An evaluation of the discrete logarithm cryptosystem

Yansheng Chen

Kristin School, Auckland, New Zealand, 0632

jasonchen1584065@gmail.com

Abstract. In modern society, the unauthorized disclosure of digital information such as military files, business details, and personal data can gravely endanger the security and privacy rights of an individual or group. Such risks highlight the significance of cryptosystems – sets of computational algorithms used for safe data transit. This paper specifically explores the algorithms of the discrete logarithm cryptosystem from a number theory perspective and investigates the cryptosystem's overall strengths and weaknesses as a result of their designs. In the end, from various algorithms such as ElGamal encryption and decryption, it was concluded that the use of discrete logarithms and large primes significantly contributed to the advantages and disadvantages of the system, respectively.

Keywords: discrete logarithm, cryptographic algorithms, asymmetric cryptosystem, ElGamal, digital security.

1. Introduction

The discrete logarithm cryptosystem (DLC), first introduced to the public by Taher ElGamal in 1985, is an integral component of communication channels due to its robust security features and versatility. This cryptographic protocol relies upon the computational complexity of the discrete logarithm problem, which renders it resistant to various forms of cryptanalysis. Over the years, the DLC concept has been the foundation of significant cryptographic algorithms, including but not limited to Diffie-Hellman key exchanges, digital signatures, DLC encryption and decryption algorithms. In this research paper, we will review such algorithms, provide a comprehensive analysis of their design, and discuss potential improvements. Essentially, this paper aims to deepen the understanding of readers on the topic of the discrete logarithm cryptosystem.

2. Concepts of public key and private key

Though DLC is comprised of many individual algorithms, any user "i" overall will be assigned two keys: a public key that is known to everyone else in the network and a private key that is only known by the user. To communicate, two functions are required: the decryption function $D_i(x)$ based on the private key of the user "i" and the encryption function $E_i(x)$ based on the public key of the user "i". The two functions are intended to be inverse functions of one another, where $D_i(E_i(x)) = E_i(D_i(x)) = x$. Last but not least, $E_i(x)$ is a one-way function: one that is computationally difficult to calculate the corresponding input of a known output.

© 2023 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

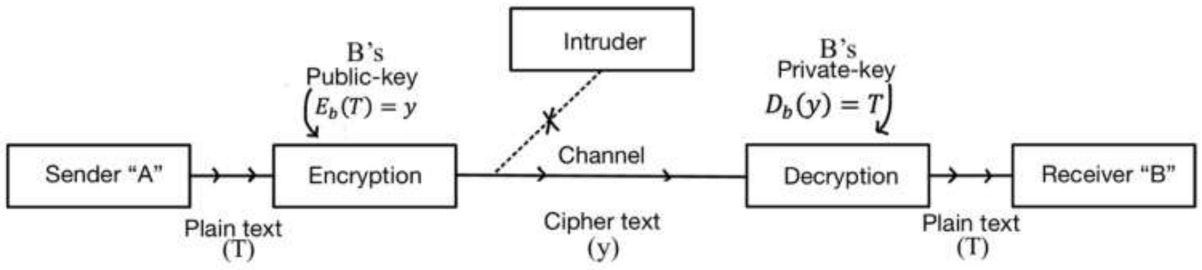

Figure 1. Diagram of the overall cryptosystem structure.

Considering the figure above, if "A" aims to send plain text (T) to "B" with a possible intruder in the network, "A" first retrieves the public key of "B" and inputs (T) into $E_b(x)$, outputting encrypted text (y) that is transferred to "B". Upon reception, "B" inputs (y) into $D_b(x)$. As $D_b(y) = D_b(E_b(T)) = T$. "B" has successfully acquired the initial text (T). Even if the intruder is successful in intercepting the cypher text (y), since $E_b(x)$ is a one-way function, neither the inverse function nor the corresponding input – plaintext (T) can be computed.

