A Comprehensive Literature Review of Asymmetric Key Cryptography Algorithms for Establishment of the Existing Gap

Juliet N. Gaithuru
Faculty of Computing
Universiti Teknologi Malaysia
81310, Skudai, Johor Bahru
Malaysia
Email: julietgaithuru@yahoo.com

Majid Bakhtiari
Advanced Informatics School
Universiti Teknologi Malaysia
54100 Kuala Lumpur
Malaysia
Email: bakhtiari.majid@gmail.com

Mazleena Salleh
Faculty of Computing
Universiti Teknologi Malaysia
81310, Skudai, Johor Bahru
Malaysia
Email:mazleena@fc.utm.my

Alwuhayd M. Muteb
Saudi Arabia
Email: mutebmmw@hotmail.com

***Abstract*—Information security has become a key concern for communication over public channels thus necessitating the use of cryptography to safeguard communication. In this paper, we present a comprehensive review of the asymmetric key algorithms, beginning from the inception of asymmetric cryptography in 1976 till the present day. This paper provides a description of their encryption and decryption operations, points out their security basis, areas of implementation, their strengths and weaknesses during operation. Finally, the paper pinpoints the existing gap based on conclusions drawn from the review, with particular emphasis on an algorithm most suited for industrial application given the current trends in cryptography towards quantum computing. This paper then narrows down on the earnest need for an algorithm that has no trade-off in encryption and decryption speeds, has low computation overhead and is secure enough to withstand quantum algorithm attacks.**

Keywords*—Public key cryptography; RSA; Knapsack; McEliece; Rabin; ECC; ElGamal; NTRU.

I. Introduction

Today, the internet is a global information superhighway which supports a host of services ranging from education, medicine, entertainment to e-commerce [1]. Cryptography is used to ensure security of information during transmission from one party to another and is used every time a user sends credit card information online during a transaction [2]. It is especially used for authentication, digital signatures and e-commerce [3]. Cryptography is defined as the science of secret writing. This helps to ensure the fundamental principles of information security; confidentiality, integrity, authentication and non-repudiation.

There are two main classifications of cryptography algorithms: symmetric key algorithms and asymmetric key algorithms. Symmetric key algorithms use a shared secret key for encrypting and decrypting information between two authorized parties. Therefore the private key is distributed before transmission of the message [4].There are various symmetric key algorithms including enigma (involves shifting letters), the one-time pad, Data Encryption Standard (DES), 2DES, 3DES and the Advanced Encryption Standard (AES) [5].

Asymmetric key algorithms, on the other hand, use a public key for encryption and the recipient decrypts the information using a private key. The public key is widely available to anyone on the network since the sender must know the recipient‘s public key so that he can encrypt the message. The private key is kept secret and is only known to the authorized recipient so as to enable him to decrypt the received message using his own private key. Therefore, unlike symmetric key algorithms, key distribution is not a problem [4]. Asymmetric key algorithms include Rabin, McEliece, Knapsack, Probabilistic key cryptosystem, Elgamal, RSA, Elliptic Curve Cryptosystem (ECC) and the most current NTRU (N^{th} Degree Truncated Polynomial) cryptosystem, which is a lattice-based cryptosystem and which is known to be resistant to quantum computing algorithms [6].

This paper focuses on the public key cryptography algorithms. It is worth noting that private key algorithms are 2 to 3 orders of magnitude faster than public key algorithms [2]. However, public key algorithms provide more enhanced security and have wider implementation. This paper provides a chronological insight into the development of public key algorithms following its first introduction in 1976 by three researchers; Whitfield Diffie, Martin Hellman and Ralph Merkle [7] up to the present day.

Cryptographic algorithms can be categorized on the basis of the number of keys used for encryption and decryption and further defined on the basis of their areas of application and use [8]. The two main categories of algorithms are: Secret key cryptography and Public key cryptography. These two categories are further elaborated in the subsequent section.

The rest of this paper is structured as follows. Section II describes secret key cryptography followed by Section III which conducts an in-depth review of public key cryptography algorithms. This is followed by a discussion in Section IV and finally a description of the future works is provided in Section

978-1-4673-8227-4/15/$31.00 ©2015 IEEE

v.

II. Secret Key Cryptography (SKC)

Secret key cryptography uses a single shared private key for both encryption and decryption processes. Given that a single key is used, secret key cryptography is also referred to as symmetric encryption. The security of the encrypted message is dependent on the nature of the key, for instance key length. Secret key algorithms can easily be implemented on hardware and have higher performance in comparison to the public key algorithms. However, one major shortcoming of secret key cryptography is the problem of key distribution between the sender and the recipient [4]. Secret key cryptography can be further sub-divided into block ciphers and stream ciphers. Modern day secret key cryptographic algorithms (block ciphers) include the Data Encryption Standard (DES), AES, CAST, IDEA, Rivest Cipher 5(RC5) and Blowfish [8].

III. Public Key Cryptography

Public key cryptography was invented in 1976 by Whitfield Diffie, Martin Hellman and Ralph Merkle [7], [9]. This is where the recipient‘s public key is used for encryption and the recipient uses his own private key for decryption. The public key is known to everyone on the network as it is only used for encryption, but only the authorized recipient can decrypt the message using his private key which is only known to him. Both keys are required and due to the implementation of two keys, it is referred to as asymmetric cryptography. The security of PKC is based on hard mathematical problems, namely the existence of one-way functions, or mathematical functions which are easy to compute but their inverse function is relatively difficult to obtain [4]. Examples include: factoring large primes -utilized in RSA, Rabin and LUC as well as solving the discrete logarithm problem- utilized in ECC [8] public key algorithms.

For instance, if Alice wants to send a message to Bob, then first of all Alice and Bob should know each other‘s public keys while the private keys are kept secret. Alice encrypts the message using Bob‘s public key and then sends the ciphertext (encrypted message) to Bob. Bob decrypts the ciphertext using his private key in order to retrieve the plaintext.

