A Privacy Protection Method using Hybrid Cryptography for IoT and MANET

Purwanto Simamora[1]*, Preddy Marpaung[2], Lamganda S Sinaga[3], Dedi Darman Halawa[4]
Universitas Mahkota Tricom Unggul
Corresponding Author: Purwanto Simamora purwantosimamora@gmail.com

A R T I C L E I N F O

Keywords: hybrid cryptography, internet of things, mobile ad hoc network, NTRU encryption, AES algorithm

A B S T R A K

This study proposes the use of hybrid cryptography that combines two cryptographic algorithms, namely AES (Advanced Encryption Standard) and NTRU (N- th degree Truncated Polynomial Ring Unit), to improve the security and efficiency of IoT data encryption in MANET. In the proposed model, AES is used as a symmetric encryption algorithm for data processing, while NTRU is used for key encryption in the key encapsulation mechanism (KEM), which provides secure key exchange in an unsecured channel. The data encryption process uses AES with a key modification technique through the NTRU modular polynomial, which increases the complexity and security of the AES key used, and reduces the risk of modern cryptographic attacks. This study also develops a data encapsulation mechanism (DEM) that ensures data security in transit using AES. The main advantage of this method is the combination of efficient AES speed and NTRU's resistance to quantum attacks and lattice- based attacks, which provides long-term security assurance. The evaluation was conducted to measure the effectiveness of key encapsulation and data encapsulation schemes in improving performance and resilience to attacks, as well as the effectiveness of fast and accurate decryption processes in limited resource environments on IoT and MANET. The results of this study indicate that the hybrid cryptography approach can improve the integrity and confidentiality of communications in IoT networks with optimal performance even in limited resource conditions.

INTRODUCTION

The rapid advancement of Internet of Things (IoT) technology has facilitated the integration of various smart devices capable of communicating autonomously without human intervention. IoT forms a dynamic and complex global network [1], where devices exchange data in real- time. In scenarios involving high mobility and direct node- to-node communication, IoT is often implemented on top of Mobile Ad hoc Network (MANET) architecture, which operates without fixed infrastructure [2][3]. However, the decentralized and dynamic nature of MANET makes it highly vulnerable to various security threats, including eavesdropping, identity spoofing, and routing-based attacks [4].

Numerous cryptographic algorithms have been proposed to secure communication within IoT and MANET systems. However, many of these approaches face limitations in terms of efficiency, power consumption, and resilience against modern and quantum-based attacks [5]. Symmetric cryptographic algorithms such as AES are known for their efficiency, but key distribution remains a major challenge [6]. On the other hand, asymmetric algorithms like NTRU offer strong security and resistance to quantum attacks, although they tend to require intensive computation and complex implementation [7][8][9].

To address these challenges, this research proposes a hybrid cryptographic scheme that combines the strengths of the AES and NTRU algorithms into a unified security framework. The proposed scheme consists of two core mechanisms: a Key Encapsulation Mechanism (KEM) using the NTRU algorithm to secure symmetric key distribution, and a Data Encapsulation Mechanism (DEM) using AES to encrypt the actual data payload. This hybrid approach not only provides strong protection against contemporary cryptographic threats but also ensures efficiency and speed in data communication among resource-constrained IoT devices.

The central problem addressed in this study is how to optimize the security of data communication in IoT within MANET environments, considering constraints in power, memory, and infrastructure. Additionally, this research aims to design a secure and efficient key management scheme that is resilient to modern cryptographic attacks. To that end, structural modifications to the AES algorithm are also proposed, such as replacing the traditional S-Box with NTRU-based polynomials and reconfiguring the key schedule using modular polynomial operations, with the aim of producing more complex and adaptive substitution mechanisms.

1) The objectives of this research include: Developing an efficient and robust hybrid cryptographic security system;
2) Designing a quantum-resistant key generation and distribution scheme;
3) Improving encryption and decryption speeds for resource-constrained IoT devices;

4) Evaluating the resilience and efficiency of the proposed hybrid scheme through simulations of node-to- node communication in MANET environments.

The significance of this research lies in its potential to provide secure IoT communication solutions that are resistant to both modern and quantum attacks, applicable in dynamic network scenarios such as military operations, disaster response, and smart cities, while also contributing to the advancement of scientific knowledge in the field of hybrid cryptographic schemes.