3. Algorithms of the cryptosystem

3.1. ElGamal key generation

Prior to using any encryption or decryption functions, users must first generate their own private and public keys. The design of the key generation algorithm is shown in the following:

"i" representing any sender generates a large prime "p" (of usually at least 100 digits); Randomly generate "a" $\epsilon\ z^+$; Randomly generate "d" $\epsilon\ z^+\ 2 \le d \le p-2$; Compute $b \equiv a^d \pmod p$, $2 \le b \le p - 1$; Public key = (a, b, p) and private key = d.

3.2. Primality tests for key generation. To generate a prime, the sender selects a large random positive integer and determines whether it is a prime using primality tests. If not, the user will repeat the process until a large prime is found. Since there are an infinite number of primes, large new ones are always possible to find. The two most commonly used primality tests will be shown in this section. Note that though such tests are probabilistic (giving the probability of an integer being a prime rather than a definitive answer), they are still the main algorithms used in real-life due to their efficiency [1]. To understand the theory of these algorithms, we first prove Fermat's little theorem .

$$\text{Lemma 1 Given a prime p, } \forall\, r \,\epsilon\, z^+, 1 \le r \le p-1, \text{ we have } p \mid {}^p_rC \tag{1}$$

Proof. ${}^p_rC = \frac{p(p-1)\ldots(p-r+1)}{r!} = n$ (an integer), we can deduce $r! \mid p(p-1)\ldots(p-r+1)$

Since $\gcd(p, r!) = 1$, then $r! \mid (p-1)\ldots(p-r+1)$, let $\frac{(p-1)(p-2)\ldots(p-r+1)}{r!} = k$. We now have $p \times k = {}^p_rC$. Here, it can be clearly seen that ${}^p_rC$ must be a multiple of p.

Theorem 1 (FLT) Given prime p, $\forall\, a \in z^+$, $a^p \equiv a \pmod p$

Proof by induction.

Base case: when $a = 1$, we have $1^p \equiv 1 \pmod p$, indicating that base case to be true.

Inductive hypothesis : suppose $a = k\ \epsilon\ z^+$, assume that FLT remains true: $(k)^p \equiv k \pmod p$

When $a = k + 1$, using the binomial theorem, we show that $(k+1)^p \equiv k + 1 \pmod p$:

$$(k+1)^p = k^p + {}^p_1C\,(k)^{p-1} + {}^p_2C\,(k)^{p-2} + \cdots + {}^p_{p-1}C\,(k) + 1 \tag{2}$$

Through lemma 1, p divides all coefficients ${}^{p}_{1}C, {}^{p}_{2}C, \dots, {}^{p}_{p-1}C$. Applying modular p, we obtain $(k+1)^p \equiv k^p + 1 \equiv k + 1$. By PMI, we have proved FLT. Note that if $\gcd(a, p) = 1$, then by dividing a on both sides, we have a special case of FLT: $a^{p-1} \equiv 1 \pmod p$.

3.2.1. Fermat's primality test. Note that while any prime satisfies $a^{p-1} \equiv 1 \pmod p$, there also exists special composite integers called Carmichael numbers that can also satisfy such equation [2]. Therefore, when an integer satisfies the FLT equation, its primality cannot be definitively determined. Yet, suppose $a^i \neq 1 \pmod i$ for coprime positive integers a and i, then i must be a composite number. The Fermat's primality test is based on such property and iterates itself through a range of values for a, $\gcd(a, i) = 1$ computing $a^{i-1} \pmod p$ [3] to provide accuracy.

Example 1 when $i = 7$ and we iterate value of a from 1 to 5:

$$1^{7-1} = 1 \equiv 1 \pmod 7$$

$$2^{7-1} = 64 \equiv 1 \pmod 7$$

$$3^{7-1} = 729 \equiv 1 \pmod 7$$

$$4^{7-1} = 4096 \equiv 1 \pmod 7$$

$$5^{7-1} = 15625 \equiv 1 \pmod 7$$

Here, the algorithm outputs that it is likley for 7 to be a prime as all congruences are 1.