In comparison to private key algorithms, public key algorithms have lower performance; they run at a lower speed and have a higher memory requirement. For instance, when encrypting a 192-bit message, a 192-bit AES provides an equivalent security level with RSA whose key size is 7680 bits while a 256-bit AES has equivalent security level with RSA whose key size is 15,360 bits [4]. The modern public key cryptography algorithms include: RSA, Knapsack, McEliece, Rabin, ECC, ElGamal, EcElgamal, NTRU and LUC.

A description of these public key algorithms in their chronological order of invention starting from 1976 to the most recent one in 1998 is provided in the subsequent sections. This is accompanied by a description of their encryption and decryption processes, strengths, weaknesses along with a discussion of the most current developments in asymmetric cryptography in the subsequent section.

The Diffie Hellman Key Exchange algorithm, proposed by its inventors in 1976 [10] was the first solution to the problem of transmitting a shared secret key over an insecure channel. This enabled secure communication between two parties even though they have not communicated with each other before and was later modified by ElGamal and implemented in ECC. This was a significant contribution in facilitating the feasibility of the implementation of public key cryptography.

A. RSA

RSA is named after its three inventors Ronald Rivest, Adi Shamir, and Leonard Adleman in 1978 [7]. RSA was the first method of public key cryptography that was invented and is still one of the most popular algorithms today [11]. RSA uses a variable size encryption block and a variable size key. It is used in hundreds of software products for key exchange, digital signatures, or encryption of small blocks of data [8].

The security of RSA is based on the difficulty in factoring large primes. However, computers‘ ability to factor large numbers has greatly improved thus making it easy to attack RSA. Today, computer systems can factor large primes with over 200 digits. Nevertheless, if a large number is created from two prime factors which are nearly the same size, no factorization algorithm is known which can solve the problem in reasonable time duration. for instance, a test carried out in 2005 to factor a 200-digit number took a duration of 1.5 years and over 50 years of computation time [8].

To generate public and private keys for use in RSA encryption, two large primes p and q are generated, which should be nearly the same size and about 512 bits in length. The values of n and φ are computed as follows:

$$n = pq; \qquad \varphi = (p-1)(q-1). \tag{1}$$

A random integer e is then selected such that $1 < e < \varphi$ and $gcd(e, \varphi) = 1$ which serves as a private exponent. Subsequently, a unique integer d is computed such that $1 < d < \varphi$ and $ed = 1 \pmod{\varphi}$. The computed public key is given by (n, e) and the private key is d. To encrypt a message m, the sender (Alice), obtains the recipients (Bob) public key (n, e). The message m is then represented as an integer such that $0 < m < n - 1$. Alice encrypts the message m by computing

$$C = m^e \mod n \tag{2}$$

and the ciphertext C is sent to Bob. When Bob receives the ciphertext C, he recovers the plaintext m using the private key d by computing

$$m = c^d \mod n. \tag{3}$$

The proof of decryption can be shown using Fermats theorem. The main strength of this algorithm is that no attack has been found that solves the integer factorization problem in reasonable time. The weaknesses of RSA are broadly categorized into three attack strategies.

The first attack strategy is the brute force attack. It is implemented by launching an attack on the factorization method given the difficulty in factoring the product of large primes. The solutions devised to solve this problem involve the use of the simple index calculus algorithm, Pollard‘s $p - 1$ method, number field sieve attack and the quadratic sieve factoring method.

The second attack strategy is the implementation attack or side channel attack which seeks to exploit faults that result in information leakage during RSA implementation. The attacks that exploit these implementation faults include timing attacks, power analysis attacks and fault analysis attacks. These attacks are especially launched on security tokens as well as smart cards.

The last attack strategy is the mathematical attack which seeks to obtain the values of p, q or φ. This is done by exploiting improper selections of e and d, the use of a low exponent e, poor message padding and relations between encrypted messages such as having a common public modulus for an entire organization [12], [13].

The probability of failure of these attack strategies can be greatly increased by choosing an exponent that is equal to or greater than 2048 bits [12], [13] based on studies conducted using the number field sieve algorithm.

B. Knapsack Public-Key Encryption

The Knapsack scheme was invented in 1978 and is built on the basis of the subset sum problem [14]. Knapsack works by selecting an instance of the subset sum problem which is easily solvable, and then disguises it as an instance of the general subset sum problem which is considered to be considerably more difficult to solve. The original knapsack set can serve as the private key, while the transformed knapsack set serves as the public key. There are several variations of the knapsack scheme which include the Merkle-Hellman knapsack encryption scheme and the Chor-Rivest knapsack encryption schemes.

The Merkle-Hellman knapsack encryption scheme was the first concrete realization of a public-key encryption scheme. Several other variations have been developed over time, but most have been proven to be insecure except for the Chor-Rivest knapsack scheme [15].

1) Merkle-Hellman Knapsack Encryption Scheme: This scheme disguises an easily solvable instance of the subset sum problem by modular multiplication and a permutation. To generate the public and private keys, n is fixed as a common system parameter. A super-increasing sequence $(b_1, b_2, \ldots, b_n)$ is obtained and modulus M such that $M > b_1 + b_2 \ldots + b_n$. A random integer W, the multiplier, is then selected , and $1 \leq W \leq M-1$ such that $gcd\,(W, M) = 1$. A random permutation Π of integers $\{1, 2, \ldots, n\}$ is generated.a_1 is obtained by computing

$$a_1 = W b_{\Pi(i)} \bmod M \text{for } i = 1, 2, \ldots, n. \tag{4}$$

The computed public key is given by $(a_1, a_2, \ldots, a_n)$ while the private key is $(\Pi, M, W, (b_1, b_2, \ldots, b_n))$. To encrypt a message m , the sender (Alice) represents it as a binary string $m = m_1, m_2, \ldots, m_n$ which has a length of n. The ciphertext C is obtained by computing:

$$C = m_1 a_1 + m_2 a_2 + \ldots + m_n a_n. \tag{5}$$

When the receiver (Bob) receives the ciphertext, he decrypts it to retrieve the message by computing

$$d = W^{-1} C \bmod M. \tag{6}$$

The obtained message bits are given by $m_i = r_{\Pi(i)}, i = 1, 2, \ldots, n$.

