Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 4, August 2023, pp. 849-855

DOI: https://doi.org/10.52436/1.jutif.2023.4.4.1129
p-ISSN: 2723-3863
e-ISSN: 2723-3871

SECURE TEXT ENCRYPTION FOR IOT COMMUNICATION USING AFFINE CIPHER AND DIFFIE-HELLMAN KEY DISTRIBUTION ON ARDUINO ATMEGA2560 IOT DEVICES

Permana Langgeng Wicaksono Ellwid Putra*[1], Christy Atika Sari[2], Folasade Olubusola Isinkaye[3]

[1,2]Study Program in informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang, Indonesia
[3]Department of Computer Science, Ekiti State University, Ekiti, Nigeria
Email: [1]langgeng86@gmail.com , [2]christy.atika.sari@dsn.dinus.ac.id , [3]folasade.isinkaye@eksu.edu.ng

(Article received: June 14, 2023; Revision: July 10, 2023; published: August 21, 2023)

Abstract

In an Internet of Things (IoT) system, devices connected to the system exchange data. The data contains sensitive information about the connected devices in the system so it needs to be protected. Without security, the data in the IoT system can be easily retrieved. One way to prevent this is by implementing cryptography. Cryptography is a technique for protecting information by using encryption so that only the sender and receiver can see the contents of the information contained therein. The implementation of cryptography on IoT devices must consider the capabilities of IoT devices because in general IoT devices have limited processing capabilities compared to computer devices. Therefore, the selection of encryption algorithms needs to be adjusted to the computational capabilities of IoT devices. In this research, the affine cipher cryptography algorithm and Diffie-hellman key distribution algorithm are applied to the arduino atmega2560 IoT device. The purpose of this research is to increase the security of the IoT system by implementing cryptography. The method used in this research involves setting up a sequence of encryption and decryption steps using an affine cipher and diffie-hellman algorithms. Furthermore, these algorithms were implemented on an Arduino IoT device. Finally, the decryption time based on the number of characters and the avalanche test were tested. The results showed that on average, Arduino can perform decryption using affine cipher and diffie-hellman algorithms in 0.07 milliseconds per character. The avalanche test produced an average percentage of 45.51% from five trials.

Keywords: *arduino, affine cipher, diffie-hellman, IoT, cryptography.*

PENGENKRIPSIAN TEKS UNTUK KOMUNIKASI IOT MENGGUNAKAN AFFINE CIPHER DAN DISTRIBUSI KUNCI DIFFIE-HELLMAN PADA PERANGKAT IOT ARDUINO ATMEGA2560

Abstrak

Dalam sistem *Internet of Things* (IoT), perangkat yang terhubung dalam sistem saling bertukar data. Data tersebut mengandung informasi sensitif tentang perangkat yang terhubung didalam sistem sehingga perlu dilindungi. Tanpa keamanan, data didalam IoT dapat dengan mudah diambil. Salah satu cara untuk mencegahnya adalah dengan menimplementasikan kriptografi. Kriptografi adalah suatu teknik untuk melindungi informasi dengan menggunakan enkripsi sehingga hanya pengirim dan penerima saja yang dapat melihat isi informasi yang terkandung didalamnya. Penerapan kriptografi pada perangkat IoT harus mempertimbangkan kemampuan dari perangkat IoT karena pada umumnya perangkat IoT memiliki kemampuan pemrosesan yang terbatas dibandingkan dengan perangkat computer. Oleh karena itu, pemilihan algoritma enkripsi perlu disesuaikan dengan kemampuan komputasi perangkat IoT. Pada penelitian ini diterapkan algoritma kriptografi affine cipher dan algoritma distribusi kunci diffie-hellman pada perangkat IoT arduino atmega2560. Tujuan dari penelitian ini adalah untuk meningkatkan keamanan sistem IoT dengan mengimplemntasikan kriptografi. Metode yang digunakan dalam penelitian ini melibatkan pengaturan urutan langkah-langkah enkripsi dan dekripsi dengan menggunakan algoritma affine cipher dan diffie-hellman. Selanjutnya, algoritma ini diimplementasikan pada perangkat IoT Arduino. Terakhir, dilakukan pengujian terhadap waktu dekripsi berdasarkan jumlah karakter serta pengujian avalanche test. Hasil penelitian menunjukkan bahwa Arduino rata-rata dapat melakukan dekripsi menggunakan algoritma affine cipher dan diffie-hellman dalam waktu 0,07 milisekon per karakter. Pengujian avalanche test menghasilkan presentase rata-rata sebesar 45,51% dari lima kali percobaan.