Example 2 when $i = 4$ and we iterate value of a from 1 to 5:

$$1^{4-1} = 1 \equiv 1 \pmod 4$$

$$2^{4-1} = 8 \equiv 0 \pmod 4$$

$$3^{4-1} = 27 \equiv 3 \pmod 4$$

$$4^{4-1} = 64 \equiv 0 \pmod 4$$

$$5^{4-1} = 125 \equiv 1 \pmod 4$$

Here, the algorithm returns that 4 is definitely not a prime, as seen in the case where $a = 3$.

3.2.2. Miller-Rabin primality test. To understand the Miller-Rabin primality test, we first calculate the solutions of the equation is an odd prime.

$$x^2 \equiv 1 \pmod p, n \in z^+, p \qquad (3)$$

We rearrange to have $(x+1)(x-1) \equiv 0 \pmod p$. If p divides both $(x+1)(x-1)$, this would imply $p \mid (x+1) - (x-1) = 2$. Yet, $p \geq 3$ (a clear contradiction), therefore, p can only either divide $(x+1)$ or $(x-1)$. Thus, the solutions are $x + 1 \equiv 0 \pmod p \Longrightarrow x \equiv -1 \pmod p$ or $x - 1 \equiv 0 \pmod p \Longrightarrow x \equiv 1 \pmod p$. Using this property on FLT, if p is a prime, rearranging the equation $a^{p-1} \equiv 1 \pmod p$ into $(a^{\frac{p-1}{2}})^2 \equiv 1 \pmod p$. It can be deduced that $a^{\frac{p-1}{2}} \equiv -1 \pmod p$ or $a^{\frac{p-1}{2}} \equiv 1 \pmod p$. Now, let $x \in z^+$ s.t. $2^x \mid\mid p - 1 \Longrightarrow p - 1 = 2^x y$ and consider set "A" where an element is the square root of the successive element: $A = \{a^{p-1}, a^{\frac{p-1}{2}}, \dots, a^{\frac{(p-1)}{2^x}}\}$, for an integer p to be a prime, all elements in T must be congruent either to -1 or $1 \pmod p$.

Using this property, the Miller-Rabin algorithm calculates all congruences of set A elements under modulo p. If the property mentioned before is unsatisfied, we know p is definitely not be a prime. However, if this criterion is satisfied, p is only likely a prime because of composite Carmichael numbers.

Similar to Fermat's test, this algorithm repeats through a range of values to ensure accuracy, much like Fermat's test.

Example: Suppose $p = 9$. Let $a = 2$, set $T = \{a^8, a^4, a^2, a\}$

$$2^{9-1} = 2^8 = 256 \equiv 4 \pmod 9$$

$$2^{\frac{9-1}{2}} = 2^4 = 16 \equiv 7 \pmod 9$$

$$2^{\frac{9-1}{4}} = 2^2 = 4 \equiv 4 \pmod 9$$

$$2^{\frac{9-1}{8}} = 2^1 = 2 \equiv 2 \pmod 9$$

Here, the algorithm returns that 9 is not a prime.

Overall, being based on the straightforward concept of FLT, both probabilistic tests are simple in theory and algorithm design, making their implementation and execution in programs to be uncomplicated. In addition, both designs by only requiring modular exponentiation makes the algorithms to be computationally efficient in cases where large primes are involved compared to other primality algorithms. However, it should be noted that a high number of iterations is required to increase their accuracy and there should be a balance between accuracy and performance.

On the other hand, a major disadvantage of both algorithms is their probabilistic nature, arising from their inability to identify Carmichael numbers. To prevent false positives, both tests, may require numerous iterations and further verification using other primality tests such as the AKS test. Last but not least, both tests are inefficient for smaller primes compared to deterministic algorithms such as the trial division. However, this disadvantage does not impact upon the overall cryptosystem as only large primes are dealt.

