On Two Novel Generalized Versions of Diffie-Hellman Key Exchange Algorithm Based on Neutrosophic and Split-Complex Integers and their Complexity Analysis

Dima Alrwashdeh[1,*], Talat Alkhouli[2], Ahmed Soiman Rashed Alhawiti[3], Ali Allouf[4], Hussein Edduweh[5], Abdallah Al-Husban[6]

[1]Department of Information Technology, School of Information Technology and System, the University of Jordan, Aqaba, Jordan

[2]Applied Science Department, Aqaba University College, Al-Balqa Applied University, Jordan

[3]Department of General Studies, Technical College of Haql, Tabuk, Kingdom of Saudi Arabia

[4]Tishreen University, Faculty Of computer engineering and automation, Latakia, Syria

[5]Department of Mathematics, the University of Texas at Arlington, Arlington, TX 76019-0407, USA

[6]Department of Mathematics, Faculty of Science and Technology, Irbid National University, P.O. Box: 2600 Irbid, Jordan

Emails: d.rawashdeh@ju.edu.jo; Talat.khouli@bau.edu.jo; ahmed.a13@tvtc.gov.sa; Ali.allouf@gmail.com; Husseinsaid.edduweh@mavs.uta.edu; dralhosban@inu.edu.jo

Abstract

The objective of this paper is to build the Split-Complex version of Diffie-Hellman key Exchange Algorithm, where we use the mathematical foundations of Split-Complex Number Theory and Integers, such as congruencies, raising a split-complex integer to a power of split-complex integer to build novel algorithms for key Exchange depending of famous Diffie-Hellman algorithm. Additionally, we present the proposed version of the Diffie-Hellman algorithm based on neutrosophic number theory. Also, we analyze the complexity of the novel algorithms with many examples that explain their applied validity.

Keywords: Split-Complex Cryptography; Split-Complex Diffie-Hellman; Hellman key Exchange Algorithms; Neutrosophic Diffie-Hellman

1. Introduction

Numerous applications of integer extension fields have recently emerged, particularly in cryptographic algorithms. Modern methods and proposed algorithms rely on enhancing the complexity of existing security strategies by utilizing neutrosophic number theory and Split-Complex number theory [1, 2, 5, 7]. In 2023, the Split-Complex Number Theory was born, where Merkepci and Abobala introduced the mathematical concepts and algebraic structures for developing a public-key encryption algorithm, specifically RSA, utilizing split-complex number theory [2].

Since the advent of Shannon's mathematical theory of communication and the subsequent evolution of digital systems, the paramount concern has been safeguarding the security of information transmitted through communication channels, protecting it from tampering and eavesdropping. Consequently, the emergence of robust encryption algorithms became imperative to shield such information. All encryption algorithms, whether symmetric or asymmetric, rely on keys for generating cipher text. Securely generating and transmitting session keys has always been a fundamental challenge. In 1976, researchers Diffie and Hellman proposed their renowned

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

key exchange algorithm [10-14]. In [2] Merkepci and Abobala suggested for the first time the idea of using Split-Complex number theory in cryptography, and in [6-9] the applications of neutrosophic number theory in generalizing classical crypto-algorithms were studied in details.

The Diffie-Hellman key exchange is a foundational cryptographic method that allows two parties to securely agree on a shared secret key, even if they communicate over an insecure channel where others might be listening. This shared secret key can then be used to encrypt and decrypt messages, providing confidentiality for their communication. The magic of Diffie-Hellman lies in modular arithmetic and one-way functions:

a) Shared Public Parameters: Two parties, Alice and Bob, begin by agreeing on a large prime number (p) and a generator (g) within a finite field. These are public values and can be known by anyone.

b) Private Keys: Alice and Bob each choose a secret, random number. Alice's is called a, and Bob's is called b. These are kept absolutely private.

c) public Key Calculation:

- Alice calculates: $A = g^a \; mod \; p$, and sends the result (A) to Bob.
- Bob calculates: $B = g^b \; mod \; p$, and sends the result (B) to Alice.

d) Shared Secret Calculation:

- Alice receives B and computes $B^a (mod \; p)$.
- Bob receives A and computes $A^b (mod \; p)$.