Kata kunci: *arduino, affine cipher, diffie-hellman, IoT, kriptografi.*

1. INTRODUCTION

Along with the advancement of the digital age, technology has also spread to every field of human life. An example of this technological development is the Internet of Things or IoT. IoT is an idea where electronic devices can communicate with each other because they are connected to the internet network[1]. The connection of these devices to the internet allows users to monitor and control devices remotely[2]. In an IoT system, devices connected to the system exchange data[3]. The data contains sensitive information about the connected devices in the system so it needs to be protected [4]. Without security, data in the IoT system can be easily retrieved[5]. An example of hacking is data retrieval by infiltrating the network or sniffing using the Wireshark application [6].

One way to prevent this is by implementing cryptography. Cryptography is a technique used to protect information with encryption so that only the sender and receiver can see the contents of the information contained therein[7]According to document number 62591 of the International Electrotechnical Commission (IEC), data encryption is an important part of IoT [8]. This is because the encryption process helps maintain the privacy and security of data from IoT devices that send data over the network [9]. The application of cryptography on IoT devices must consider the capabilities of IoT devices because in general IoT devices have limited processing capabilities compared to computer devices[10]. Therefore, the choice of data encryption algorithm needs to be adjusted to the limitations of the computing capabilities of IoT devices [11]. In this research, the algorithm used is the affine cipher algorithm for the encryption process and the Diffie-Hellman algorithm for key distribution. The selection of the affine cipher algorithm is based on the fact that this algorithm is a development of the simple cryptography Caesar cipher where the encryption or decryption step involves multiplying the value and shifting the plaintext [12]. While the selection of the Diffie-Hellman algorithm is based on the fact that this algorithm can generate a sliding key with two private keys to provide additional security.

The use of affine cipher algorithm has been used in several studies such as Nasution's research which modifies the affine cipher encryption algorithm to better maintain its security [13], Nurjamiyah's research which uses the affine cipher encryption algorithm to encrypt email transmission data [12], and Dwi et al.'s research which uses the affine cipher encryption algorithm in sending One Time Pad (OTP) messages [14]. In addition, the Diffie-Hellman algorithm has also been used in several previous studies such as in the research of Khairani et al. together with the affine cipher encryption algorithm to produce a random and difficult-to-decipher written message[15] and in the research of Wang & Mogos who used this algorithm to produce a raspberry pie system that was not easily hacked[16].

The IoT device used is the arduino atmega2560. Arduino atmega2560 is an open-source microcontroller that can be developed freely[17]. For specifications, Arduino atmega2560 has a 16Mhz clock, 256kb memory, 54 digital pins, and 16 analog pins. Developing with this device can be done using the c programming language and the arduino IDE [18].

The implementation of cryptography on Arduino has been done in several previous studies. In the research of Budiyanto et al., the implementation of the Vernam cipher and Three Pass Protocol was carried out on the Arduino ATmega328 IoT device. This algorithm can be implemented by producing an average encryption processing time of 1.9 milliseconds for 16 characters[19]. In the research conducted by Nurrohmah et al., the implementation of Grain V1 and 128-bit was carried out on the Arduino Mega2560 IoT device. These two algorithms can be implemented without sacrificing performance and get the results by analyzing the probability values of 0.193, 0.835, and 0.036 at 8-bit, 12-bit, and 16-bit[20].

The purpose of this research is to increase the security of the Internet of Things (IoT) system by implementing cryptography. Specifically by using the affine cipher algorithm for the encryption process and the diffie-hellman algorithm for key distribution.