1) The main contributions of this research include: The integration of NTRU modular polynomial operations into the internal structure of AES;
2) The development of an optimized KEM–DEM scheme for secure IoT data transmission in MANETs;
3) The formulation of NTRU key parameters that are more responsive to varying network conditions and threat profiles.

Through a systematic approach focused on efficiency, security, and resistance to attacks, this study aims to lay a foundational framework for the future development of hybrid cryptography-based IoT security systems.

METHODOLOGY

A. Research Approach

This study adopts an experimental approach in engineering a hybrid cryptographic system, aiming to combine the strengths of symmetric (AES) and asymmetric (NTRU) cryptographic algorithms to enhance data communication security in IoT devices operating within MANET environments. The research focuses on efficiency, resilience against modern and quantum attacks, and power consumption optimization for resource-constrained devices.

The proposed security system architecture consists of two core mechanisms:

1. Key Encapsulation Mechanism (KEM): NTRU is employed to encapsulate and securely distribute the symmetric key. NTRU is chosen due to its resilience against quantum-based attacks and its suitability for infrastructure-less network environments such as MANETs;
2. Data Encapsulation Mechanism (DEM): Data encryption and decryption are performed using the AES algorithm after the symmetric key is securely delivered via KEM. AES is selected for its efficiency in data processing, which is crucial for IoT devices with limited computational resources.

B. Structural Modification of the AES Algorithm

To enhance adaptability against modern cryptographic threats, the internal structure of AES is modified as follows:

1. The traditional S-Box is replaced with an NTRU- based polynomial transformation to provide more complex substitutions, thereby improving resistance to conventional cryptanalysis techniques;
2. The AES key schedule is reconfigured using modular polynomial operations to increase session key diversity and randomness.

C. KEM–DEM Scheme Design

The KEM–DEM scheme in this study is designed to be modular and efficient, consisting of the following stages:

1. NTRU Key Initialization: The source node generates an NTRU public–private key pair;
2. AES Key Distribution: The AES key is encrypted using the recipient's NTRU public key;
3. Data Transmission: The data is encrypted using AES and transmitted alongside the NTRU key capsule to the recipient;
4. Data Decryption: The recipient uses their NTRU private key to recover the AES key and decrypt the data.

D. Evaluation and Simulation

The evaluation is conducted through node-to-node communication simulations within a MANET environment using network simulation tools such as NS-3 or OMNeT++. The evaluation criteria include:

1. Encryption and decryption time efficiency;
2. Power consumption of IoT devices;
3. Success rate of key exchange;
4. Resistance to attacks such as eavesdropping, spoofing, and routing attacks

Experiments are performed to assess:

1. The efficiency of the hybrid security system in dynamic network conditions
2. The robustness of the scheme against classical and quantum attacks
3. The encryption performance on memory- and processor-constrained devices

The tools and environment used include Python for cryptographic implementation, the NTRU Encrypt library, and NS-3 for MANET simulation. Security parameters are evaluated based on the NTRU parameters (N, p, q), AES key length (128-bit or 256-bit), data size, and the number of nodes involved.

Hybrid Cryptography Analysis

A. Innovation in Hybrid Cryptography

Hybrid cryptography is a combination of two cryptographic approaches: symmetric key cryptography and asymmetric key cryptography. In the implementation discussed in this study, the NTRU algorithm is used as the public key cryptography, while AES (Advanced Encryption Standard) is employed as the symmetric key cryptography. This combination aims to create a secure and efficient system, particularly in the context of the Internet of Things (IoT), which often operates under power and computational resource constraints.

The main innovation of this hybrid cryptographic scheme lies in the modification of AES's internal processes, especially in the key expansion and Mix Columns transformation, by incorporating the polynomial structure of NTRU. Typically, the Mix Columns step in AES uses a fixed matrix in the finite field GF(28). In this hybrid design, the matrix is replaced with random values derived from NTRU- based polynomials, providing enhanced resistance against various types of attacks, such as differential and linear cryptanalysis.

At the initial stage of communication, NTRU is used to encrypt the AES key, while subsequent encryption processes are carried out using AES. This approach is efficient because NTRU is only used once at the beginning (for keyexchange), while AES is used for the rest of the communication, which is less computationally intensive.

Moreover, the system is equipped with dynamic cryptographic switching, which allows the algorithm to automatically select an encryption variant based on the battery capacity of the IoT device. If the battery is low, the system switches to a lightweight encryption mode, such as AES-CTR, which offers higher speed.