The Merkle-Hellman knapsack scheme is deemed to be insecure due to the presence of a known polynomial time algorithm which can be used to successfully retrieve the plaintext. It can also be broken by solving the short vector lattice problem [11].

2) Chor-Rivest knapsack encryption: Following the proof of successful attacks on the Merkle-Hellman Knapsack scheme, the Chor-Rivest scheme was developed in 1988, which was based on the unique representation of sums in finite sequences [16]. The Chor-Rivest scheme is considered to be the only known knapsack public-key encryption scheme which does not use a form of modular multiplication in order to disguise an easy subset sum problem. The encryption process of this scheme takes the form of the equation:

$$C = \sum_{i=0}^{p-1} m_i \, c_i \bmod \left(p^h - 1\right). \tag{7}$$

The decryption process of this scheme takes the form

$$m \leftarrow 0; l \leftarrow h m_{i-1} = 1 \text{set } m \leftarrow m + \binom{p-i}{l}; \quad l \leftarrow l - 1. \tag{8}$$

The strength of this scheme is that it has a fast encryption speed but decryption is slow. The main weakness of this scheme is that the size of the public key is fairly large. Furthermore, the Chor-Rivest scheme is vulnerable to attack by algorithms which are used for lattice basis reduction [11].

C. McEliece Public-Key Encryption Scheme

The McEliece public-key encryption scheme was developed in 1978 and is based on error-correcting codes. This encryption scheme was based on the concept that there is a decoding scheme for Goppa codes, but none exists for a general linear code [17]. Therefore, this resulted in the generation of the idea of first selecting Goppa codes (a class of error correcting codes) which have a known efficient decoding algorithm and then the code is disguised as a general linear code. Given that decoding an arbitrary linear code involves a difficult process, a description of the original code can serve as the private key, while a description of the transformed code serves as the public key.

The McEliece encryption scheme has been known to be resistant to cryptanalysis to this date. It is also considered to be the first public-key encryption scheme to apply randomization in the encryption process. However, this encryption scheme has not been widely implemented practically due to the very large size of its public keys [15]. To generate the public and private keys, the integers k, n, t are fixed as common system parameters. Then $k \times n$ generator matrix G for a binary (n, k) is chosen. This is a linear code which can correct t errors, with a known decoding algorithm. A random $k \times k$ binary non-singular matrix S is selected, followed by a random $n \times n$ permutation matrix P . The $k \times k$ matrix G is obtained by computing $G = SGP$. The computed public key is given by (G, t), while the private key is (S, G, P).

To encrypt a message m of length k , the sender (Alice) chooses a random binary vector z of length n having at most t 1's. She then encrypts the message to obtain the ciphertext C by computing the binary vector $\hat{c}$

$$C = m \cdot \widehat{G} + z. \tag{9}$$

The ciphertext is sent to the recipient (Bob). When Bob receives the ciphertext, he computes $\widehat{c} = c\,P^{-1}$ where P^{-1} is the inverse of matrix P. The decoding algorithm for the code generated by G is then used to decode $\widehat{c}$ to $\widehat{m}$. The recipient retrieves the plaintext m by computing

$$m = \widehat{m} \cdot \ S^{-1}. \tag{10}$$

The recommended parameter sizes are $n = 1024$, $t = 38$, $k \geq 644$.

The strength of the McEliece encryption scheme is that it has relatively fast encryption and decryption speeds. However, the weakness of this scheme is that it has a very large size of public key. In addition, it has a message expansion factor of $\frac{n}{k}$, which if applied to the recommended parameters gives a factor of 1.6. These two drawbacks make it less popular in industrial applications [11].

D. Rabin

The Rabin encryption scheme was proposed in 1980 as a solution to the problem of determining whether numbers are prime or composite through employing the use of probabilistic techniques. This generated solution helped to generate very large primes which meet a desired criteria [18]. Rabin is based on difficulty in factoring large primes. In the Rabin scheme, each entity creates a public key and a corresponding private key [19].

To generate the public and private keys, two large random and distinct primes p and q are generated, which are nearly the same size. The value of n is obtained by computing $n = pq$. The computed public key is given by n, while the private key is given by (p, q). To encrypt a message m, the sender (Alice) obtains the recipient's (Bob's) public key n. The message to be sent is represented as an integer m which lies in the range $0 < m < n-1$. The sender encrypts the message by computing

$$C = m^2 \mod n \tag{11}$$

to obtain the ciphertext, which is then sent to the recipient (Bob). Upon receiving the ciphertext, Bob decrypts it using the Chinese Remainder Theorem to find the four square roots m_1, m_2, m_3 and m_4 of $C\,modulo\,n^2$. The plaintext is either m_1, m_2, m_3 or m_4 with equal probability [11], [19].

The strengths of the Rabin encryption scheme include security against a passive adversary attack. Secondly, the encryption process in the Rabin scheme is extremely fast due to the single modular squaring [11].

The Rabin scheme also possesses some weaknesses. Firstly, the Rabin scheme is not deterministic since the recipient (Bob) has to choose the correct plaintext from four equally likely possibilities. Secondly, the scheme has a slower decryption speed, but which is comparable to RSA. Thirdly, the Rabin encryption scheme is also susceptible to the RSA attacks [11].