3.3. ElGamal encryption and decryption functions

Prior to message encryption, all plaintext x must be translated into a numerical expression, and the conversion process must be known by both the sender and receiver. (e.g. letters' corresponding numerical value in the ASCII protocol). Knowing this, we proceed to the ElGamal encryption and decryption functions:

ElGamal Encryption algorithm [4] for message $M, 0 \leq M \leq p - 2$:

Randomly compute $k \in Z^+$ (kept private) $1 \leq k \leq p - 2$

$$\text{Compute } r \equiv a^k \pmod p \tag{4}$$

$$\text{Compute } t \equiv b^k \times M \pmod p \tag{5}$$

$$\text{Encrypted message sent to receiver is } (r, t) \tag{6}$$

$$\text{ElGamal Decryption algorithm [5] for } (r, t)\text{: } t \times r^{-d} \equiv M \pmod p \tag{7}$$

$$\text{This is true as } t \times r^{-d} \equiv b^k \times M \times \left(a^k\right)^{-d} \equiv \left(a^d\right)^k \times M \times \left(a^k\right)^{-d} \equiv M \pmod p \tag{8}$$

Validity for being a one-way function

We have proven the two algorithms to be inverses of each other. Now we proceed to analyze how the encryption function is a one-way function. Suppose an intruder was able to intercept the cipher message (r, t). To decrypt it, the value of d must be known. However, the only known relationship in which d is the only unknown variable is $a^d \equiv b \pmod p$. Implying $d \equiv \log_a(b) \pmod p$ [6]. However, the search space for such a discrete logarithm problem grows exponentially, making a brutal case bash infeasible and extremely computationally difficult. In addition, the use of an discrete logarithm in the encryption algorithm's design prevents exploitation of its structure to find a solution. In other words, this is indeed a one-way function [7].

3.4. ElGamal digital signature scheme

In the above section, it has been shown that the message is successfully encrypted. However, there are situations where an identity verification of the sender is also required, for example, in financial transactions to prevent fraud. In this case, the discrete logarithm cryptosystem allows any user to sign a message with their digital signature that the receiver can verify.

ElGamal digital signature signing [8] on user i' s message $M, 0 \leq M \leq p-2$:

The user i has public key (a, b, p) and private key d. k is the random number during encryption.

$$\text{Compute } r \equiv a^{k} \pmod{p}, 1 \leq r \leq p-1 \tag{7}$$

Compute k^{-1} (the modular inverse of k in modular p)

$$\text{Compute } y \equiv k^{-1}(M-dr) \pmod{p-1}, 0 \leq y \leq p-2 \tag{8}$$

The pair (r, y) is the signature that user i provides for the message M

ElGamal digital signature verification for the receiver of the message M:

Decrypt received message to acquire plaintext message M

Verify if $b^{r} \times r^{y} \equiv a^{M} \pmod{p}$. If yes, then the message indeed comes from i

Here, we can see that

$$\left(a^{d}\right)^{r} \times \left(a^{k}\right)^{(M-dr)k^{-1}} \equiv a^{M-dr+dr} \equiv a^{M} \pmod{p} \tag{9}$$

Proof for being a non-replicable signature by other users

If an intruder wants to mimic the digital signature of the user i for any message m, then the intruder must construct (r, y) s.t. $b^{r} \times r^{y} \equiv a^{m} \pmod{p}$. In this expression, a, b, are known and r can also be constructed as the intruder can randomly select the value of k himself. However, to compute y, the intruder cannot use $(M-dr)k^{-1}$ as he or she does not know the private key d. The only other method of calculating y is through the equation $y = \log(\frac{a^{m}}{b^{r}}) \pmod{p}$. This is infeasible with the computational difficulty described in Section 3.3.

However, when evaluating the digital signature algorithm's design, there are 2 major safety flaws. Before analyzing these flaws, the linear congruency theory will be explained to understand it.

Suppose $ax \equiv b \pmod{c}$, where a, b are given positive integers smaller than c and x is an integer variable smaller than c. Let $\gcd(a, c) = D$.

Theorem 2 There only exists a solution for x if $D \mid b$

Proof. From $ax \equiv b \pmod{c}$, $ax = c(K) + b$, where $K \in z^{+}$ $\because D \mid a$ and $D \mid c$ $\therefore D \mid b$. However, if $D \nmid b$ for the given value. Then no solution exists.

