Binary Representation Strings for a New Version of Key Exchange Scheme

Ruma Kareem K. Ajeena[1] and Sara Jabbar Yaqoob[2]

[1]Department of Computer Science, College of Education for Pure Science (Ibn Al-Haitham), University of Baghdad and Scientists Foundation for Development, 10053 Baghdad, Iraq

[2]Department of Computer Science, College of Computer Science and Information Technology, University of Kerbala, 56001 Kerbala, Iraq

ruma.k.kh@ihcoedu.uobaghdad.edu.iq, sarah.j@uokerbala.edu.iq

Keywords: Cryptography, Key Exchange Scheme, Binary Representation String, Security.

Abstract: The key exchange scheme (KES) is considered an essential component of many encryption algorithms. Several versions of KES have previously been proposed by numerous researchers in the cryptographic research community. In this work, a new version of KES is proposed over the binary extension field F_{2^m}. The public and private parameters of the proposed KES are represented as binary representation strings (BRSs) over F_{2^m}. The users' private keys are computed as series of BRSs through random selection of binary element strings with different lengths. The public keys are also computed as BRS series over F_{2^m}. In addition, a new algorithm for generating the shared secret key as a BRS series (BRSs-KES) is presented. Some experimental results of the proposed BRSs-KES are implemented using Python programming language. The random generation of BRSs determines the security level of the proposed scheme. By using the proposed BRSs-KES algorithm, more suitable and secure communication is achieved in practice.

1 INTRODUCTION

Mathematics and computer science, with their various specializations [1]–[6], are sciences that have been utilized to model many real problems in our life, especially in cryptographic applications [7]–[10]. One of these applications is the Diffie–Hellman key exchange scheme (KES-DH), which has been presented in various modified versions by many researchers.

In 2005, C. Soojin et al. [11] proposed an alternative KES using the commutative sub-algebras of a full matrix algebra. The KES security here depended on solving difficult matrix equations of the form XRY = T, where R and T are given matrices. In 2016, Hala Abdul Wahab et al. [12] used a proposed homogeneous Diffie–Hellman scheme and Chebyshev polynomials to encrypt a watermarking image. Other related research works can be seen in [13], [14].

In 2019, T. Mefenza and D. Vergnaud [15] proved lower bounds on the polynomial degree and weight that interpolate the Naor–Reingold functions for several keys based on fixed points in a finite field to avoid breaking the security of the generalized KES type DH. See also [16]–[18]. In 2022, A. Ruggeri and M. Villari [19] extended the analysis of the triple Diffie–Hellman (X3DH) protocol to fit generic smart city environments with Edge and IoT nodes.

In 2024, Ruma Ajeena [20] proposed a more secure version of DHKE based on IM2×2 to create the private keys, while the public keys are computed using LPIM2×2 and RPIM2×2. In 2025, P. Kanagala [21] proposed a novel technique to encrypt and decrypt voice signals. The Diffie–Hellman key exchange scheme was used as a basis for the proposed technique. In the same year, D. Alrwashdeh et al. [22] proposed another version of DHKE based on split-complex number theory, integers, and neutrosophic number theory.

This work presents a KES based on BRSs over F_{2^m}. In the proposed BRSs-KES, a shared secret key (SSK) for both users is computed as a BRS. Fast computations are achieved using BRSs, resulting in a higher security level.

This work is organized as follows. Section 2 explains some facts related to F_{2^m}. Section 3 discusses the proposed BRSs-KES. In Section 4, a case study on the proposed BRSs-KES is explained. Additional experimental results are presented in Section 5. Finally, security issues and conclusions are drawn in Section 6.

2 FACTS ON THE BRSS

2.1 Representation of the Elements over F_{2^m}

In the binary field F_2, the size is 2. In other words, it has two elements which are: 0 and 1. The extension fields over F_2, which are named binary extension fields F_{2^m} with $m > 1$. The F_{2^m} field contains 2^m elements. All these elements are represented by binary polynomials (strings) of degree at most $m-1$, namely

$$F_{2^m} = \{a_{m-1}x^{m-1}+\ldots+a_1x + a_0, a_i \in \{0,1\}\}. \quad (1)$$

In binary fields, the arithmetic operations are performed modulo-2, namely the addition and multiplication are done based on the bitwise exclusive or (XOR) and the bitwise and (AND) operations. For example, in the binary extension field F_{2^3}, then, there are $2^3 = 8$ polynomial elements of degree $< m$ which can be represented by string of m bit binary numbers. In more details, one can see Table 1.