B. AES Key Schedule Modification Using NTRU Algorithm

Modifying the AES key schedule using the NTRU algorithm results in a hybrid encryption scheme that combines the classical security of AES with quantum- resistant cryptography based on NTRU. In this scheme, the sender and the receiver are two parties that use NTRU to securely share the AES key. Once the AES key is securely exchanged, the AES encryption and decryption processes proceed as usual.

In AES, the key is expanded using rotation and substitution operations involving an S-box. In each AES round, the stages SubBytes, ShiftRows, MixColumns, and AddRoundKey are replaced with polynomial-based transformations using NTRU. Each byte in the key exchange process can be encrypted using NTRU encryption. If Sub Bytes is replaced with an NTRU-based operation, Column in the state matrix S into a degree-3 polynomial as follows:

Polynomials reside in the ring Rq = Zq[x]/(xN - 1), i.e., the ring of polynomials with coefficients in Zq. Instead of using a fixed matrix, a random polynomial R(x) in Rq is used,

defined as:

$$R(x) = r0 + r1x + r2x2 + r3x3$$

with coefficients ri randomly selected from Zq. This replaces the Shift Rows function with modular polynomial convolution:

$$S'(x) = S(x) * R(x) \bmod (xN - 1, q)$$

Data encryption for IoT using AES in CBC mode (cipher block chaining) with a random IV (initialization vector) ensures that encryption of the same data produces different ciphertexts. The encryption key is securely generated using the cryptography library and the PBKDF2 (Password-Based Key Derivation Function 2) method with SHA-256 hashing. The key length used is 32 bytes, corresponding to AES-256, which provides strong protection against brute-force attacks by performing 100,000 iterations.

The Mix Columns transformation is modified using NTRU modular polynomials: substitution no longer relies on GF(2^8), but instead utilizes modular polynomial operations. The Sub Bytes Where:

$$C'(x) = C(x) * M(x) \bmod (xN - 1, q)$$

transformation is carried out by multiplying each byte with a specific polynomial to obtain a result modulo q. AES is designed for high efficiency, while NTRU involves more computationally intensive polynomial operations. Therefore, applying it to every byte in Sub Bytes could lead to performance degradation. Assume P(x) is a substitution polynomial in the NTRU domain with modulus q and degree N:

Modified Sub Bytes transformation using NTRU modular polynomial operations is written as: M(x) is the polynomial replacing the Mix Columns matrix C(x) represents the polynomial for a column in the state matrix N is the polynomial degree in the ring Rq Similar to Mix Columns, the Add Round Key operation is modified using NTRU modular polynomials. The secure AES key, generated as described earlier, is used to derive the key matrix.

The transformation is defined as:

$S' = M(x) * P(x) \bmod q$

$R'(x) = S(x) * K(x) \bmod (xN - 1, q)$

Since the function operates in Rq = Zq[x]/(xN-1), calculations are done by reducing the polynomials:

Where:

S(x) and K(x) are polynomials representing rows of the state matrix and key matrix

$S'(x) = (M(x) * P(x)) \bmod (xN - 1, q)$

mod (xN - 1, q) ensures reduction within ring bounds

q is a prime number to maintain bounded polynomial

To preserve AES-level security, the polynomial P(x) must be invertible in Rq and should not be a simple polynomial that's easy to reduce. The S-box is no longer used since it is replaced by the NTRU modular polynomial. To understand this modification, consider an example using a 4x4 matrix block output from the standard AES Sub Bytes transformation:

1. Convert the matrix to decimal form for ease of calculation;
2. Compute S' by multiplying each element M(x) with the substitution polynomial P(x) using modular polynomial operations;
3. Write the resulting matrix from the new Sub Bytes transformation.

The modified Shift Rows approach using NTRU modular polynomials in Rq involves transforming each value.

C. Hybrid Cryptography Method

There are two hybrid cryptography methods used in this research, namely the Key Encapsulation Mechanism (KEM) and the Data Encapsulation Mechanism (DEM). An overview of hybrid cryptography is shown in Figure 1.