Theorem 3 There is a unique solution if $\gcd(a, c) = D = 1$.

Proof. For this theorem, we will prove both the existence and uniqueness of a solution Existence: As $\gcd(a, c) = 1$, by Bezout's identity, $\exists\ x_0, y$ s.t. $ax_0 + cy = 1$. We multiply both sides by b to have $a(bx_0) + cby = b$. Applying modular c, then $a(bx_0) \equiv b \pmod{c}$. Thus bx_0 is a solution. However, if bx_0 is not smaller than c, then let $bx_0 = (A)c + r$. Where $A \in z^{+}$ and $r < c$. Then, $b \equiv a(Ac + r) \equiv (aA)c + ar \equiv ar \pmod{c}$. Therefore, r is a solution to $ax \equiv b \pmod{c}$ that is smaller than c.

Uniqueness: FTSOC, suppose $\exists$ solutions r and $x_1 < c$, $r \neq x_1$,to $ax \equiv b \pmod{c}$. Then $b \equiv ax_1 \equiv ar \pmod{c}$. Since $(a, c) = 1$, we have $x_1 \equiv r \pmod{c}$. This a contradiction as both are different and smaller than c. Thus, r is the only unique solution smaller than c.

Corollary 1. If $ax \equiv b \pmod{c}$, where $\gcd(a, c) = D$, then there only exist D distinct solutions, all of which are smaller than c, to this linear congruence equation.

Proof. Let $a = dm, c = dn$, note that $\gcd(m, n) = 1$. From theorem 3, there will be a unique solution smaller than n to $\frac{a}{d}(x) \equiv \frac{b}{d} \pmod{\frac{c}{d}}$ as $\gcd\left(\frac{ax}{d}, \frac{c}{d}\right) = 1$, this can be also expressed in the form $mx \equiv \frac{b}{d} \pmod{n}$. Let this solution be s. Consider $s, s + \frac{n}{d}, \ldots, s + (d-1)n$. These values are all smaller than c as $s < n$. Thus the $s + (d-1)n < dn$. These D distinct solutions will also satisfy the original equation as $ax = dm(s + in) \equiv dms \equiv b \pmod{c = dn}$ for $1 \leq i \leq d - 1$. There will not be any more solution as they are in between the intervals of these solutions and thus will not be congruent to b modulo c.

Coming back to analyzing the flaws of this digital signature scheme, the sender cannot use the same randomly selected value k, for two different messages M_1, M_2 as the receiver can then compute the sender's private key d by the following process: Suppose the user i has given two digital signatures, where having the same k value, and thus a. We have the following equations:

$$y_1 \equiv (M_1 - dr)k^{-1} \pmod{p-1} \tag{10}$$

$$y_2 \equiv (M_2 - dr)k^{-1} \pmod{p-1} \tag{11}$$

This forms the simultaneous equation : $k(y_1 - y_2) \equiv M_1 - M_2 \pmod{p-1}$

Let $\gcd(y_1 - y_2, p - 1) = D$. From corollary 1, there are exactly D number of distinct solutions for k in this equation. The actual value of k used in the digital signature algorithm by the user i out of all D number of possible values can be checked by the equation $a^k \equiv r \pmod{p}$ if $\gcd(r, p-1) = 1$. The probability of $r, p-1$ being coprime is $(1 - \frac{\varphi(p-1)}{p-1}) \approx 1$. Once the value of k has been calculated, one can find d by $d \equiv (M_1 - y_1 k)r^{-1} \pmod{p-1}$. Thus, the private key of the user i is compromised. Such a flaw is fundamentally due to the algorithms' reliance on a random number generator for producing the ephemeral (single use) key during the signing process. If the random number generator is predictable or if, at any moment, the key k is reused, it will lead to a total loss of security.

Secondly, the ElGamal signature in real-life produces large signatures that are often twice the size of the modulus used. As this requires greater storage requirements, it is not suitable in situations where communication space and buffers are limited or should be minimized. Thirdly, similar to this flaw, the signature algorithm is computationally inefficient due to its use of modular exponentials for both signing and verification. Being slow and resource intensive, this is a significant drawback for common devices, which have limited computational power. Fourthly, the algorithm is designed in such a way that its security is significantly influenced by the proper selection of parameters, where weak parameters can lead to security vulnerabilities.