2. METHOD

2.1 Encryption and Decryption Process Workflow

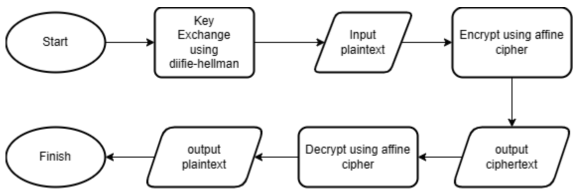

Start
Key Exchange using diifie-hellman
Input plaintext
Encrypt using affine cipher
output ciphertext
Decrypt using affine cipher
output plaintext
Finish

Figure 1. Encryption and Decryption Process Workflow

Based on Figure 1 the encryption and decryption workflow in this research consists of several steps. First, there is a key using the Diffie-Hellman algorithm between the system and the Arduino. Then the system enters the plain text that will be used as the encrypted message. The plaintext will be encrypted by the system using the affine cipher algorithm to produce cipher text. After that, the encrypted text will be sent to the Arduino and decrypted using the affine cipher method to produce plain text again.

1. Key exchange

Key exchange using the Diffie-Hellman algorithm involves several steps that must be followed as follows. First, The communicating parties are the system and Arduino. Initially, the parties agree to use the same parameters R and G, provided that R is a random prime number between 1 and 100 and G is a random number whose value is greater than 1 and smaller than R (1<G<R). Then the system and Arduino randomly choose a secret number between 1 and 32. This number will be the private key for each party. After setting the parameters and private keys, both parties will calculate the public key with each party's private key. The system uses equation 1 to generate public key A, while Arduino uses equation 2 to generate public key B. The public keys are then exchanged.

$$A = G^a \ mod \ R \tag{1}$$

$$B = G^b \ mod \ R \tag{2}$$

Based on equations (1) and (2), the system gets public key B from Arduino and Arduino gets public key A from the system. Next, each party calculates the shared key X using the received public key and its private key. The system calculates with equation (3) and Arduino calculates with equation (4). The result X which is a shared key will be used as a shifting key. The slider key will be used in the encryption and decryption process using the affine cipher algorithm.

$$X = B^a \ mod \ R \tag{3}$$

$$B = G^b \ mod \ R \tag{4}$$

2. Encryption

The following are the steps in the encryption process using the affine cipher algorithm. First of all, parameter a is determined as a random number between 1 and 26. This parameter must fulfill two conditions, namely a must be relatively prime to 26 and must have an inverse modulus. Next, plaintext is determined to be encrypted. Then, each letter in the plaintext is converted into a decimal number using ASCII encoding. Each decimal number in the plaintext will be encrypted using equation (5). In the last step, the resulting numbers will be converted back into ASCII characters. Thus, the ciphertext is formed, which is the encrypted sentence of the plaintext.

$$C = (a \ x \ P + X) \ mod \ 26 \tag{5}$$

3. Decryption

The following are the steps in the decryption process using the affine cipher algorithm. First, a ciphertext is prepared for decryption. Then, each letter in the ciphertext is converted into a decimal number using ASCII encoding. Each decimal number in the ciphertext is decrypted using equation (6).

$$P = \left(a^{-1} \ x \ (C - X)\right) mod \ 26 \tag{6}$$

In the last step, the resulting numbers will be converted back into ASCII characters. Thus, the ciphertext is formed, which is the encrypted sentence of the plaintext.

2.2 Implementation

In the implementation stage, the first step is to create a code that will run the affine cipher and diffie-hellman algorithms. This code will be implemented in the Arduino C and Python as the system. After the code has been created, then the code will be implemented in Arduino by connecting it to a laptop and then uploading the code through the Arduino IDE.

2.3 Testing

In this study, two tests will be carried out, namely time testing and avalanche effect testing. The avalanche effect is a technique commonly used to describe the security level of a symmetric key encryption process and hash function [21]. In time testing, ciphertext data with a varying character count will be used, and then the average time required by the Arduino to decrypt for each variation in the character count will be calculated.