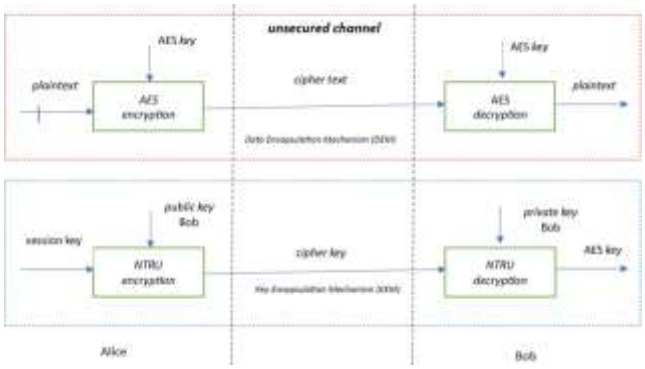


Fig. 1. Overview of Hybrid Cryptography

The Key Encapsulation Mechanism (KEM) is used to encrypt the AES key using NTRU encryption for secure transmission. Alice encrypts the AES key using NTRU so it can be sent securely. The result is a cipher key that is sent to Bob through an unsecured channel. Bob receives the cipher key, then uses his private NTRU key to decrypt it and retrieve the original AES key. With the decrypted AES key, Bob can use it to decrypt the cipher text that was encrypted using the Data Encryption Mechanism (DEM).

To obtain a secure AES session key, Alice generates a random AES key for the communication session. Alice uses a function to generate a random AES key of 16 bytes (128- bit key). This function ensures that the generated key has a sufficient variation of bytes to verify that more than half of the bytes differ.

Session Key (hex): 7afd2e5e13e0ac4a5c6ab436d908ec0aef7a74751dcfdd7082c 20c99e85285b7

Alice uses Bob's NTRU public key with the parameters N=29, p=10, q=11, $f(x)=x+x^2+x^5+x^6+x^9$, and $g(x)=1+x^2+x^4+x^6+x^8$. The result of Python code for this is as follows.

Public key h(x): 2x28 + 5x27 + 2x26 - 2x25 - 2x24 + 4x23 - 2x22
+ 4x21 + 2x20 - 4x19 - 2x18 + 4x17 - x16 - 4x14 - 4x13 + 3x11 + 4x10 - 2x8 + 4x7 - x6 + x5 - 5x4 - 5x3 - 4x2 - 2x - 5

The NTRU encryption process is used to produce a cipher key that is difficult to read. In this case, NTRU encryption is a combination of Alice's session key and Bob's public key. The cipher key is smaller in size than the session key to ensure the transmission process remains fast and secure when sent through an unsecured channel.
Cipher key: eea6b762936142f70689c147f98e8a7d07c1919c2b1666e3e4 1ef93c01c8d59c
To obtain a secure AES key, the transmitted cipher key must first be decrypted using Bob's private key.
Recovered AES Session Key: 7afd2e5e13e0ac4a5c6ab436d908ec0aef7a74751dcfdd7082c 20c99e85285b7

The Data Encapsulation Mechanism (DEM) is used to encrypt messages using AES. Alice has a plaintext message she wants to send to Bob. Alice encrypts the plaintext using the AES algorithm with a session key generated by herself.

After Bob successfully retrieves the original AES key, he uses it to decrypt the message sent by Alice. The data encrypted with AES (cipher text) is transmitted through an unsecured channel. Bob uses the AES key obtained via NTRU to decrypt the cipher text and recover the plaintext.

In the AES algorithm, the plain text undergoes several stages (Sub Bytes, Shift Rows, Mix Columns, and Add Round Key) to produce cipher text. In this case, the plain text contains IoT data which is then converted into cipher text using AES encryption, where the entire AES key scheduling transformation process has been modified using NTRU polynomials to obtain a more secure key.

To generate a secure AES key, the author uses the cryptography library to generate the key randomly. This code produces a highly secure AES key using PBKDF2 with SHA-256, salt, a random IV, CBC mode for data encryption, padding of the plaintext to match the 16-byte block size, and 100,000 iterations to increase resistance to brute-force attacks. The key length can be adjusted (16 bytes for AES-128, 24 bytes for AES-192, and 32 bytes for AES-256). Alice randomly generates the AES key using the following code.
AES Key (hex): 72f2a644066c41270fa0e7a43c6b502b

Alice uses the randomly generated AES key to encrypt the plaintext into cipher text that cannot be directly read without a decryption process. The cipher text is sent through an unsecured channel. When Alice encrypts the plain text using the AES key, Bob must have

the same AES key to decrypt the cipher text and retrieve the original plain text. For example, the data to be encrypted is: 'Living Room Light, Floor 1 - Living Room, ON, Automatic, Light Sensor, Intensity: 80%'. This data is then transformed into cipher text as follows.
Cipher key:
IW/TnwI8W9d72Dy46OGeSkSNrrh7kxHLrf8DxHzlm0M8s
mrzeikAb3C6E15qHP+IjycHAWQ0UsA437rkRR7nk4/w+nx
VtWiqgCU02bNTqcpLa0wjeQqcWOyhoPMbBX3zRcSWcSR Qzs1gI+cEENkYEQ==
Bob will decrypt the message sent by Alice in the form of cipher text using the same AES key that Alice used during the encryption of the plain text.