3.5. *Multi-party secrets*

In cryptography terms, secret sharing refers to the process of either sharing a private key or "splitting" it into many subkeys that are then distributed to multiple parties. The goal of secret sharing is to avoid any potential abuse of authority. or to be used in situations where a consensus is required. Both private key sharing and splitting will be explored in this section through the Diffie-Helman's key exchange algorithm and the key splitting algorithm respectively.

3.5.1. Diffie-Helman key exchange. The Diffie-Helman key exchange is an algorithm that allows two parties to securely establish a shared secret over an insecure communication channel [9]. The shared secret can be used in symmetric cryptosystems such as the AES (Advanced Encryption Standard) or potentially message authentication. The exact process is provided below:

User X and user Y agree upon the same modulo p (a prime), and generator g.

Let the private key of users X and Y be denoted as d_X and d_Y.

Compute the public key $b_X \equiv g^{d_X} \pmod{p}$ and $b_Y \equiv g^{d_Y} \pmod{p}$.

Both public keys are shared with all other users through the insecure channel.

X and Y use other's public key and their own private key to compute their shared secret:

Shared secret of X: $S_X \equiv b_Y^{d_X} \equiv g^{(d_X)(d_Y)} \pmod{p}$

Shared secret of Y: $S_Y \equiv b_X^{d_Y} \equiv g^{(d_X)(d_Y)} \pmod{p}$

Note that in this process, the private keys of X and Y has not been leaked as the calculation of them in $b_X \equiv g^{d_X} \pmod{p}$ and $b_Y \equiv g^{d_Y} \pmod{p}$ would again be associated with the difficulty of discrete logarithm problems. Furthermore, despite having an insecure channel, both users were still able to compute a shared secret.

Analyzing the design of this algorithm, the use of a generator is advantageous as it contributes to the security of the difficulty of discrete logarithm problem [10]. By generating a number of distinct elements under modulo p, it will be more difficult for a system, intruder to compute the discrete logarithm. In addition, the use of a generator ensures that the key space is sufficiently large to provide adequate security. As the generator produces congruences that cycle through all integers of modulo p, this increases the difficulty of brute-force attacks.

3.5.2. Key splitting algorithm. The key splitting algorithm is a process where an original key is split into many subkeys to be distributed to multiple parties. These subkeys cannot decrypt information or allow any individual party to access any private information. Only when all parties come together, where all subkeys are shared, can they reconstruct the original private key with complete information. The key splitting algorithm is described in the following:

The server randomly generates integers $C_{j-1}, C_{j-2}, \dots, C_1$ to create the function $F(x)$:

$$F(x) = C_{j-1}x^{j-1} + C_{j-2}x^{j-2} + \cdots + C_1x^1 + C_0 \tag{12}$$

$C_{j-1} \neq 0$ and original private key $= C_0 = F(0)$

Algorithm randomly generates $u_1, u_2, \dots, u_n \neq 0 \in Z^+$ and computes $F(u_1), F(u_2) \dots F(u_n)$, the pair $(i, F(a_i))$ is a subkey that is distributed to person i for $1 \leq i \leq n$

To understand how the original private key can be constructed with all subkeys, we first explore the Lagrange interpolation explained below:

Theorem 4. For n pairs of coordinates $(a_1, b_1), \dots, (a_n, b_n)$ the f(x) below is the only polynomial of degree $\leq n - 1$ that goes through of all n points :

$$f(x) = \sum_{i=1}^{n} \left(b_i \times \prod_{i \neq j}^{n} \frac{(x - a_j)}{(a_i - a_j)} \right) \tag{13}$$

$$= b_1 \frac{(x-a_2)(x-a_3)\dots(x-a_n)}{(a_1-a_2)(a_1-a_3)\dots(a_1-a_n)} + b_2 \frac{(x-a_1)(x-a_3)\dots(x-a_n)}{(a_2-a_1)(a_2-a_3)\dots(a_2-a_n)} + \cdots + b_n \frac{(x-a_1)(x-a_2)\dots(x-a_{n-1})}{(a_n-a_1)(a_n-a_2)\dots(a_n-a_{n-1})} \tag{14}$$

