp.1-7, Jan 2009.
[9] Hong Lai and et al., "Fast and simple high-capacity quantum cryptography with error detection", Scientific Reports, 2017.
[10] A. R. Dixon & H. Sato, "High speed and adaptable error correction for megabit/s rate quantum key distribution", Scientific Reports, 2014.
[11] A. Beatrice Dorothy and S. Britto Ramesh Kumar, "An approach for IoT security using Quantum key distribution", International Journal of Scientific & Technology Research, pp. 1569-1574, 2019.
[12] A. Beatrice Dorothy and S. Britto Ramesh Kumar, "DORBRI: An Architecture for the DoD Security Breaches Through Quantum IoT", Springer Nature Singapore, pp. 491-496, 2018.