Table 1: Elements in F_{2^3}.

F_{2^3}		
Polynomial	Binary	Decimal
0	000	0
1	001	1
x	010	2
$x+1$	011	3
x^2	100	4
x^2+1	101	5
x^2+x	110	6
x^2+x+1	111	7

2.2 Arithmetic Operations on F_{2^m}

The arithmetic operations are explained as follows. Addition. The addition of two polynomials $f(x)$ and $g(x)$ with a maximum degree of k, in ordinary way, is done as component-wise by

$$f(x) + g(x) = \sum_{i=0}^{k} (f_i + g_i)x^i. \quad (2)$$

While, the $f(x) + g(x)$ in F_{2^m} is performed as given in (2) with modulo $p(x)$. In other words, the difference here is the coefficient sum is done modulo p, namely $f_i + g_i \pmod p$. For example, if $f(x) = x^2 + x + 1$ and $g(x) = x+1$ are two polynomials in F_{2^3} with the binary representation of 111 and 011 respectively. Then, to do the addition $f(x) + g(x)$, it can use the XOR operation $\oplus$ by:

$$f(x) + g(x) = (x^2 + x + 1) \oplus x+1 = x^2 + (1\oplus 1)x + (1\oplus 1) = x^2,$$

which is corresponded to:

$$f(x) + g(x) = 111 + 011 = 100.$$

Subtraction. If $f(x)$ and $g(x)$ are two polynomials over F_{2^m}. The subtraction can be done in terms of addition

$$\text{f(x) + g(x) (mod p)= f(x) + (-g(x)) (mod p).}$$

Since, the arithmetic is done over p = 2, so

$$(f+f) \ (mod\ 2) = (f - f) \ (mod\ 2).$$

The multiplication $f(x)\cdot g(x)$ of two polynomials $f(x)$ and $g(x)$ is determined in the convolutional way by

$$f(x)\cdot g(x) = \sum_{j=0}^{i} f_j g_{i-j}, \quad (3)$$

with resulted polynomial has degree $\deg(f) + \deg(g)$. While, the multiplication $f(x)\cdot g(x)$ over F_{2^m} is done modulo $p(x)$. In other words, the difference here is the coefficient multiply in (3) is done modulo p. For example, $f(x) = x^2 + x + 1$ and $g(x) = x+1$ are two polynomials in F_{2^3} with the binary representation of 111 and 011 respectively.

$$f(x)\cdot g(x) = (x^2 + x + 1) \cdot (x+1) \ (\text{mod } 2) = x^3 + (1\oplus 1)\, x^2 + (1\oplus 1)\, x + 1 = x^3+1.$$

which is corresponded to

$$f(x)\cdot g(x) = 111 \cdot 011 = 1001.$$

Division. This operation can be defined in terms of multiplication. If f and g are defined over F_{2^m} then $f(x)/g(x) = f(x)\cdot g(x)^{-1}$, where $g(x)^{-1}$ is the unique element in F_{2^m} such that $g(x) \cdot g(x)^{-1} = 1$, where $g(x)^{-1}$ is called the inverse of $g(x)$.

For more details, it can see [23]-[25].

3 THE KES BASED ON BRSS

A new version of the KES has been proposed in this section. This proposal depends on the binary representation string defined over binary extension field F_{2^m}. The basic idea of proposed scheme which is called BRS-KES is explained as follows.

Suppose g is any BRS given by $g_{BRS} = (g_m g_{m-1} \dots g_2 g_1 g_0)$, with g_i is equal to 0 or 1. which is an element in F_{2^m}. The public domain parameters in BRSS-KES are: g_{BRS} and F_{2^m}. Two users choose respectively two binary representation strings of the binary sequences randomly their elements are in F_2 by

$$a_{BRS} = ((a_n)_2, (a_{n-1}), \dots, (a_1)_2, (a_0)_2),$$

and

$$b_{BRS} = ((b_n)_2, (b_{n-1}), \dots, (b_1)_2, (b_0)_2).$$

with a_i and b_i for i= 1,2,.., n are BRSs.
They compute their private keys respectively by

$$P_{r_a} = \sum_{i=0}^{n} (a_i)_2 = (a_0)_2 + (a_1)_2 + \dots + (a_{n-1})_2 + (a_n)_2$$

and

$$P_{r_b} = \sum_{i=0}^{n} (b_i)_2 = (b_0)_2 + (b_1)_2 + \dots + (b_{n-1})_2 + (b_n)_2.$$