Crucially, due to the properties of modular arithmetic, both Alice and Bob will arrive at the same shared secret value

2. Main discussion

In this section, we will elucidate the rationale behind our selection of positive neutrosophic integers as the foundation for the novel proposed algorithm. The neutrosophic integer ring (I) finds applications in cryptography due to the inherent difficulty of splitting neutrosophic positive integers. Neutrosophic integer rings make cryptographic systems more complex because breaking down these special whole numbers is a tougher problem.

- **Remark** [11]

a) Let $a + bI$, $c + dI$ be two neutrosophic integers, then:

$a + bI \leq c + dI$ if and only if $a \leq$ c, $a + b \leq c + d$.

b) $a + bI$ is called positive neutrosophic integer if $a > 0$ and $a + b > 0$.

2.1. Proposed neutrosophic algorithm

The Description of neutrosophic Diffie-Hellman Algorithm:

a) Alice and Bob agree on a neutrosophic prime $p = p_1 + p_2 I$, i.e. p_1, $p_1 + p_2$ are classical primes and a base $g = g_1 + g_2 I > 0$, i.e. g_1, $g_1 + g_2 > 0$.

b) Alice chose a secret number $\boldsymbol{a = a_1 + a_2 I > 0}$, and sends Bob $\boldsymbol{g^a (mod\; p)}$.

[Remark that $\boldsymbol{g^a (mod\; p) = {g^a}_1 (mod\; p_1) + I[\left((g_1 + g_2)^{a_1+a_2} (mod\; p_1 + p_2) - {g^a}_1 (mod\; p_1)\right]}$.

c) Bob choose a secret number

d) Alice computes:

$$\boldsymbol{(g^b)^a (mod\; p) = {g^{a_1 b_1}}_1 (mod\; p) + I[\left((g_1 + g_2)^{(a_1+a_2)(b_1+b_2)} (mod\; p_1 + p_2) - {g^{a_1 b_1}}_1 (mod p_1)\right]}.$$

e) Bob computes:

$$\boldsymbol{(g^a)^b (mod\; p)}.$$

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

- **Example**

Assume that Alice and Bob have agreed on $\boldsymbol{p = 3 + 4I}$, and $\boldsymbol{g = 5 + I}$.

Alice chooses the secret neutrosophic number $\boldsymbol{a = 2 + 3I}$. Alice sends Bob$\boldsymbol{(5 + I)^{2+3I}(mod\ p) = 5^2(mod\ 3) + I\left[6^5(mod\ 7) - 5^2(mod\ 3)\right] = 1 + 5I}$.

Bob chooses the secret neutrosophic number $\boldsymbol{b = 4 - 2I}$, and sendsAlice$\boldsymbol{(5 + I)^{4-2I}(mod\ p) = 5^4(mod\ 3) + I[6^2(mod\ 7) - 5^4(mod\ 3)] = 1}$.

Alice computes $\boldsymbol{1^{2+3I}(mod\ p) = 1}$.

Bob computes $\boldsymbol{(1 + 5I)^{4-2I}(mod\ p) = 1}$.

Thus, we observe that both parties generated the same secret key value, and therefore the neutrosophic algorithm works correctly.

- **Results**

I. As we can see, the neutrosophic Diffie-Hellman algorithm involves more computational steps and operations compared to the traditional Diffie-Hellman algorithm. While the overall complexity remains $O((log\ n)^\wedge 3)$, the neutrosophic version has a larger constant factor due to the additional modular exponentiations and subtractions required to handle the neutrosophic parameters.

II. The neutrosophic Diffie-Hellman algorithm is a more complex version of the traditional Diffie-Hellman algorithm. It offers potential security advantages but comes with a higher computational cost. The choice between the two algorithms depends on the specific security requirements and computational resources available.