While testing the avalanche effect, the private key used is in the form of a sentence. Later the private key will be converted into decimal numbers using ASCI coding and then added to each decimal number to become a number-shaped private key. The method used in this test is to encrypt a plaintext until the ciphertext result is obtained. Then change 1 sentence in the private key used in the previous encryption. After that, encrypt the same plaintext with the changed private key. The last step is to compare the results of the two ciphertexts and calculate how many bits have changed then calculate with equation (7) to get the avalanche test percentage. If the percentage result can reach 40% to 60%, it can be said that the algorithm used has a good level of security

$$AE = \left(\frac{total \ bits \ flip}{total \ bits}\right) x \ 100\% \tag{7}$$

3. RESULT

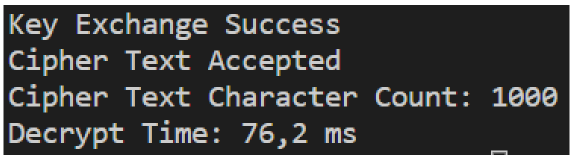
3.1 Time Testing

Table 1. Time Trial Results Against the Character Count

Character Count	Total Experiment	Average Time
100	10	8,6 ms
200	10	15,5 ms
300	10	23,7 ms
400	10	32,6 ms
500	10	38,9 ms

600	10	47,5 ms
700	10	54,3 ms
800	10	61,8 ms
900	10	68,2 ms
1000	10	76,6 ms

Based on Table 1, the results of experiments conducted to examine the relationship between character count and the time taken for decryption. The experiments consisted of a total of 100 trials, with 10 trials conducted for each variation in the character count. The table demonstrates that the average time required by the Arduino for decryption computation increases significantly as the character count increases. For instance, in the experiment with 100 characters, the average time was 8.6 milliseconds (ms), while with 1000 characters, the average time was 76.6 milliseconds (ms). On average, each increase in character count resulted in a difference of 7.5 milliseconds (ms) in the average decryption time. A more detailed calculation reveals that the Arduino takes approximately 0.07 milliseconds (ms) to decrypt each character.



```
Key Exchange Success
Cipher Text Accepted
Cipher Text Character Count: 1000
Decrypt Time: 76,2 ms
```

Figure 2. Time Testing Command Line Result

Based on Figure 2, approximately 76.2 milliseconds are required for Arduino to decrypt the cipher text 1000 character count. This data is taken from a particular experiment involving 1000 character count as cipher text input. The decryption time data is sent from the Arduino to the system (in this research replaced with Python) and displayed on the command line.

3.2 Avalanche Effect Testing

Avalanche tests have been conducted on the affine cipher and diffie-hellman algorithms on an Arduino device as shown Table 2. First experiment, there were 39 bit flips out of a total of 80 bits, resulting in an avalanche effect presentation of 48.7%. In this experiment, there was a small modification of the private key from "python" to "dython". In the second experiment, there were 74-bit flips or bit changes out of a total of 168 bits, which resulted in an avalanche effect presentation of 44.04%. In this experiment, there was a small modification to the private key from "arduino" to "Nduino". In the third experiment, there were 227-bit flips or bit changes out of a total of 520 bits, which resulted in a presentation of the avalanche effect of 43.65%. In this experiment, there was a small modification to the private key from "udinus" to "Idinus". In the fourth experiment, there were 230-bit flips or bit changes out of a total of 536 bits, which resulted in an avalanche effect presentation of 42.91%.

Table 2. Avalanche Effect Testing Table

Description	**Experiment**				
	1	2	3	4	5
Plaintext	Hello World	Selamat datang di udinus	Mempelajari kriptografi dengan algoritma affine cipher dan Diffie hellman	Jalan jalan ke kota semarang jangan lupa mampir ke lawang sewu di deket tugu muda	Universitas dian nuswantoro Jalan Imam Bonjol Pendrikan Kidul Kecamatan Semarang Tengah Kota Semarang Jawa Tengah
Private key	python	arduino	udinus	semarang	informatika
Ciphertext	FQZZO CODZL	XFOLTLC ALCLYP AZ HAZYHX	BNBQNWTMTAH RAHQKLXATSH INGXTG TWXLAHKBT TSSHGN DHQCNA ITG IHSSHN CNWWBTG	MTWTG MTWTG RN RLKT FNBTATGX MTGXTG WPQT BTBQHA RN WTZTGX FNZP IH INRNK KPXP BPIT	JABOHUZBENZ CBNA AJZTNAEFUF GNQNA BVNV SFAGFQ AFVFU KHACUBLNA LBCJQ LHXNVNENA ZHVNUNAR EHARNW LFEN ZHVNUNAR GNTN EHARNW
Private key modification	dython	Nrduino	Idinus	Gemarang	Anformatika
Ciphertext modification	KVEET HTIEQ	EMVSASJ HSJSFW HG OHGFOE	OAODAJGZGNU ENUDXYKNGFU VATKGT GJKYNUXOG GFFUTA QUDPAN VGT VUFFUA PAJJOGT	ZGJGT ZGJGT EA EYXG SAOGNGTK ZGTKGT JCDG OGODUN EA JGMGTK SAMC VU VAEAX XCKC OCVG	WNOBUHMORAM POAN NWMGANRSHS TADAN OIAI FSNTSD NSISH XUNPHOYAN YOPWD YUKAIARAN MUIAHANE RUNEAJ YSRA MUIAHANE TAGA RUNEAJ
Total bit	80	168	520	536	808
Bit flip	39	74	227	230	390
Avalanche Effect precentage	48,7%	44,04%	43,65%	42,91%	48,26%