Then, the public keys are computed as the BRSs by

$$P_{k_a} \equiv g_{BRS} \cdot P_{r_a} \pmod 2 \text{ and } P_{k_b} \equiv g_{BRS} \cdot P_{r_b} \pmod 2.$$

These values, namely P_{k_a} and P_{k_b} are exchanged between two users. First user receives P_{k_b} and she/he computes her/his shared secret key $A_{SSK\text{-}BRS}$ as the BRS by

$$A_{SSK-BRS} \equiv P_{ra} \cdot P_{kb} \pmod 2.$$

On the other hand, second user receives P_{k_a} and he/she computes his/ her shared secret key $B_{SSK\text{-}BRS}$ as the BRS by

$$B_{SSK-BRS} \equiv P_{rb} \cdot P_{ka} \pmod 2.$$

The BRSs of $A_{SSK\text{-}BRS}$ and $B_{SSK\text{-}BRS}$ are same, namely

$$A_{SSK-BRS} \equiv B_{SSK-BRS} \pmod 2.$$

The equality relation of SSK of both users can be proved mathematically as follows.

$$\begin{aligned}
A_{SSK-BRS} &\equiv P_{ra} \cdot P_{kb} (mod\ 2) \\
&\equiv P_{ra} \cdot (g_{BRS} \cdot P_{r_b}) (mod\ 2) \\
&\equiv P_{ra} \cdot (P_{r_b} \cdot g_{BRS}) (mod\ 2) \\
&\equiv (P_{ra} \cdot P_{r_b}) \cdot g_{BRS} (mod\ 2) \\
&\equiv (P_{r_b} \cdot P_{ra}) \cdot g_{BRS} (mod\ 2) \\
&\equiv P_{r_b} \cdot (P_{ra} \cdot g_{BRS}) (mod\ 2) \\
&\equiv P_{r_b} \cdot (g_{BRS} \cdot P_{ra}) (mod\ 2) \\
&\equiv P_{r_b} \cdot P_{ka} (mod\ 2) \\
&\equiv B_{SSK-BRS} (mod\ 2).
\end{aligned}$$

For numerical results, it is possible to implement the following algorithm.
Algorithm 3.1. The proposed BRSs-KES algorithm.
Input. $F_2{}^m$, with $m > 1$ and g_{BRS}.
Output. $A_{SSK\text{-}BRS} = B_{SSK\text{-}BRS}$ = SSK.

First user:

1) Generates randomly a sequence a_{BRS} of the BRSs.
2) Computes her/his private key P_{r_a} as a series of the BRSs.
3) Computes her/his public key P_{k_a} as a series of the BRS.
4) Exchanges the value P_{k_a} with second user.
5) Receives the value P_{k_b} from second user.
6) Computes her/his $A_{SSK\text{-}BRS}$ which represents the SSK.

Second user:

1) Generates randomly a sequence b_{BRS} of the BRSs.
2) Computes his/her private key P_{r_b} as a series of the BRSs.
3) Computes his/her public key P_{k_b} as a series of the BRS.
4) Exchanges the value P_{k_b} with first user.
5) Receives the value P_{k_a} from first user.
6) Computes his/her $B_{SSK\text{-}BRS}$ which represents the SSK.

4 STUDY CASE ON BRSS-KES

Two users agree to work over $F_{2^{27}}$. They chooses g is any BRS given by

$$g_{BRS} = 1001.$$

Two users select respectively two binary representation strings a_{BRS} and b_{BRS} randomly as the elements in $F_{2^{27}}$ by

$$a = ((1110001101)_2, (1000010001)_2, (1010110011)_2, (1001000101)_2)$$

and

$$b = ((1111100101)_2, (1000000001)_2, (1111111101)_2, (1010101001)_2).$$

Then, the users compute

$$P_{r_a} = \sum_{i=0}^{n} a_i = (1110001101)_2 + +(1000010001)_2 + (1010110011)_2 +(1001000101)_2 = (0101101010)_2$$

and

$$P_{r_b} = \sum_{i=0}^{n} b_i = (1111100101)_2 + +(1000000001)_2 + (1111111101)_2 + (1010101001)_2 = (0010110000)_2.$$

Then, the public keys are computed as the BRSs by

$$P_{k_a} \equiv g_{BRS} \cdot P_{r_a} (mod\, 2) =(1001)_2 \cdot (0101101010)_2 =(101000111010)_2$$

and

$$P_{k_b} \equiv g_{BRS} \cdot P_{r_b} (mod\, 2). =(1001)_2 \cdot (0010110000)_2 =(10100110000)_2.$$