2.2. Complexity Analysis compared to the classical version

Now, We will compare Diffe- Hellman and neutrosophic Diffe- Hellman by the duration needed to be broken by using brute-force: (All are measured in seconds in the table bellow):

Table 1: Compareson between Diffe- Hellman and neutrosophic Diffe- Hellman by the duration

Classical Diffe-Hellman	**Duration**	**Neutrosophic Diffe- Hellman**	**Duration**
For 12 bit prime number p	**0.0009770393371582031**	For 12 bit primes numbers p1 and p2	**0.0019540786743164062**
For 18 bit prime number p	**0.001410508155822754**	For 18 bit primes numbers p1 and p2	**0.002821016311645508**
For 24 bit prime number p	**0.009863948822021484**	For 24 bit primes numbers p1 and p2	**0.019727897644042968**

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

We can see that the neutrosophic version of Diffe- Hellman needs more time to be broken, and its complexity is around. The complexity of the brute force attack is O(p1 * p2), where p1 and p2 are the prime numbers used in the Neutrosophic Diffie-Hellman key exchange.

The reason for this complexity is that the attack tries all possible combinations of private keys (a1, a2) to find the correct one that matches the shared secret. The nested loop in the code iterates over the ranges (1, p1) and (1, p2) for a1 and a2, respectively.

The worst-case scenario occurs when the correct private key is the last combination to be tried, which would require (p1 - 1) * (p2 - 1) iterations. Therefore, the time complexity of the attack is proportional to the product of p1 and p2.

2.3. Side Channel attacks and proposed use of the Neutrosophic algorithm

Side-channel and fault attacks are serious threats to the security of cryptographic implementations, particularly in hardware devices like smartcards and embedded systems. These attacks exploit physical characteristics or unintended behavior of the hardware during the execution of cryptographic algorithms, allowing an attacker to potentially recover sensitive information or cryptographic keys. [4]

Side-channel attacks are based on the analysis of physical effects, such as timing information, power consumption, electromagnetic emanations, or cache behavior, which can leak information about the internal state of the cryptographic operations. By carefully measuring and analyzing these physical effects, an attacker may be able to deduce information about the secret keys or intermediate values used in the cryptographic computations.

Fault attacks, on the other hand, involve introducing faults or errors into the cryptographic computations, either by exposing the device to external factors like glitches, voltage spikes, or electromagnetic pulses, or by exploiting hardware vulnerabilities. These faults can cause the device to behave in an unintended manner, potentially revealing sensitive information or allowing the attacker to bypass security mechanisms.

We believe that The Neutrosophic Diffie-Hellman (NDH) key exchange protocol, which is an extension of the classical Diffie-Hellman protocol, can provide some resistance against side-channel attacks, but it does not completely eliminate the risk.

- **Complex arithmetic**: NDH introduces complex number arithmetic into the key exchange process. Instead of working with regular modular arithmetic, it operates on complex numbers modulo a complex modulus. This increased complexity can make it more difficult for an attacker to deduce information from side-channel leakages, as the operations involve both real and imaginary components.
- **Key randomization**: In NDH, the private keys used by Alice and Bob are complex numbers, with both real and imaginary parts. This introduces an additional layer of randomness compared to the classical Diffie-Hellman protocol, where the private keys are real numbers. The increased randomness can help obfuscate the side-channel information, making it harder for an attacker to interpret the leakages.
- **Increased computational complexity**: The complex arithmetic operations in NDH are generally more computationally intensive than the regular modular arithmetic used in classical Diffie-Hellman. This increased computational complexity can make it more challenging for an attacker to correlate the side-channel leakages with specific operations or intermediate values.