Proof: In each section $f_i(x) = b_i \frac{(x-a_1)\dots(x-a_{i-1})(x-a_{i+1})\dots(x-a_n)}{(x_i-a_1)\dots(x_i-a_{i-1})(x_i-a_{i+1})\dots(x_i-a_n)}$, where the numerator has degree $\leq n - 1$ and denominator has degree of 0. This implies that $\deg(f_i(x)) \leq n - 1$. Thus the sum of all $f_i(x)$ which is f(x) will also have degree $\leq n - 1$. To prove f(x)'s uniqueness: FTSOC, let there be two polynomials f(x) and g(x), both of degree smaller than $n - 1$, that goes through every point. Consider a new function $h(x) = f(x) - g(x)$. As $f(x) = g(x)$ at x coordinates $a_1, a_2, \dots, a_n$, h(x) which has degree of at most $n - 1$ is equal to zero n times, and thus has n roots. Then deg (h) must be $\geq n$. This is a contradiction. We now prove that the polynomial goes through all n coordinates (a_i, b_i), $1 \leq i \leq n$. When $x = a_i$, $f_i(a_i) = b_i \frac{(a_i-a_1)\dots(a_i-a_{i-1})(a_i-a_{i+1})\dots(a_i-a_n)}{(a_i-a_1)\dots(a_i-a_{i-1})(a_i-a_{i+1})\dots(a_i-a_n)} = b_i$, and other fractions equal to 0 as their numerators have a root $(x - a_i)$. We conclude that for $1 \leq i \leq n$, $f(a_i) = b_i$. In other words, when all parties share their subkeys, the original function F(x) can be calculated using theorem 4 and finds its y intercept F(0) – the original private key [11].

Now, we shown the key threshold that even if $n - 1$ parties share their $n - 1$ subkeys with each other, the function F(x) cannot be determined and the original private key remains secret:

Proof. WLOG assume $(a_{1,} f(a_1))$, $(a_2, f(a_2)) \dots (a_{n-1}, f(a_{n-1}))$ is known. Without further information, a_n and $f(a_n)$ has equal possibility being any integer and there exists a corresponding polynomial of degree $\leq n-1$ for each possibility based on theorem 4. Thus, the $n-1$ parties can never calculate the original $F(x)$ to determine the original private key.

4. Evaluation of the overall cryptosystem

4.1. Advantages

The discrete logarithm cryptosystem's ability to provide robust and reliable security for users against network intruders arises from the discrete logarithm problem's computational difficulty, which, under the current computational power of classical computers, is used as its one-way function. Additionally, the safety is not limited by known mathematical shortcuts or algorithms, as the larger the primes used in encryption, the higher the security of the overall encryption. In other words, by adjusting the parameters, a network can always achieve the desired level of security.

Secondly, the simplicity of each algorithm design in the cryptosystem allows them to be easily implemented in various programming languages as the. underlying mathematical operations are simple. Thus, informatics specialists can focus on designing secure systems without being drawback by complex cryptographic primitives. Ultimately, the algorithms' design has resulted in the cryptosystem having various applications [12].

Thirdly, the design of the discrete logarithm algorithms allows them to be flexible and offer multiple variants, making the cryptosystem adaptable to different use cases and requirements. One important variant is the elliptic curve digital signature algorithm (ECDSA), which is based on the elliptic curve variant of the discrete logarithm problem and is used as the basis for modern cryptocurrency transfers.

Finally, DLC's nature as an asymmetric cryptosystem that has a public and private key offers efficient key management for users. On the other hand, in symmetric cryptosystems, multiple secret keys must be securely exchanged and renewed between parties. This efficient key management simplifies the process of establishing secure communications and contributes to the scalability of these systems.