The values P_{k_a} and P_{k_b} are exchanged between two users. First user receives

$$P_{k_b} = (10100110000)_2.$$

She/He computes her/his shared secret key $A_{SSK\text{-}BRS}$ as the BRS by

$$A_{SSK-BRS} \equiv P_{ra} \cdot P_{kb} (\mathrm{mod}\, 2) = (0101101010)_2 \cdot (10100110000)_2 = (1001111100111100000)_2.$$

On the other hand, second user receives

$$P_{k_a} = (101000111010)_2.$$

He/She computes his/her shared secret key $B_{SSK\text{-}BRS}$ as the BRS by

$$B_{SSK-BRS} \equiv P_{rb} \cdot P_{ka} (\mathrm{mod}\, 2) = (0010110000)_2 \cdot (101000111010)_2 = (1001111100111100000)_2.$$

Thus,

$$A_{SSK-BRS} \equiv B_{SSK-BRS} (\mathrm{mod}\, 2) = (1001111100111100000)_2.$$

5 THE COMPUTATIONAL RESULTS ON BRSS-KES

This section presents new experimental results of the proposed BRSs-KES algorithm with various values of m as shown in Table 2.

The proposed BRSs-KES algorithm is a more efficient, since it is implemented over binary extension fields that gives the fast representations of elements in F_{2^m} in compare to the previous versions on the KES type DH which implemented over prime fields F_p, where p is a prime number.

6 SECURITY

The random selection of binary representation strings (BRSs) forming the sequences a and b, which are used to compute the private keys of two users, is a strong point in the proposed BRSs-KES. The strength arises from the random generation of BRSs over GF(2^m), the variable number of BRSs in sequences aBRS and bBRS, and the choice of different strings with various lengths. This makes it more difficult for attackers to determine the private keys and recover the session key (SSK), since the lengths of the BRSs are unknown and many probabilistic cases need to be tried to identify bit positions in the sequences. The probability of correctly detecting the private BRS of either user is 1/2^m. With a large m, many possibilities for creating long BRSs exist. Therefore, the BRSs-KES resists brute force attacks and other hacking methods relying on trial and error to recover private keys and the SSK. Thus, the proposed BRSs-KES algorithm provides enhanced security for cryptographic applications.

Table 2: Experimental results of the proposed BRSs-KES.

Experimental Result 1		Experimental Result 2	
F_{2^m}	F_{2^8}	F_{2^m}	$F_{2^{17}}$
g_{BRS}	$(1100)_2$	g_{BRS}	$(11100)_2$
a_{BRS}	$((1001)_2,(1101)_2,(0011)_2)$	a_{BRS}	$((1100101)_2,(1000101)_2,$ $(1111111)_2,(1001001)_2)$
b_{BRS}	$((0001)_2,(1000)_2,(1101)_2)$	b_{BRS}	$((1000101)_2,(1101011)_2,(1010111)_2,$ $(1000001)_2)$
P_{r_a}	$(0111)_2$	P_{r_a}	$(0010110)_2$
P_{r_b}	$(0100)_2$	P_{r_b}	$(0111000)_2$
P_{k_a}	$(100100)_2$	P_{k_a}	$(00110001000)_2$
P_{k_b}	$(110000)_2$	P_{k_b}	$(01010100000)_2$
$A_{SSK-BRS}$	$(10010000)_2$	$A_{SSK-BRS}$	$(00010010111000000)_2$
$B_{SSK-BRS}$	$(10010000)_2$	$B_{SSK-BRS}$	$(00010010111000000)_2$
$A_{SSK-BRS} \equiv B_{SSK-BRS}$	$(10010000)_2$	$A_{SSK-BRS} \equiv B_{SSK-BRS}$	$(00010010111000000)_2$
Experimental Result 3		Experimental Result 4	
F_{2^m}	$F_{2^{22}}$	F_{2^m}	$F_{2^{23}}$
g_{BRS}	$(111011)_2$	g_{BRS}	$(10001)_2$
a_{BRS}	$((111111010)_2,(101111011)_2,$ $(110101010)_2)$	a_{BRS}	$((1111111111)_2,(1011110001)_2,$ $(1000010011)_2,(1111111100)_2,$ $(1111001101)_2)$
b_{BRS}	$((111100000)_2,(111110110)_2,$ $(100000111)_2)$	b_{BRS}	$((1000101010)_2,(1111000000)_2,$ $(1000000000)_2,(1111110010)_2)$
P_{r_a}	$(100101011)_2$	P_{r_a}	$(1100101100)_2$
P_{r_b}	$(100010001)_2$	P_{r_b}	$(0000011000)_2$
P_{k_a}	$(11110111110101)_2$	P_{k_a}	$(11000111101100)_2$
P_{k_b}	$(11100010001011)_2$	P_{k_b}	$(00000110011000)_2$
$A_{SSK-BRS}$	$(1111100001011110100101)_2$	$A_{SSK-BRS}$	$(00000101001000110100000)_2$
$B_{SSK-BRS}$	$(1111100001011110100101)_2$	$B_{SSK-BRS}$	$(00000101001000110100000)_2$
$A_{SSK-BRS} \equiv B_{SSK-BRS}$	$(1111100001011110100101)_2$	$A_{SSK-BRS} \equiv B_{SSK-BRS}$	$(00000101001000110100000)_2$