E. Elliptic Curve Cryptosystems

The use of elliptic curves in cryptography was first introduced in 1985 as an alternative to other public key cryptosystems [20]. The points on an elliptic curve which are defined over a finite field have favourable properties for use in cryptographic applications. The security of ECC is based on the difficulty in solving the discrete logarithm problem as opposed to other public key cryptosystems whose security is based on integer factoring and finding discrete logarithms over finite fields which are relatively easier to solve. Therefore, elliptic curve cryptography provides comparable security level with significantly smaller system parameters, such as smaller key size, thus providing efficiency in software and hardware implementations [21]. For instance, a 160-bit ECC key size has security level equivalent to RSA and DSA with modulus 1024 bits.

ECC is favourable because to this day, no sub-exponential algorithm has been found that has been known to solve the discrete logarithm problem and trying to solve it would take fully exponential time. It uses significantly smaller parameters in comparison to DSA and RSA at equivalent security levels. Using a smaller key size means that there will be faster computations, reduced processing power, storage space as well as reduced bandwidth which makes it suitable for implementation in devices with memory constraints such as PDAs, smart cards and mobile phones [22].

For Elliptic curves over prime fields, the most common key size is 160 bits [23]. For ellipitic curves over non-prime fields, the most common key size is $p = 2^{163}$.

A mathematical computation for the NIST prime elliptic curves involves the following: An elliptic curve in the $GF(p)$ where p is a prime number and $p > 3$ is an odd prime has values x, y which satisfy the equation $y^2 = x^3 + ax + b \pmod p$ where $a, b \in GF\,(p)$. F_p is the finite prime field with p elements, where F_p is a set of the integers $\{0, 1, 2, \ldots p-1\}$ with the following mathematical operations [24]:

Addition modulo $p : a + b = r$ and where $0 \leq r \leq p-1$.

Multiplication modulo $p : a \cdot b = s$ where $0 \leq s \leq p-1$.

Any two points R and S on the curve can be added to each other resulting in a new point $R + S$. The addition operation is carried out as follows:

For two points $P = (x_1, y_1) \in E\,(F_p)$ and $Q = (x_2, y_2) \in E\,(F_p)$, the result of addition would be $P + Q = (x_3, y_3)$. The point addition for the two possible instances where $P \neq \pm Q$ and $P = Q$ is carried out as follows:

For $P \neq \pm Q$
The result of the point addition $P + Q = (x_3, y_3)$ would be obtained by computing:

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$
$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{12}$$

For $P = Q$
The result of the point addition $P + Q = 2P =$

(x_3, y_3) would be obtained by computing [24]:

$$x_3 = \lambda^2 - 2x_1 \qquad y_3 = \lambda\,(x_1 - x_3) - y_1$$
$$\lambda = \frac{\left(3x_1^2 + a\right)}{(2y_1)} \tag{13}$$

F. ElGamal

ElGamal provides an alternative to RSA for public key encryption. ElGamal modifies the Diffie Hellman algorithm, except that instead of the two parties (Alice and Bob) computing their partial keys simultaneously, in ElGamal the computation of the keys is done sequentially [25], [26].

The security of ElGamal is based on the difficulty in solving the discrete logarithm problem as well as the Diffie Hellman problem. For example, given $g^a = x$, how do you find a given that $0 \leq a \leq n-1$? Thus, in order to find the exponent a , then a can be viewed as $\log_g x$ [25].

ElGamal cryptosystem begins with the process of key generation where public and private keys are created. Firstly, a large prime p is selected. An integer d is randomly chosen to be a member of the group $G = \langle Z_p^*, X\rangle$ such that $1 \leq d \leq p$. Then e_1 is selected to be a primitive root in the group $G = \langle Z_p^*, X\rangle$. The value of e_2 is obtained by computing $e_2 = e_1^d \bmod p$. Therefore, the computed public key is given by (e_1, e_2, p) while the private key is given by d.

To encrypt a message P (plaintext), the sender (Alice) obtains the recipients(Bob) public key. Then a random integer r is chosen, which belongs to the group $G = \langle Z_p^*, X\rangle$. The sender encrypts the message P by computing:

$$C_1 = e_1^r \bmod p \qquad C_2 = \left(P \times e_2^r \bmod p\right). \tag{14}$$

The resulting ciphertext is C_1 and C_2 , which Alice then sends to Bob. Encryption in ElGamal can be done in polynomial time complexity [19]. When the recipient (Bob) receives the ciphertext pair $C_1,\ C_2$ he can decrypt the message to retrieve the plain text P by computing

$$P = \left(\frac{C_2}{C_1^d}\right) \bmod\ p. \tag{15}$$

The main strength of Elgamal is that it employs randomized encryption so as to provide enhanced cryptographic security. The same plaintext gives a different ciphertext each time it is encrypted. This is because the private key d is a randomly chosen integer which is different for each encryption (similar to the one-time pad). ElGamal is one of the algorithms which utilize randomization in the encryption process, others include McEliece encryption algorithm.

Randomized encryption is used to increase the cryptographic security of the algorithm in either of these following ways: the first option is by increasing the size of the original plaintext message thus making the plaintext indistinguishable; the second alternative is by reducing the likelihood of a successful chosen-plaintext attack as a result of the existence of a one-to-many mapping between the plaintext and ciphertext; and the third option is by reducing the effectiveness of statistical attacks by levelling the priori probability distribution of inputs [2].

The ElGamal encryption scheme also has some weaknesses, which include: Firstly, the ciphertext is expanded by a factor of two meaning that the ciphertext size is twice as long as the plaintext [11], [15]. Secondly, the encryption operation is unconditionally malleable or can be changed without following any condition, thus making it prone to the chosen ciphertext attack. For example, given that the encryption of M results in a ciphertext (C_1, C_2) , then an attacker can easily construct $(C_1, 2\,C_2)$ for the message $2M$.

The third weakness of ElGamal is the possibility of forged signatures, especially if the message is not signed, thus making it vulnerable to a man-in-the-middle attack because the recipient cannot be able to detect that the message‘s contents were altered. However, this problem can be overcome by implementing a web of trust or through the use of a central certification authority [15].