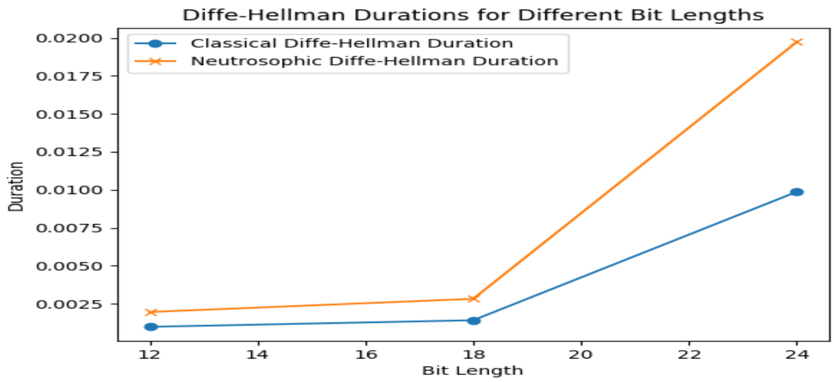
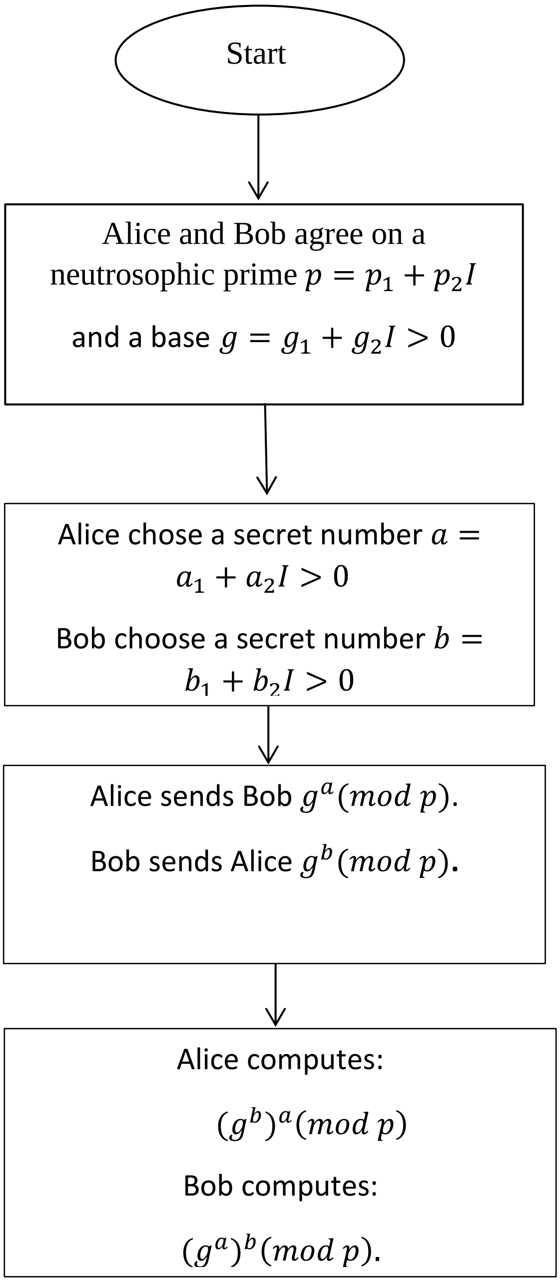


Figure 1. Diffe-Hellman duration for different Bit lengths

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

2.4. The flow chart of the Neutrosophic DH:



2.5. Split-Complex Version of DH Algorithm

We recall some basic concepts in Split-Complex Number Theory and Integers.

- **Definition.** [2]

Let $x = p + qj$ and $y = c + dj$ be two split-complex integers, where:

p and c are real numbers.

q and d are coefficients of the split-complex unit j, which satisfies $j^2 = 1$.

We say x divides y (denoted $x \mid y$), if there exists another split-complex integer $z = m + nj$ such that:

y is equal to the product of x and z: $y = x \times z$.

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

- **Definition.** [2]

Let $x = p + qj, y = c + dj, = m + nj$ be three split-complex integers, then:

I. $x \equiv y(mod\ z)\ if\ and\ only\ if\ z|x - y.$
II. $x \leq y\ if\ and\ only\ if\ p - q \leq c - d\ and\ p + q \leq c + d.$

- **Remark**

We will define the specific rules for calculating this operation, allowing us to work with complex expressions involving split-complex numbers and exponents.

$$(a + bj)^{(c+dj)} = \frac{1}{2}[(a - b)^{c-d} + (c + d)^{c+d}] + \frac{1}{2}j[(a + b)^{c+d} - (a - b)^{c-d}].$$

- **Remark**

The Diffie-Hellman (DH) key exchange algorithm is a cornerstone of modern cryptography, playing a crucial role in secure communication over insecure channels. Its importance stems from several key factors such as: Enabling Secure Key Exchange, Foundation for Secure Protocols, and Forward Secrecy.