Recovered AES Key:
1c5a3ae316dbb98288a73079436b9286
Decrypted Plaintexts:
Living Room Light, Floor 1 - Living Room, ON, Automatic, Light Sensor, Intensity: 80%

D. Visualization of IoT Networks in MANET

This program generates a network graph with 50 nodes, connecting them with edges randomly assigned with specific weights. Each node is labeled according to its sequence number (node 1, node 2, and so on). This visualization reflects an IoT network that can illustrate data flow within a Mobile Ad Hoc Network (MANET). The network is represented in the form of a diagram, with node positions arranged using a spring layout algorithm, which. Attempts to minimize the attractive and repulsive forces between nodes to achieve a neater appearance.

Each node transmits data represented in random byte sizes. This data is encrypted using a hybrid encryption technique (AES and NTRU) for each node. At each node, the encryption time is calculated to determine the duration required to encrypt the data using the hybrid method (AES+NTRU). Subsequently, the encrypted data is decrypted to simulate the data flow back. The decryption time is measured to ascertain the duration of the decryption process. The results will indicate how long it takes to encrypt and decrypt data at each node.

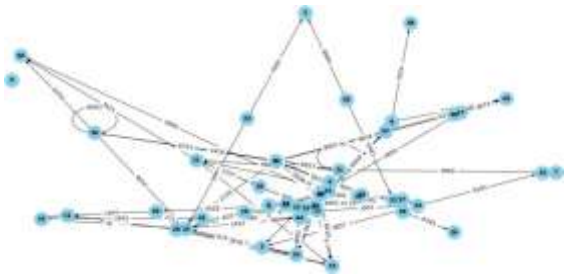


Fig. 2. IoT Node Movement Process

The above figure shows a line graph comparing the encryption times of various encryption methods based on data size (in bytes). The x-axis (horizontal) represents the data size in bytes, with values of 128, 512, 1,024, 5,120, and 10,000 bytes. The y-axis (vertical) indicates the encryption time in seconds, ranging from 0 to 0.15 seconds. The colored lines consist of three segments: the blue line represents the encryption time using the AES method. The encryption time is relatively low and increases slightly as the data size increases. The orange line indicates the encryption time using the NTRU method. The encryption time is higher compared to AES but remains relatively stable with slight increases. The green line shows the encryption time using the combined AES and NTRU method. This method has the highest encryption time among the three, but it decreases slightly with larger data sizes. The red line represents the encryption time using the KEM+DEM method. The encryption time is very small, nearly zero, and does not change with data size.

Overall, this graph demonstrates the performance comparison of encryption times from several different algorithms, where AES is the fastest among the measured methods, while the hybrid method takes the longest. The KEM+DEM method shows an extremely small encryption time that is almost unmeasurable in this graph. The encryption time table for the entire graph is shown in the following table.

Tabel 1. Encryption Time Table

Node AES	Data Size (bytes)	Time (s)	NTRU Time (s)	Hybrid Time (s)
1	52,991	0.033381	0.076454	0.111325
2	98,182	0.029516	0.092160	0.118670
3	99,679	0.038170	0.075463	0.136506
4	85,725	0.039271	0.074254	0.112855
5	81,719	0.044214	0.078803	0.114753
6	80,824	0.022781	0.083241	0.129570
7	21,241	0.045385	0.084632	0.114872
8	77,748	0.020565	0.096724	0.126768
9	43,176	0.034965	0.096756	0.131091
10	21,812	0.033652	0.079565	0.112224
11	91,051	0.040112	0.099654	0.109393
12	71,043	0.037181	0.086779	0.104767
13	35,335	0.043815	0.096861	0.133932
14	54,803	0.029953	0.093368	0.133821
15	11,274	0.038666	0.084492	0.110997
16	25,841	0.043002	0.075263	0.138520
17	52,078	0.027804	0.095803	0.103166
18	32,387	0.026224	0.097610	0.135873
19	87,451	0.021363	0.089164	0.118197
20	58,187	0.037009	0.098224	0.137845
21	69,806	0.033022	0.085155	0.129238