Last but not least, another weakness of ElGamal is that it is possible to break the algorithm if weak choices of the large prime p and the primitive root e_1 are made. However, this shortcoming can be overcome by the use of special preprocessing. Lastly, the encryption requires two modular exponentiations $e_1^r \bmod p$ and $\left(e_1^d\right)^r \bmod p$ which slows down computation. However, these exponentiations can be sped up by using random exponents with low Hamming weights and having exponents large enough to prevent a search using the baby-step giant-step algorithm [11]. The exponent should be high enough to ensure security but small enough to allow reasonable computation time [15].

G. Elliptic Curve Cryptography Simulating ElGamal

The most common technique of encryption using elliptic curves over the $GF(p)$ or $GF(2^n)$ is by simulating the ElGamal cryptosystem. If Bob wants to send a message to Alice, the process begins with key generation followed by encryption by Bob and decryption of the received message by Alice. To generate public and private keys, $E(a, b)$ is chosen with an elliptic curve over the $GF(p)$ or $GF(2^n)$. Then a point $e_1\,(x_1, y_1)$ is selected on the curve. A randomly chosen integer D is selected and used to obtain $e_2\,(x_2, y_2)$ by computing

$$e_2\,(x_2, y_2) = d \times e_1\,(x_1, y_1)\,. \tag{16}$$

This multiplication involves multiple point addition, as is the case in mathematical operations in elliptic curves. Therefore, the computed public key is given by $(e_1\,(x_1, y_1)\,, e_2\,(x_2, y_2)\,, E\,(a, b))$, which is announced, while the private key d is kept secret [19]. To encrypt a message P (plaintext), the sender (Alice) obtains the recipients (Bob) public key and proceeds as follows: She selects a random integer r and uses it to obtain the ciphertext, given as a pair of points, by computing

$$C_1 = e_1 \times rC_2 = P + r \times e_2. \tag{17}$$

The resulting ciphertext C_1 and C_2 is then sent to Bob. Upon receiving C_1 and C_2, Bob decrypts the ciphertext to retrieve the plain text P by computing

$$P = C_2 - (d \times C_1) \tag{18}$$

where the minus sign indicates addition with the inverse of the resulting product of $d \times C_1$. Finding the product involves

multiple point addition. The proof of the decryption process is given by

$$P + r \times e_2 - (d \times e_1 \times r) = P + (r \times d \times e_1) - (r \times d \times e_1) = P + 0 = P. \quad (19)$$

In an elliptic curve, adding two inverse points results in a zero point. The points P, C_1, C_2, e_1 and e_2 all lie on the curve [19].

Presently, the most effective attack against elliptic curve cryptosystems (ECC) is the Pollards rho method which is parallelized over several processors so as to create a linear speedup. The expected run time of the attack is $0.88\sqrt{q}$ group operations. The estimated number of field multiplications per group operation is 12. It is estimated that an attack on 109-bit ECC in which $p = 2^{109}$ would take 18,000 years on a single computer, or one year on 18,000 computers if parallelized [23], [27].

H. NTRU

The N^{th} Degree Truncated Polynomial (NTRU) encryption scheme was proposed in 1996 [26]. This scheme combines mixing and reduction modulo two prime numbers for the encryption process and uses unmixing for the decryption process, which uses the probability theory [27]. Therefore, the security of this scheme is based on the use of polynomial mixing as well as the independence of the reduction modulo operation on the prime integers. Security is also assured by the fact that it is difficult to find very short vectors in most of the lattices [28].

Since then, it has been standardized in IEEE Std 1363.1-2008 and ASC X9.98 [6], [29]. NTRU is a public key algorithm whose development was motivated by the need for a faster public key cryptosystem based on complex mathematical problems other than integer factoring and the discrete logarithm problem [6].

NTRU is a public key cryptosystem which is made up of NTRUEncrypt, which is a public-key encryption scheme, and NTRUSign, which is a digital signature scheme [6]. The security of NTRU is based on the difficulty in solving certain lattice problems, namely: the approximate Closest Vector Problem (appr-CVP) or the Shortest Vector Problem (SVP) in convolution modular lattices. The lattice structure in NTRU is not vulnerable to quantum computing algorithms thus is described as being a quantum-resistant cryptosystem [6], [30]. The NTRU cryptosystem is based on polynomials in the polynomial convolution ring

$$R = \frac{Z[X]}{X^N - 1} \quad (20)$$

. NTRU is based on three integer parameters (N, p, q) and four sets L_f, L_g, L_φ, L_m of polynomials of degree $N - 1$ which have integer coefficients. The integers N, p, q are chosen such that $N > 1$ and p and q are relatively prime. q is much larger than p and p is taken to be 2 or 3 or the polynomial $2 + X$. The values $p, q, X^N - 1$ must be relatively prime in the ring $Z[X]$ [6], [28], [30].Multiplication in this ring is denoted by $*$.

To generate public and private keys for implementation in the encryption, the process proceeds as follows. The generated private key is an element $f \in L_f$ which is invertible modulo p and modulo q thus satisfying these two conditions:

$$F_q * f \equiv 1 \pmod{q} \qquad F_p * f \equiv 1 \pmod{p}. \quad (21)$$

The extended Euclidean algorithm can be used to obtain these values with ease. The private key is expressed as the pair (f, F_p). The public key h is obtained by computing

$$h \equiv p\, F_q * g \pmod{q}. \quad (22)$$

Therefore, the public key is given by the polynomial h. To encrypt a message m , the sender (Alice) selects a message m from the set of plaintexts L_m. She then randomly selects a polynomial $\varphi \in L_\varphi$ and uses the recipient's (Bob) public key h to encrypt the message as follows:

$$e = \varphi * h + m \pmod{q}. \quad (23)$$

Alice then sends the encrypted message e to Bob [28] . When Bob receives the encrypted message e, he will decrypt it using his private key f. He first calculates the value of a as follows:

$$a \equiv f * e \pmod{q}. \quad (24)$$

Where the coefficients of a are chosen such that $-\frac{q}{2} \leq a \leq \frac{q}{2}$. By treating the polynomial a as a polynomial with coefficients which are integers, Bob can retrieve the plaintext message m by computing

$$m = F_p * a \pmod{p} \quad (25)$$

thus retrieving the original plaintext [28].