However, the classical DH algorithm also has limitations, prompting the need for enhancements: Vulnerability to Man-in-the-Middle Attacks, Computational Complexity, and Quantum Computing Threat.

3. The proposed Diffie-Hellman based on Split-Complex Number Theory

The Description of Split-Complex Diffie-Hellman Algorithm:

a) Alice and Bob agree on a Split-Complex prime $p = p_1 + p_2 j$, (it is preferred to take $p_1 - p_2, p_1 + p_2$, as large prime numbers and a base $g = g_1 + g_2 j$.

b) Alice chooses a secret number $a = a_1 + a_2 j$, and sends Bob $g^a(mod\ p)$. Remark that:

$$g^a = \frac{1}{2}[(g_1 - g_2)^{a_1 - a_2} + (g_1 + g_2)^{a_1 + a_2}] + \frac{1}{2}j[(g_1 + g_2)^{a_1 + a_2} - (g_1 - g_2)^{a_1 - a_2}].$$

c) Bob chooses a secret number $\boldsymbol{b = b_1 + b_2 j > 0}$, and sends Alice $\boldsymbol{g^b(mod\ p)}$.

d) Alice computes:

$$\boldsymbol{(g^b)^a(mod\ p)}$$

e) Bob computes:

$$\boldsymbol{(g^a)^b(mod\ p)}.$$

- **Example**

We Assume that Alice and Bob have agreed on $\boldsymbol{p = 5 + 3j}$, and $\boldsymbol{g = 5 + j}$. Alice choose the secret Split-Complex number $\boldsymbol{a = 3 + 2j}$. And Bob Bob choose the secret neutrosophic number $\boldsymbol{b = 4 - 2j}$.

Alice sends Bob: $\boldsymbol{(5 + j)^{3+2j}(mod\ 5 + 3j) = \frac{1}{2}[(5 - 1)^{3-2} + (5 + 1)^{3+2}] + \frac{1}{2}j[(5 + 1)^{3+2} - (5 - 1)^{3-2} =}$

$\boldsymbol{(3890 + 3886j\ )(mod\ 5 + 3j) = \frac{1}{2}[(3890 + 3886)(mod\ 8) + (3890 - 3886)(mod\ 8) + \frac{1}{2}j[(3890 + 3886)(mod\ 8) - (3890 - 3886)(mod\ 2)] = 0}.$

Bob sends Alice: $\boldsymbol{(5 + j)^{4-2j}(mod\ 5 + 3j) = 2 + 2j}.$

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

Alice computes: $\mathbf{0}^{3+2j}(\boldsymbol{mod}\ \mathbf{5}+\mathbf{3}\boldsymbol{j})=\mathbf{0}$.

Bob computes: $(\mathbf{2}+\mathbf{2}\boldsymbol{j})^{4-2j}(\boldsymbol{mod}\ \mathbf{5}+\mathbf{3}\boldsymbol{j})=\mathbf{0}$.

Thus, we observe that both parties generated the same secret key value, and therefore the neutrosophic algorithm worked correctly.

- **Results**

The Split-Complex Diffie-Hellman algorithm introduces additional computational complexity compared to the traditional Diffie-Hellman algorithm, while potentially offering an additional layer of security due to the use of split-complex numbers. The trade-off between security and computational overhead should be carefully considered before adopting this approach in practical applications.

- **Remark**

The codes which have been used:

```
import random

import time

# Function to check if a number is prime

def is_prime(n):

    if n <= 1:

        return False

    if n <= 3:

        return True

    if n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i * i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

# Function to compute modular exponentiation (base^exp mod mod)

def mod_exp(base, exp, mod):

    result = 1

    base = base % mod

    while exp > 0:

        if exp % 2 == 1:

            result = (result * base) % mod

        exp = exp // 2

        base = (base * base) % mod

    return result

# Function to perform complex modular exponentiation (base^exp mod mod)

def complex_mod_exp(base, exp, mod):
```


DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

```
    real = mod_exp(base.real, exp.real, mod.real)
    imag = mod_exp(base.imag, exp.imag, mod.imag)
    return complex(real, imag)
# Neutrosophic Diffie-Hellman key exchange
def neutrosophic_diffie_hellman(p1, p2, g1, g2, a1, a2, b1, b2):
    # Check if p1 and p2 are prime
    if not is_prime(p1) or not is_prime(p2):
        raise ValueError("p1 and p2 must be prime numbers.")
    # Compute p = p1 + p2i
    p = complex(p1, p2)
    # Check if g1 and g2 are positive
    if g1 <= 0 or g2 <= 0:
        raise ValueError("g1 and g2 must be positive numbers.")
    # Compute g = g1 + g2i
    g = complex(g1, g2)
    # Compute g^a (mod p)
    ga_mod_p = complex_mod_exp(g, complex(a1, a2), p)
    # Compute g^b (mod p)
    gb_mod_p = complex_mod_exp(g, complex(b1, b2), p)
    # Compute (g^b)^a (mod p)
    gab_mod_p = complex_mod_exp(gb_mod_p, complex(a1, a2), p)
    # Compute (g^a)^b (mod p)
    gba_mod_p = complex_mod_exp(ga_mod_p, complex(b1, b2), p)
    return gab_mod_p, gba_mod_p
# Example usage
p1 = 163  # Classical prime p1
p2 = 59  # Classical prime p2
g1 = 2   # Base g1
g2 = 61   # Base g2
# Generate 200 key pairs and test brute force attack
for _ in range(100):
    # Generate random private keys
    a1 = random.randint(1, p1 - 1)  # Secret number a1
    a2 = random.randint(1, p2 - 1)  # Secret number a2
    b1 = random.randint(1, p1 - 1)  # Secret number b1
    b2 = random.randint(1, p2 - 1)  # Secret number b2
    # Calculate the shared secret
```


DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

```
    alice_result, bob_result = neutrosophic_diffie_hellman(p1, p2, g1, g2, a1, a2, b1, b2)
    # Perform brute force attack
    start_time = time.time()
    found_key = False
    for i in range(1, p1):
        for j in range(1, p2):
            # Calculate the shared secret using the brute forced private key
            test_alice_result, test_bob_result = neutrosophic_diffie_hellman(p1, p2, g1, g2, i, j, b1, b2)
            # Check if the calculated shared secret matches the original shared secret
            if test_alice_result == alice_result and test_bob_result == bob_result:
                found_key = True
                end_time = time.time()
                time_taken = end_time - start_time
                execution_times.append(time_taken)
                print("Brute force attack successful!")
                print("Private key found: (", i, ",", j, ")")
                print("Time taken:", time_taken, "seconds")
                break
        if found_key:
            break
    else:
        end_time = time.time()
        time_taken = end_time - start_time
        execution_times.append(time_taken)
        print("Brute force attack failed.")
        print("Time taken:", time_taken, "seconds")
import matplotlib.pyplot as plt
bits = [12, 18, 24]
p_times = [0.0009770393371582031, 0.001410508155822754, 0.009863948822021484]
p1_p2_times = [0.0019540786743164062, 0.002821016311645508, 0.019727897644042968]
plt.plot(bits, p_times, marker='o', label='Classical Diffe-Hellman Duration')
plt.plot(bits, p1_p2_times, marker='x', label='Neutrosophic Diffe-Hellman Duration')
plt.xlabel('Bit Length')
plt.ylabel('Duration')
plt.title('Diffe-Hellman Durations for Different Bit Lengths')
plt.legend()
plt.show()
```


DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024

4. Conclusion

In this paper, we have introduced for the first time the neutrosophic and Split-Complex versions of the Diffie-Hellman algorithm. As we demonstrated in both enhanced algorithms, the complexity increased due to the additional computational operations, thereby providing more secure secret keys compared to the traditional Diffie-Hellman algorithm. This was achieved by utilizing extensions of integer numbers in cryptography. We propose utilizing the enhanced algorithm to secure online file transmission by employing the improved secret key as an encryption key for an algorithm such as AES-128.

References

[1] Merkepci, M., and Abobala, M., "Security Model for Encrypting Uncertain Rational Data Units Based on Refined Neutrosophic Integers Fusion and El Gamal Algorithm ", Fusion: Practice and Applications, 2023.