Experimental Result 5	
F_{2^m}	$F_{2^{29}}$
g_{BRS}	$(1011)_2$
a_{BRS}	$((11110000001)_2,(10000100000)_2,(10000000001)_2,$ $(11000010001)_2,(10010010010)_2)$
b_{BRS}	$((11000000011)_2,(10101010101)_2,(10101011110)_2,$ $(10001000101)_2,(10000011001)_2,(11111110110)_2)$
P_{r_a}	$(10100100011)_2$
P_{r_b}	$(00110100010)_2$
P_{k_a}	$(10011001111101)_2$
P_{k_b}	$(00111111110110)_2$
$A_{SSK-BRS}$	$(00110001110000000011011010)_2$
$B_{SSK-BRS}$	$(00110001110000000011011010)_2$
$A_{SSK-BRS} \equiv B_{SSK-BRS}$	$(00110001110000000011011010)_2$

7 CONCLUSIONS

This study presents a BRS-based key exchange scheme (BRSs-KES) that leverages randomness and variability in binary representation strings to strengthen cryptographic security. The algorithm demonstrates robust resistance to brute-force and probabilistic attacks by exponentially increasing the complexity of private key recovery as the field size parameter m grows. The proposed approach offers several advantages: enhanced security through randomized generation of variable-length binary representation strings, which increases the entropy of private keys and improves resilience against cryptanalytic attacks; computational efficiency due to implementation over binary extension fields (F_{2^m}), enabling faster arithmetic operations (bitwise XOR/AND) and reducing computational overhead compared to schemes over large prime fields; scalability and adaptability, as the framework can be easily scaled for larger field sizes or integrated into multi-user cryptographic environments, making it suitable for modern distributed systems and IoT applications; and implementation feasibility, supported by experimental results (see Table 2), which confirm the practical viability of the proposed algorithm, demonstrating stable performance and low computation times even for higher degrees.

REFERENCES

[1] R. K. K. Ajeena and H. Kamarulhaili, “On the distribution of scalar k for elliptic scalar multiplication,” in AIP Conf. Proc., vol. 1682. Melville, NY, USA: AIP Publishing, Oct. 2015.

[2] H. J. Muhasin, A. Y. Gheni, and H. A. Yousif, “Proposed model for data protection in information systems of government institutions,” Bull. Elect. Eng. Inform., vol. 11, pp. 1715–1722, 2022.

[3] W. H. Abdulsalam, Z. H. Ibrahim, B. H. Majeed, and H. T. S. AlRikabi, “Utilizing machine learning techniques to predict university students’ digital competence,” Int. J. Eng. Pedagogy, vol. 15, 2025.

[4] M. H. Abd, O. A. Raheem, J. Kh-Madhloom, and V. Bugrov, “Frequency dispersion of the signal in the recursive digital section of the second order,” J. Phys.: Conf. Ser., vol. 1530, p. 012086, May 2020.

[5] L. A. Tawfeeq, S. S. Hussein, and S. S. Altyar, “Leveraging transfer learning in deep learning models for enhanced early detection of Alzheimer’s disease from MRI scans,” 2025.

[6] B. J. AlKhafaji, M. A. Salih, S. A. Shnain, O. A. Rashid, A. A. Rashid, and M. T. Hussein, “Applying the artificial neural networks with multiwavelet transform on phoneme recognition,” J. Phys.: Conf. Ser., vol. 1804, p. 012040, Feb. 2021.