22	56,976	0.043481	0.082675	0.111023
23	22,745	0.039740	0.074468	0.106959
24	28,619	0.045694	0.073520	0.126127
25	24,678	0.042954	0.100075	0.127210
26	36,933	0.047837	0.071965	0.103845
27	70,794	0.041021	0.074551	0.125823
28	54,267	0.039333	0.092513	0.119587
29	29,006	0.044847	0.073302	0.133657
30	53,724	0.024302	0.070641	0.119735
31	27,526	0.044664	0.089544	0.108599
32	94,100	0.038496	0.096275	0.123418
33	57,710	0.030844	0.098069	0.131697
34	5,037	0.036780	0.084235	0.132036
35	84,979	0.041018	0.082782	0.141974
36	93,090	0.023023	0.096381	0.128958
37	79,614	0.026378	0.093451	0.105662
38	58,546	0.049086	0.099763	0.111375
39	65,664	0.026704	0.073030	0.120952
40	32,437	0.032128	0.083035	0.141541
41	42,167	0.024525	0.092056	0.120940
42	25,768	0.049373	0.098901	0.130281
43	66,506	0.040612	0.090507	0.130361
44	4,165	0.031057	0.094758	0.108088
45	28,539	0.049452	0.094679	0.143332
46	21,619	0.039502	0.074932	0.122801
47	30,077	0.045244	0.072486	0.098167
48	54,166	0.038338	0.072498	0.112160
49	17,172	0.025366	0.083506	0.106704
50	29,117	0.031959	0.097150	0.129407

The table illustrates the data transmission, encryption, and decryption processes performed by 50 nodes using a hybrid encryption method. Each node transmits data to the next node after performing encryption. The transmitted data varies in size, ranging from 5,037 bytes (node 34) to 99,679 bytes (node 3).

The encryption time for each node varies, with the longest time recorded at node 5 (0.142610 seconds) and the shortest at node 9 (0.101243 seconds). The average encryption time for all nodes is approximately 0.118 seconds.

The decryption time is very fast, ranging from 0.000002 to 0.000005 seconds for all nodes. This indicates that the decryption process is highly efficient and does not significantly impact performance. Each node successfully decrypts the encrypted data, resulting in a unique hexadecimal string for each node. The decrypted data confirms that the encryption and decryption processes were successful, with no errors reported.

The previously provided encryption time table displays the encryption times for the AES, NTRU, and Hybrid algorithms. The hybrid encryption time for each node is generally longer compared to the encryption times of AES and NTRU, as shown in the table. This indicates that while hybrid encryption offers enhanced security, the time required for encryption is higher.

RESULTS AND DISCUSSION

The encryption and decryption processes conducted on 50 IoT nodes demonstrate that hybrid encryption requires a longer encryption time compared to other algorithms such as AES and NTRU, despite providing higher security. The highly efficient decryption time indicates that once the data is encrypted, the decryption process can be performed almost instantaneously, which is particularly beneficial for real-time applications.
All nodes successfully decrypted the data, confirming the integrity and reliability of the encryption process. The unique hexadecimal strings generated during decryption verify that the data remains intact throughout the transmission process.

Future research may focus on optimizing the hybrid encryption algorithm to reduce encryption time without compromising security. This could involve exploring more efficient cryptographic techniques or hardware acceleration. Further testing should be conducted to evaluate the scalability of the encryption process as data sizes increase significantly. Understanding how encryption time evolves with larger datasets is crucial for applications related to big data. Implementing hybrid encryption methods in real-world scenarios, such as cloud storage and secure communications, will provide practical insights into performance and security in operational environments.

ADVANCED RESEARCH

The authors would like to express their gratitude to Mr. Opim Salim Sitompul, Dr. Mohammad Andri Budiman, and Mr. Jos Timanta Tarigan from the Faculty of Computer Science and Information Technology, University of North Sumatra, for their guidance and advice in this paper

REFERENCES

Mansour et al., “Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions,” Open Access, 2023, 16(8), 3465.

Alenezi, M.N., Alabdulrazzaq, H., Mohammad, N.Q., "Symmetric Encryption Algorithms: Review and Evaluation Study," IJCNIS, 2020, vol.12, No.2.