[2] Merkepci, M., and Abobala, M., "On Some Novel Results about Split-Complex Numbers, the Diagonalization Problem and Applications to Public Key Asymmetric Cryptography", Journal of Mathematics, Hindawi, 2023.

[3] S. A. Aparna J R, "Image Watermarking using Diffie Hellman Key Exchange," in International Conference on Information and Communication Technologies, Kochi, India, 2015.

[4] https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und Empfehlungen/ Kryptografie/ KI-in-der-Kryptografie/ki-in-der-kryptografie_node.html. Last visted: 6/17/2024.

[5] Abobala, M., and Allouf, A., " On A Novel Security Scheme for The Encryption and Decryption Of 2×2 Fuzzy Matrices with Rational Entries Based on The Algebra of Neutrosophic Integers and El-Gamal Crypto-System", Neutrosophic Sets and Systems, vol.54, 2023.

[6] Merkepci, M., Abobala, M., and Allouf, A., " The Applications of Fusion Neutrosophic Number Theory in Public Key Cryptography and the Improvement of RSA Algorithm ", Fusion: Practice and Applications, 2023.

[7] Abobala, M., (2021). Partial Foundation of Neutrosophic Number Theory. Neutrosophic Sets and Systems, Vol. 39.

[8] Hasan Sankari, Mohammad Abobala, "On a Generalization of RSA Crypto-system By Using 2-Cyclic Refined Integers", Journal of Cybersecurity and Information Management, Vol 12, 2023.

[9] Mohammad Abobala, Hasan Sankari, and Mohamed Bisher Zeina, " On Novel Security Systems Based on the 2-Cyclic Refined Integers and the Foundations of 2-Cyclic Refined Number Theory", Journal of Fuzzy Extension and Applications, 2024.

[10] Alhasan, Y. A., Alfahal, A. M. A., Abdulfatah, R. A., Ali, R., & Aljibawi, M. (2023). On a novel security algorithm for the encryption of 3x3 fuzzy matrices with rational entries based on the symbolic 2- plithogenic integers and El-Gamal algorithm. International journal of neutrosophic science, 21(1), 88–95. DOI:10.54216/IJNS.210108

[11] Shihadeh, A., Matarneh, K. A. M., Hatamleh, R., Hijazeen, R. B. Y., Al-Qadri, M. O., & Al-Husban, A. (2024). An Example of Two-Fold Fuzzy Algebras Based On Neutrosophic Real Numbers. Neutrosophic Sets and Systems, 67, 169-178.

[12] Abdallah Shihadeh, Khaled Ahmad Mohammad Matarneh, Raed Hatamleh, Mowafaq Omar Al-Qadri, Abdallah Al-Husban. (2024). On The Two-Fold Fuzzy n-Refined Neutrosophic Rings For 2 $\leq$3. Neutrosophic Sets and Systems, 68, 8-25.

[13] Al-Husban, A., Salleh, A. R., & Hassan, N. (2015). Complex fuzzy normal subgroup. In AIP Conference Proceedings (Vol. 1678, No. 1). AIP Publishing.

[14] Abdallah Al-Husban & Abdul Razak Salleh 2015. Complex fuzzy ring. Proceedings of 2nd International Conference on Computing, Mathematics and Statistics. Pages. 241-245. Publisher: IEEE 2015.

[15] Roy, S., Pan, Z., Abu Qarnayn, N., Alajmi, M., Alatawi, A., Alghamdi, A., ... & Rana, J. (2024). A robust optimal control framework for controlling aberrant RTK signaling pathways in esophageal cancer. Journal of Mathematical Biology, 88(2), 14.

[16] Roy, S., Ambartsoumian, G., & Shipman, B. (2023). OPTIMAL CONTROL FRAMEWORKS FOR MODELING DYNAMICS AND ANDROGEN DEPRIVATION THERAPIES IN PROSTATE CANCER (Doctoral dissertation).

DOI: https://doi.org/10.54216/IJNS.250201
Received: January 29, 2024 Revised: April 28, 2024 Accepted: July 20, 2024