NTRU has several strengths which include: Firstly, its lattice structure makes it resistant to quantum attacks. Secondly, RSA is faster in comparison to its predecessors; RSA and ECC. In comparison to RSA, it is approximately 5 orders of magnitude faster and in comparison to ECC, it is approximately 3 orders of magnitude faster [31]. These results were obtained subsequent to experimentation on a graphical processing unit.

Thirdly, NTRU allows for parallelized implementation, which is useful for servers used for processing many secured connnections and for use in attack strategies which involve numerous encryption operations [31]. Fourthly, NTRU provides a higher security level in comparison to RSA. The security level of NTRU in comparison to an equivalent symmetric key size, ECC and RSA algorithms is as shown in Table I. These values are based on [32],the EESS1v3 standard of 2015 , IEEE1363.1 standard [33], [34] , NIST recommended key sizes [31] and the results of tests conducted using Java [27], [28], [35], [36]. Table I shows that NTRU can provide equivalent security levels with RSA using a smaller key size. A 167-bit NTRU has equivalent security level with the 512-bit RSA, while the 263-bit NTRU has comparative security level with 1024-bit RSA.

However, ECC has an overall smaller key size in comparison to RSA and NTRU, which makes it suitable for implementation in devices with a low memory requirement. NTRUs convolution structure makes it possible for it to implement considerably smaller key sizes [6], [30].

However, one weakness of NTRU is the possibility of the occurrence of a decryption failure. However, this can be overcome by a careful selection of parameters [30].

TABLE I. COMPARATIVE SECURITY LEVELS OF NTRU, RSA, ECC AND SYMMETRIC KEY ALGORITHMS.

Symmetric Key Size (bits)	RSA	ECC	NTRU
56	512	112	167
80	1024	160	263
112	2048	224	401
128	3072	256	439
192	7680	384	593
256	15360	521	743

NTRU algorithms have wide acceptance in comparison to other recent public key algorithms (such as Braid cryptosystems, Curve 25519). It has been implemented in commercial applications as well as open source models [30]. Due to its high speed and low memory, it has been implemented in mobile devices and smart cards. NTRU Encrypt was accepted as a X9.98 Standard for implementation in the financial services industry. It is also applied in industrial sensors, RFID and medical devices [37].

Other public key algorithms which are less commonly implemented in industrial applications include:

I. LUC

LUC public-key cryptosystem was designed in 1993. It is designed on the basis of Lucas sequences and its security is based on factorization of integers (similar to RSA) [38], [39]. It can be applied for encryption purposes and for signatures which use integer factoring.

To encrypt a message m, the sender (Alice), obtains the recipient's (Bob) public key (e, n) whereby [38]:

$$n = pq \qquad \text{where } p \text{ and } q \text{ are distinct primes} \tag{26}$$

$$e \quad \text{satisfies the condition} \qquad gcd\left[e, \left(p^2-1\right)\left(q^2-1\right)\right]. \tag{27}$$

Then Alice encrypts the message by computing:

$$C = V_e\left(m,\, Q\right) \bmod n. \tag{28}$$

When Bob receives the encrypted message, he decrypts it using his private key d by computing:

$$M = V_{d_c}\left(C, Q\right) \bmod n. \tag{29}$$

The decryption exponent d_c is ciphertext-dependent and is selected in such a way as to satisfy the condition $e\, d_c \equiv 1 \bmod \ lcm\left[p - \left(\frac{D}{p}\right), q - \left(\frac{D}{q}\right)\right]$ where $D = c^2 - 4$ and $\frac{D}{p}$ denotes the Legendre symbol [39]. In addition, $Q = 1$.

Given the similarity between RSA and LUC (security based on the difficulty in factoring large primes), they have similar strengths and weaknesses. In comparison to RSA, LUC utilizes more computational power during encryption and decryption operations of 50% and 80% respectively due to the need for LUC to evaluate Lucas sequences during encryption and decryption operations. This was revealed during a study conducted on an Intel Core 2 Quad processor running parameter sizes p and q of 1024 bits thus resulting in n of 2048 bits and a randomly chosen exponent e of 40 bits in length [39]. This therefore implies that LUC is more resource intensive in comparison to RSA.

A summary and evaluation of the discussed public key algorithms is shown in Table II and elaborated in the subsequent section.

IV. DISCUSSION

As depicted in Table II and as discussed in the previous section, the Merkle-Hellman Knapsack scheme, which is the oldest scheme, based on the subset-sum problem has been deemed to be insecure. The Chor-Rivest knapsack scheme has a fast encryption speed, slow decryption but fairly large public key. Like the Merkle Hellman knapsack scheme, it is insecure because it can be broken by a lattice reduction algorithm.

The McEliece algorithm has a fast encryption and decryption speed, but has a very large public key size which limits its areas of implementation. This is similar to the NTRU algorithm, which despite its fast encryption and decryption speed, has a fairly large public key size. The Rabin encryption scheme has a fast encryption speed but low decryption speed. In addition, Rabin is susceptible to RSA attacks due to their similar modular exponentiation structure. Given that the security of LUC public key cryptosystem is also based on integer factorization, it is also susceptible to similar attacks with RSA. In addition, LUC is more resource intensive than RSA therefore limiting its areas of implementation to those devices that have sufficient memory and processing power to facilitate evaluation of Lucas sequences during each encryption and decryption operation. The ElGamal scheme uses a randomized encryption scheme which enhances security but has a plaintext-ciphertext expansion factor of two. This means that more memory is required to store the ciphertext, though it is undecipherable due to the message expansion. The NTRU algorithm, which is the most recent, is the only algorithm that is resistant to quantum algorithm attacks, thus making it significantly more secure than other algorithms.