[7] S. B. Sadkhan and D. M. Reda, “A proposed security evaluator for cryptosystem based on information theory and triangular game,” in Proc. Int. Conf. Advanced Science and Engineering (ICOASE), Oct. 2018, pp. 306–311.

[8] N. K. Abbas and R. K. K. Ajeena, “More secure on the DL-encryption schemes using the TFM function,” in AIP Conf. Proc., vol. 2398. Melville, NY, USA: AIP Publishing, Oct. 2022.

[9] G. E. Arif, F. A. Abdullah, and Y. Al-Douri, “Modeling of structural properties of hexagonal semiconductors,” Procedia Eng., vol. 53, pp. 707–709, 2013.

[10] M. H. Hashem and R. K. K. Ajeena, “The tensor product bipartite graph for symmetric encryption scheme,” in AIP Conf. Proc., vol. 2591. Melville, NY, USA: AIP Publishing, Mar. 2023.

[11] S. Choi, K.-C. Ha, Y.-O. Kim, and D. Moon, “Key exchange protocol using matrix algebras and its analysis,” J. Korean Math. Soc., vol. 42, pp. 1287–1309, 2005.

[12] H. B. A. Wahab, A. J. Abdul-Hossen, and A. S. Kadhom, “Encrypted image watermark in audio files using homogenous Diffie–Hellman with Chebyshev polynomial,” Eng. Tech. J., vol. 34, 2016.

[13] P. Deshpande, S. Santhanalakshmi, P. Lakshmi, and A. Vishwa, “Experimental study of Diffie–Hellman key exchange algorithm on embedded devices,” in Proc. Int. Conf. Energy, Communication, Data Analytics and Soft Computing (ICECDS), Aug. 2017, pp. 2042–2047.

[14] J. Partala, “Algebraic generalization of Diffie–Hellman key exchange,” J. Math. Cryptol., vol. 12, pp. 1–21, 2018.

[15] T. Mefenza and D. Vergnaud, “Polynomial interpolation of the generalized Diffie–Hellman and Naor–Reingold functions,” Des. Codes Cryptogr., vol. 87, pp. 75–85, 2019.

[16] N. Mäurer, T. Gräupl, C. Gentsch, and C. Schmitt, “Comparing different Diffie–Hellman key exchange flavors for LDACS,” in Proc. AIAA/IEEE 39th Digital Avionics Systems Conf. (DASC), Oct. 2020, pp. 1–10.

[17] M. Kara, A. Laouid, M. AlShaikh, A. Bounceur, and M. Hammoudeh, “Secure key exchange against man-in-the-middle attack: Modified Diffie–Hellman protocol,” J. Ilmiah Tek. Elektro Komputer dan Inform., vol. 7, pp. 380–387, 2021.

[18] J. M. Philip, M. J. Thomas, A. J. Sarthik, and R. Aishvarya, “Secure text transfer using Diffie–Hellman key exchange based on cloud,” Int. J. Adv. Eng. Manag., vol. 3, pp. 998–1004, 2021.

[19] A. Ruggeri and M. Villari, “Improving the key exchange process of the extended triple Diffie–Hellman protocol with blockchain,” in Proc. Eur. Conf. Service-Oriented and Cloud Computing. Cham, Switzerland: Springer, Mar. 2022, pp. 49–58.

[20] R. K. K. Ajeena, “A proposed modification of Diffie–Hellman key exchange based on integer matrices,” Int. J. Math. Comput. Sci., vol. 19, pp. 211–218, 2024.

[21] P. Kanagala, “Design and analysis of a Diffie–Hellman-based network security and cryptography approach,” in Research Advances in Network Technologies. Boca Raton, FL, USA: CRC Press, 2025, pp. 206–223.

[22] D. Alrwashdeh, T. Alkhouli, A. S. R. Alhawiti, A. Allouf, H. Edduweh, and A. Al-Husban, “On two novel generalized versions of Diffie–Hellman key exchange algorithm based on neutrosophic and split-complex integers and their complexity analysis,” Int. J. Neutrosophic Sci., vol. 25, pp. 1–10, 2025.

[23] G. W. Reitwiesner, “Binary arithmetic,” in Advances in Computers, vol. 1. New York, NY, USA: Elsevier, 1960, pp. 231–308.

[24] M. D. Fried and M. Jarden, Field Arithmetic, vol. 11. Berlin, Germany: Springer, 2005.

[25] M. I. Saju, “Algebraic extension fields over finite fields and their applications to cryptography,” Ph.D. dissertation, St. Joseph’s College, 2017.