4.2. Disadvantages

A significant disadvantage that arises from the encryption algorithm's design is the large prime numbers that must be generated to ensure a high level of security. As the key size increases, so does the computational power, memory, and bandwidth consumption. As a result, devices with limited resources, such as mobile phones, are less suitable for the use of cryptosystems.

In addition, another limiting factor is the primality algorithms' design, where numerous iterations are made for an accurate test. The extensive computations and consequently high processing demand naturally slow down the entire process. In other words, this drawback was brought about by the lack of a quick primality test in its design. Thus, DLC is not the first option when it comes to encrypting large data files, such as election poles. Similar to this, the extensive prime number calculations required for decryption and cypher text transfer also lowers the overall efficiency of the system. As a result, additional study in cryptography may be conducted on potential improvements to the algorithms for decryption and primality testing [13].

Additionally, as the DLC can be attacked through quantum computing, it may no longer be applicable in the future. Also, the use of a discrete logarithm one-way function fully remains under the presumption that the secret key cannot be computed by outside intruders. However, in practice, it just takes an extensive long time to compute the large prime factors. Thus, as quantum computing advances, potential quantum assaults may be able to exploit this weakness. Currently, IBM Quantum has suggested the Shor's method, which theoretically has a high efficiency in factoring huge numbers [14].

5. Conclusion

Overall, this paper has investigated algorithms such as primality tests, ElGamal key generation, encryption, decryption, splitting, and sharing algorithms. Based on the knowledge of such algorithms, it was concluded that the advantage of the cryptosystem's design is primarily attributed to the use of discrete logarithms, which provides security, simplicity, and flexibility. On the other hand, the disadvantage of such a system mainly arises from its reliance on large prime numbers, which increase the need of computational power and memory. Reflecting on this evaluation, the derivation for time complexity of each algorithm, which is important in the process of cryptanalysis, could be included. Also, inclusion of elliptical curve variant algorithms may be considered for this paper to establish a wider scope of understanding. Last but not least, other suggested quantum encryption algorithms may also be explored, as they are the cryptography of the future.

References

[1] Graduate cryptography [Lecture notes]. (n.d.). https://inst.eecs.berkeley.edu/~cs276/fa20/
[2] Primality proving [Lecture notes]. (2017, March 20). MIT math.
[3] Agrawal, M. (n.d.). Primality Tests Based on Fermat's Little Theorem [Lecture notes]. Retrieved April 18, 2023.
[4] Will, M. A., & Ko, R. K. (2015). A guide to homomorphic encryption [Afterword]. In The cloud security Ecosystem.
[5] Qin, F. K. (n.d.). Number Theory and Cryptography (J. B. Hou, Ed.) [PDF]. Bejing Science. http://www.ecsponline.com/yz/BFD922C8D8A5441DD86678D7A1C955658000.pdf
[6] The discrete logarithm problem - 18.783 elliptic curves [Lecture notes]. (2022, February 28). MIT Math.
[7] Vaikuntanathan, V. (2022, June 1). MIT 6.5620/6.875/18.425 foundations of cryptography [Lecture notes]. mit6875. http://mit6875.org/
[8] Galbraith, S. (2012). Mathematics of public key cryptography. Cambridge University Press. https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html
[9] Joux, A., Odlyzko, A., & Pierrot, C. (2014). The past, evolving present and future of discrete logarithm. University of Minnesota.
[10] Lynn, B. (n.d.). Generators [Fact sheet]. Stanford Cryptography. Retrieved March 5, 2023, from https://crypto.stanford.edu/pbc/notes/numbertheory/gen.html
[11] Boneh, D., Boyen, X., & Halevi, S. (n.d.). Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles. Stanford Cryptography.
[12] Odlyzko, A. M. (n.d.). Discrete logarithms in finite fields and their cryptographic significance. University of Minnesota. Retrieved November 4, 2022.
[13] Shor, P. W. (1996, January). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer [Lecture notes].
[14] Singh, S. (n.d.). The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography (2nd ed.). Anchor.