In this experiment, there was a small modification to the private key from "Semarang" to "Gemarang". The next experiment, there were 390-bit flips or bit changes out of a total of 808 bits, which resulted in an avalanche effect presentation of 48.26%. In this experiment, there was a small modification to the private key from "informatika" to "Anformatika".

Table 3. Experiment 1 Ciphertext In Biner

Ciphertext in biner	**Ciphertext modification in biner**
0100011001010001010	0100101101010110010
1101001011010010011	0010101000101010101
1100100000010000110	0000100000010010000
1001111010001000101	1010100010010010100
101001001100	010101010001

The results of converting ciphertext into binary numbers in experiment 1 shown in Table 3. On the left side of the column, there are binary numbers before the modification of the private key, while on the right side, there are binary numbers after the modification. In the 4, 7, 13, 14, 21, 23, 29, 31, 35, 52, 59, 68, 71, 77, 79, 83, and 87 binary number sequences, there is a change from 0 to 1, while in the 5, 15, 19, 20, 22, 27, 28, 30, 36, 38, 39, 54, 55, 60, 62, 63, 69, 75, 76, 78, 84, and 85 binary number sequences, there is a change from 1 to 0.

Based on Table 3, 39 bits flip in the first experiment of a total of 80 bits when the private key was modified. If we apply the avalanche test formula that calculates the percentage of bit changes, the steps are to divide the number of bit flip, which is 39, by the total number of bits, which is 80, and then multiply the result by 100. Thus, the percentage of bit changes is 48.7%. The equation that describes the above case more clearly is shown in equation (8).

$$AE = \left(\frac{39}{80}\right) x\ 100\% = 48.7\% \qquad (8)$$

4. DISCUSSION

In time testing, it was found that the longer the ciphertext, Arduino requires more time to perform the decryption process. This is because as the character count or bits in the data increases, the mathematical operations required for encryption and decryption become longer and more complex. Due to the computational limitations of the Arduino device, these longer and more complex mathematical operations take a longer time to process. Some related studies also prove this statement, such as in the research of Joyoputro et al. where the time required to encrypt 216-bit plaintext is 28225 milliseconds while the time required for 352-bit plaintext is 49340[6]. In addition, in the research of Mufarokah et al. [22], the amount of encryption time also increases proportional to the increasing number of bits in the key, at a key of 128 bits it takes a total time of 1047.6 milliseconds (ms), at a key of 192 bits it takes a total time of 1080 milliseconds (ms) and at a key of 256 bits it takes a total time of 1114.6 milliseconds (ms).

In this research, an avalanche test was conducted by changing one letter in the private key and comparing the resulting cipher text. In the research of Nuraeni et al., the avalanche test was conducted by changing one bit of the letter in the plaintext. The avalanche test was used in the study to measure the security level of the AES and super encryption algorithms in the context of securing land tax data. The results show that the avalanche test on the AES algorithm is 6.90%, while the super encryption algorithm is 11.30%[23]. Furthermore, in Rizky's and Anwar's research, the avalanche test is also used to measure the security of the AES algorithm in real-time chat applications. The results show that the average avalanche test reaches 50% when one letter in the plaintext is changed, and 49% when one letter in the key is changed[24].