Despite the advent of many new public key algorithms, RSA continues to have the highest popularity in implementation, with a popularity of 43% [40]. Since presently longer key sizes are required for ensuring security (2048 bits) in RSA which results in inefficiency in terms of memory consumption thus leading to slow performance in devices with limited memory and processing power, it has necessitated the search for alternative public key algorithms.

As highlighted in the previous section, NTRU is one of the most recently developed public key algorithms. Despite its perceived benefits in terms of high speed of encryption and decryption, it has not experienced widespread implementation. It is mostly implemented in the financial services industry [37].

According to a performance evaluation conducted in 2000, in comparison to RSA1024 and ECC168 (EcElGamal), NTRU had the fastest encryption, decryption and key generation speed. NTRUs speed was approximately two orders of magnitude faster than ECC (in a CPU) . However, NTRU had the largest public key size and the message expansion is twice as much as that of ECC at an equivalent security level [27].

V. FUTURE WORKS

The most current development in the field of cryptography is quantum cryptography. Quantum cryptography was proposed by Charles H. Bennet in 1984 [41] and is now becoming

TABLE II. SUMMARY AND EVALUATION OF THE MODERN PUBLIC KEY ALGORITHMS.

Public Key Algorithm	**Year**	**Computational problem-security basis**	**Advantages**	**Disadvantages**
Merkle-Hellman knapsack	1978	Subset sum problem	-	Insecure- can be broken using known polynomial-time algorithm [15], [16]
RSA	1983	Integer factorization problem	No attack possible in reasonable computation time	Susceptible to brute force, side channel and mathematical attacks [12], [13] To ensure long-term security, modulus of at least 2048-bits is recommended [12], [13].
Chor-Rivest knapsack	1988	Subset sum problem	Fast encryption There is no feasible attack on it [11], [16]	Slow decryption Fairly large public key length [11], [16]
McEliece	1978	Linear code decoding problem	Relatively fast encryption and decryption [11]	Very large size of the public key [11]
Rabin	1980	Integer factorization problem square roots modulo composite	Secure against attack by passive adversary Extremely fast encryption due to single modular squaring [11]	Not deterministic Slower decryption speed but comparable to RSA Susceptible to RSA attacks [11]
ECC	1985	Discrete logarithm problem	Uses smaller key size than RSA, DSA, Diffie-Hellman and ElGamal Low computation overhead, low power, memory and bandwidth [24]	The ECDLP can be attacked using Pollards parallelizable rho method [23], [27]
ElGamal	1985	Discrete logarithm problem Diffie-Hellman problem	Uses randomization encryption [11]	Ciphertext is twice as long as plaintext-factor of two Unconditionally malleable thus prone to chosen ciphertext attack Possibility of forged signatures [15] Can be broken in case of weak choice of p and e Encryption is slow as it requires two modular exponentiations [11], [15]
LUC	1993	Integer factorization problem	No attack possible in reasonable computation time	Requires more computational effort for encryption and decryption in comparison to RSA [39]
NTRU	1998	Difficulty in solving certain lattice problems-approximate Closest Vector Problem (appr-CVP) [6]	Quantum-resistant cryptosystem Faster than RSA and ECC Uses smaller key sizes [6], [31]	Decryption failure [30]

popular due to its promise of providing communication that is absolutely secure against eavesdropping. Unlike conventional cryptography, the security of quantum key distribution (QKD) is founded on the laws of physics rather than mathematical complexity [42].

The interest in quantum-based security methods has been growing rapidly in recent years. New implementations of quantum key distribution and new network services supported by this solution are being introduced. The reason behind the growing popularity of quantum cryptography is its unrivaled security level: all eavesdroppers can be revealed through the application of the laws of physics. The no-cloning theorem of quantum mechanics ensures that any measurement modifies the state of the transmitted quantum bit. This modification can be discovered by the sender and the receiver, which makes passive eavesdropping impossible [43], [44].

Bearing in mind this current trend, there is need for the development of an algorithm suitable for industrial application which has fast encryption and decryption speeds without the trade-offs present in the current popular algorithms, as described in the preceding section. The algorithm should also have a low computation overhead in terms of memory and bandwidth and small public key size which makes it suitable for devices with limited processing power and memory like mobile devices, PDAs, RFID and smart cards. The algorithm should also have high security making its cryptanalysis impossible in reasonable time and most of all, it should be resistant to quantum computing algorithm attacks.

Lattice-based cryptography is beneficial in this respect as it is able to provide resistance against quantum-algorithm attacks and can provide speed and size advantages. The computational problems in lattices which form the basis of security of lattices is the difficulty in finding the closest vectors and shortest vectors, otherwise referred to as the Closest Vector Problem (CVP) and the Shortest Vector Problem (SVP) respectively, which are described as being difficult to solve (NP-Hard). The NTRU public key cryptosystem, whose structure is based on lattices, is considered to be the most viable post-quantum public-key encryption algorithm [45]. In a bid to enhance the understanding of the shortest vector problem which is the basis of security in the NTRU, Security Innovation set up an NTRU Challenge [46]. This is expected to further heighten research on the security of NTRU which further solidifies the observations

highlighted in the preceding sections.

REFERENCES

[1] S. Sim and S. Rudkin, "The internetpast, present and future," *BT technology journal*, vol. 15, no. 2, pp. 11–23, 1997.

[2] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[3] J. Katz, "Public-key cryptography," in *Handbook of Information and Communication Security*. Springer, 2010, pp. 21–34.

[4] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 4, no. 05, pp. 877–882, 2012.

[5] F. Lin, "Cryptographys past, present, and future role in society."