Huang, D., Zhou, J., Mi, B., Kuang, F., & Liu, Y., "Key-Based Data deduplication via Homomorphic NTRU for Internet of Vehicles," IEEE, 2022, vol 72 Issue: 1.

Alshurideh, M., & Kurdi, B., "Factors Affecting Social Networks Acceptance: An Extension to the Technology Acceptance Model using PLS-SEM and Machine Learning Approach," IJDNS, 2023, 7(1), 489-494.

Vishwanathrao, B.A., Vikhar, P.A., "Reinforcement Machine Learning-based Improved Protocol for Energy Efficiency on Mobile Ad-Hoc Networks," IJISAE, 2024, 12(8s), 654-670.

Kumar et al., "A Low Area High Speed FPGA Implementation of AES Architecture for Cryptography Application," Electronics, 2021,10, 2023.

Yassein, H. R., Al-Saidi, N. M., Farhan, A. K., "A new NTRU cryptosystem outperforms three highly secured NTRU-analog systems through an innovational algebraic structure," Journal of Discrete Mathematical Sciences and Cryptography, 2022, 25(2), 523-542.

Ahmad, K., Kamal, A., Ahmad, K. A. B., Khari, M., & Crespo, R. G., "Fast hybrid-Mix Net for Security and Privacy using NTRU Algorithm," JISA, 2021, 60, 102872.

Salman, H. S., & Yassein, H. R., "An Innovative HSS Algebra for Designing a Secure Like-NTRU Encryption, " Mathematical Statistician and Engineering Applications, 2022, 71(4), 6098-6113.Pelanggan PT. PLN (Persero) Unit Layanan Pelanggan Tomohon. *Jurnal EMBA: Jurnal Riset Ekonomi, Manajemen, Bisnis Dan Akuntansi*, 10(4), 62–71. https://ejournal.unsrat.ac.id/v3/index.php/emba/article/view/43763.

Leonardo, B. R., Wenas, R. S., & Poluan, J. G. (2021). Pengaruh Kualitas Produk, Promosi Dan Harga Terhadap Kepuasan Pelanggan Di Marketplace Shopee Pada Saat Pandemi Covid-19 (Studi Pada Mahasiswa Fakultas Ekonomi Dan Bisnis Unsrat. *Jurnal EMBA: Jurnal Riset Ekonomi, Manajemen, Bisnis Dan Akuntansi*, 9(3), 1721–1730. https://ejournal.unsrat.ac.id/index.php/emba/article/view/35960.

Made, N., Febriani, A., Ketut, I. G. A., & Ardani, S. (2021). The Influence of Customer Experience, Ease of Use, And Trust on Repurchase Intention (Case Study of Tokopedia Consumers in Denpasar). American Journal of Humanities and Social Sciences Research, 5, 378–383. https://www.ajhssr.com/wp-content/uploads/2021/02/ZR2152378383.pdf

Pratama Yudha, S. A., Arifin, R., & Hufron, M. (2019). Pengaruh Kemudahan Penggunaan, Kualitas Pelayanan, Harga dan Promosi Terhadap Kepuasan Pelanggan Grabfood. *Riset Manajemen*, 1(2) 101–113. http://www.academia.edu/download/38127186/Review_Jurnal_Impact_of_CRM_Factors_on_Customer_Satisfaction_and_Loyalty.docx.

Qibtiyah, D., Hurruyati, R., & Hendrayati, H. (2021). The Influence of Discount on Repurchase Intention. Proceedings Of The 5th Global Conference on Business, Management and Entrepreneurship (Gcbme 2020), 187(2020), 385–389. https://www.atlantis-press.com/proceedings/gcbme-20/125960467

Sasongko, A. D., & Wuryaningsih, D. L. (2020). Pengaruh Harga, Kualitas Layanan, dan Promosi Terhadap Preferensi Pilihan Konsumen pada Grab-Food di Kabupaten Sukoharjo (Doctoral dissertation, Universitas Muhammadiyah Surakarta). https://eprints.ums.ac.id/83993/1/NASKAH%20PUBLIKASI.pdf

Haerani, S., Parmitasari, R. D. A., Aponno, E. H., & Aunalal, Z. I. (2019). Moderating effects of age on personality, driving behavior towards driving outcomes. *International Journal of Human Rights in Healthcare*. https://doi.org/10.1108/IJHRH-08-2017-0040