5. CONCLUSION

Based on the results of the research and testing that has been done, it is found that the affine cipher cryptographic algorithm and the diffie-hellman key distribution algorithm can be successfully applied effectively on the Arduino ATmega2560 device. The combination of these two algorithms results in a decryption time of 0.07 milliseconds per character. In addition, in the avalanche test, both algorithms showed a high level of security by producing an average avalanche test percentage of 45.51% from five trials. This figure falls into the category of avalanche effect percentage that is considered good, which is between 40% and 60%.

For future research to get higher value, experiment might be conducted using modern cipher based on key permutation or randomized in pseudorandom generator such as Linear Congruential and Blum Blum Sub Generator. It experiment might be set in another cryptography media such as image or audio file.

BIBLIOGRAPHY

[1] I. Ardiansah, N. Bafdal, E. Suryadi, and A. Bono, “Greenhouse monitoring and automation using arduino: A review on precision farming and Internet of Things (IoT),” *Int J Adv Sci Eng Inf Technol*, vol. 10, no. 2, pp. 703–709, 2020, doi: https://doi.org/10.18517/ijaseit.10.2.10249.

[2] A. R. Kedoh, N. Nursalim, H. J. Djahi, and D. E. D. G. Pollo, “Sistem Kontrol Rumah Berbasis Internet of Things (Iot) Menggunakan Arduino Uno,” *Jurnal Media Elektro*, pp. 1–6, 2019, doi: 10.35508/jme.v8i1.1403.

[3] Wilianto and A. Kurniawan, “Sejarah , Cara Kerja Dan Manfaat Internet of Things,”

Matrix, vol. 8, no. 2, pp. 36–41, 2018, doi: https://doi.org/10.31940/matrix.v8i2.

[4] Z. B. Celik *et al.*, "Sensitive information tracking in commodity IoT," in *Proceedings of the 27th USENIX Security Symposium*, 2018, pp. 1687–1704.

[5] P. Arpaia, F. Bonavolontá, and A. Cioffi, "Problems of the advanced encryption standard in protecting Internet of Things sensor networks," *Measurement (Lond)*, vol. 161, Sep. 2020, doi: http://doi.org/10.1016/j.measurement.2020.107853.

[6] K. Joyoputro, A. Kusyanti, and F. A. Bakhtiar, "Implementasi Algoritme Kriptografi Lizard untuk Mengamankan Pengiriman Data Menggunakan Arsitektur Web Service REST pada Mikrokontroler NodeMCU," vol. 2, no. 12, pp. 6292–6299, 2018, [Online]. Available: http://j-ptiik.ub.ac.id

[7] R. Bhandari and V. B. Kirubanand, "Enhanced encryption technique for secure iot data transmission," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, pp. 3732–3738, 2019, doi: https://doi.org/10.11591/ijece.v9i5.pp3732-3738.

[8] A. Zubaidi, R. I. Sardi, and A. H. Jatmika, "Pengamanan Internet of Things Berbasis NodeMCU Menggunakan Algoritma AES Pada Arsitektur Web Service REST," *Edumatic: Jurnal Pendidikan Informatika*, vol. 5, no. 2, pp. 252–260, 2021, doi: https://doi.org/10.29408/edumatic.v5i2.4113

[9] F. Wahyudi and L. T. Utomo, "Perancangan Security Network Intrusion Prevention System Pada PDTI Universitas Islam Raden Rahmat Malang," *Edumatic: Jurnal Pendidikan Informatika*, vol. 5, no. 1, pp. 60–69, Jun. 2021, doi: https://doi.org/10.29408/edumatic.v5i1.3278

[10] M. Babar and M. Sohail Khan, "ScalEdge: A framework for scalable edge computing in Internet of things–based smart systems," *Int J Distrib Sens Netw*, vol. 17, no. 7, 2021, doi: https://doi.org/10.1177/15501477211035332

[11] M. M. Al-Kofahi, M. Y. Al-Shorman, and O. M. Al-Kofahi, "Toward energy efficient microcontrollers and Internet-of-Things systems," *Computers and Electrical Engineering*, vol. 79, Oct. 2019, doi: http://doi.org/10.1016/j.compeleceng.2019.106457.

[12] Nurjamiyah, "QUERY: Jurnal Sistem Informasi Implementasi Algoritma Affine Cipher untuk Keamanan Data Teks," 2020, doi: https://doi.org/10.58836/query.v4i1.8174.