[6] W. Whyte and J. Hoffstein, "Ntru," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 858–861.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 26, no. 1, pp. 96–99, 1983.

[8] G. C. Kessler, "Introduction to cryptography," 2012.

[9] B. Schneier, "Applied cryptography: Protocols, algorithms, and source code in c (cloth), jan. 1, 1996."

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

[11] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, "Applied cryptography," 2010.

[12] F. F. Moghaddam, M. T. Alrashdan, and O. Karimi, "A hybrid encryption algorithm based on rsa small-e and efficient-rsa for cloud computing environments," *Journal of Advances in Computer Network*, vol. 1, no. 3, 2013.

[13] A. Abubakar, S. Jabaka, B. I. Tijjani, A. Zeki, H. Chiroma, M. J. Usman, S. Raji, and M. Mahmud, "Cryptanalytic attacks on rivest, shamir, and adleman (rsa) cryptosystem: Issues and challenges." *Journal of Theoretical & Applied Information Technology*, vol. 61, no. 1, 2014.

[14] R. Merkle and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *Information Theory, IEEE Transactions on*, vol. 24, no. 5, pp. 525–530, 1978.

[15] A. V. Meier, "The elgamal cryptosystem," 2005.

[16] B. Chor and R. L. Rivest, "A knapsack-type public key cryptosystem based on arithmetic in finite fields," *Information Theory, IEEE Transactions on*, vol. 34, no. 5, pp. 901–909, 1988.

[17] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN progress report*, vol. 42, no. 44, pp. 114–116, 1978.

[18] M. O. Rabin, "Probabilistic algorithm for testing primality," *Journal of number theory*, vol. 12, no. 1, pp. 128–138, 1980.

[19] A. F. Behrouz, "Cryptography and network security," *McGramHill, International Edition*, 2008.

[20] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in CryptologyCRYPTO85 Proceedings*. Springer, 1986, pp. 417–426.

[21] I. Biehl, B. Meyer, and V. Müller, "Differential fault attacks on elliptic curve cryptosystems," in *Advances in CryptologyCRYPTO 2000*. Springer, 2000, pp. 131–146.

[22] J. Lopez and R. Dahab, "An overview of elliptic curve cryptography," 2000.

[23] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of cryptology*, vol. 14, no. 4, pp. 255–293, 2001.

[24] N. United States Government, "Mathematical routines for the nist prime elliptic curves," , Tech. Rep., 2010.

[25] J. Rothe, "Other public-key cryptosystems and protocols," *Complexity Theory and Cryptology: An Introduction to Cryptocomplexity*, pp. 361–412, 2005.

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

[27] P. Karu and J. Loikkanen, "Practical comparison of fast public-key cryptosystems," in *Telecommunications Software and Multimedia Lab. at Helsinki Univ. of Technology, Seminar on Network Security*. Citeseer, 2001.

[28] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *Algorithmic number theory*. Springer, 1998, pp. 267–288.

[29] "Ieee standard specification for public key cryptographic techniques based on hard problems over lattices," *IEEE Std 1363.1-2008*, pp. C1–69, March 2009.

[30] K. Jarvis and M. Nevins, "Etru: Ntru over the eisenstein integers," *Designs, Codes and Cryptography*, pp. 1–24, 2013.

[31] J. Hermans, F. Vercauteren, and B. Preneel, "Speed records for ntru," in *Topics in Cryptology-CT-RSA 2010*. Springer, 2010, pp. 73–88.

[32] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang, "Choosing parameters for ntruencrypt," 2015. [Online]. Available: http://eprint.iacr.org/2015/708.pdf

[33] "Ieee standard specifications for public-key cryptography," *IEEE Std 1363-2000*, pp. 1–228, Aug 2000.

[34] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte, "Choosing ntruencrypt parameters in light of combined lattice reduction and mitm approaches," in *Applied cryptography and network security*. Springer, 2009, pp. 437–455.

[35] R. D'Souza, "The ntru cryptosystem: Implementation and comparative analysis," pp. 4–9, 2011.

[36] L. Giripunje and S. Nimbhorkar, "Comprehensive security system for mobile network using elliptic curve cryptography over gf (p)," *International Journal*, vol. 3, no. 5, 2013.

[37] T. Buktu. (2011) The ntru project. [Online]. Available: http://ntru.sf.net/

[38] Z. Jiang, Y. Hao, and Y. Wang, "A new public-key encryption scheme based on lucas sequence," *Journal of Electronics*, vol. 22, no. 5, pp. 490–497, 2005. [Online]. Available: http://dx.doi.org/10.1007/BF03037006

[39] G. Brandner, "Rsa, dickson, luc and williams: a study on four polynomial-type public-key cryptosystems," *Applicable Algebra in Engineering, Communication and Computing*, vol. 24, no. 1, pp. 17–36, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00200-012-0181-9

[40] M. Malhotra and A. Singh, "Study of various cryptographic algorithms," vol. 1, no. 3, pp. 77–78, 2013.

[41] C. H. Bennett, G. Brassard *et al.*, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, vol. 175, no. 150. New York, 1984, p. 8.

[42] P. Lam and T. Ralph, "Quantum cryptography: Continuous improvement," *Nature Photonics*, vol. 7, no. 5, pp. 350–352, 2013.

[43] M. Niemiec and A. R. Pach, "Management of security in quantum cryptography," *Communications Magazine, IEEE*, vol. 51, no. 8, 2013.

[44] N. Diwaker, B. Gupta, N. Kumari, and S. Tanwar, "Quantum cryptography: A new approach to information security," *Available at SSRN 2245514*, 2013.

[45] D. Stehlé and R. Steinfeld, "Making ntru as secure as worst-case problems over ideal lattices," in *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 27–47.

[46] M. Wilmington. (2015) Security inovation challenges the crypto community to defeat ntru. [Online]. Available: http://www.prweb.com/releases/2015/02/prweb12492128.htm