[13] A. B. Nasution, "MODIFIKASI ALGORITMA AFFINE CIPHER UNTUK MENGAMANKAN DATA," *Jurnal Teknologi Informasi*, vol. 4, no. 2, 2020, doi: https://doi.org/10.36294/jurti.v4i2.1345.

[14] B. J. Dwi, M. Joko Priono, A. Suhendri, B. Dwi Juniansyah, and D. Darwis, "IMPLEMENTASI KOMBINASI AFFINE CIPHER DAN ONE-TIME PAD DALAM," *Jurnal Informatika*, vol. 18, no. 2, 2018, doi: https://doi.org/10.30873/ji.v18i2.

[15] T. Khairani, K. Agung, and A. Kamsyakawuni, "Pengkodean Monoalphabetic Menggunakan Affine Cipher dengan Kunci Diffie-Hellman," *Prisma (Prosiding Seminar Nasional Matematika)*, vol. 4, pp. 553–559, 2021, Accessed: Jun. 07, 2023. [Online]. Available: https://journal.unnes.ac.id/sju/index.php/prisma/article/view/45027

[16] Y. Wang and G. Mogos, "Diffie-hellman Protocol on Raspberry pi," *J Phys Conf Ser*, vol. 1813, no. 1, 2021, doi: https://doi.org/10.1088/1742-6596/1813/1/012047.

[17] A. Alfaris and M. Yuhendri, "Sitem Kendali Dan Monitoring Boost Converter Berbasis GUI (Graphical User Interface) Matlab Menggunakan Arduino," 2020. doi: https://doi.org/10.24036/jtein.v1i2.83.

[18] S. R. Ningsih, A. H. S. Budi, A. T. Nugraha, and T. Winata, "Automatic farmer pest repellent with Arduino ATmega2560 based on sound displacement technique," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, May 2020. doi: https://doi.org/10.1088/1757-899X/850/1/012034.

[19] Budiyanto, R. Primananda, and F. A. Bakhtiar, "Implementasi Enkripsi Vernam Cipher dan Distribusi Kunci Three-Pass Protocol untuk Mengamankan Data Chatting pada ATmega328," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 5, no. 3, pp. 1119–1125, 2021, Accessed: Jun. 07, 2023. [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/8744

[20] A. Nurrohmah, A. Kusyanti, and R. Primananda, "Implementasi Algoritme Grain V1 Dan 128 Bit Pada Arduino Mega 2560," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 4, pp. 1436–1445, 2018, [Online]. Available: http://j-ptiik.ub.ac.id

[21] C. Umam, L. B. Handoko, C. A. Sari, E. H. Rachmawanto, and L. A. R. Hakim,

“Kombinasi Vigenere dan Autokey Cipher dalam Proses Proteksi SMS Berbasis Android,” *Prosiding Sains Nasional dan Teknologi*, vol. 12, no. 1, p. 492, Nov. 2022, doi: https://doi.org/10.36499/psnst.v12i1.7108.

[22] Z. Mufarokah, M. H. H. Ichsan, and A. Kusyanti, “Analisis Performa Algoritme SPECK Pada Arduino Uno,” … *Teknologi Informasi dan Ilmu …*, vol. 3, no. 1, pp. 1085–1092, 2019, Accessed: Jun. 07, 2023. [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5124

[23] F. Nuraeni, Y. H. Agustin, A. E. Purnama, D. Stmik Tasikmalaya, and M. Stmik Tasikmalaya, “IMPLEMENTASI CAESAR CIPHER AND ADVANCED ENCRYPTION STANDARD (AES) PADA PENGAMANAN DATA PAJAK BUMI BANGUNAN,” *Jurnal Ilmiah MATRIK*, vol. 22, no. 2, 2020, doi: https://doi.org/10.33557/jurnalmatrik.v22i2.949.

[24] Rizky F and Anwar, “Implementasi Kriptografi Dengan Metode Advanced Encryption Standard (AES) Untuk Realtime Chat Berbasis Mobile Pada E-Learning Politeknik Negeri Lhokseumawe,” *JAISE : Journal of Artificial Intelligence and Software Engineering*, vol. 1, no. 2, 2021, doi: http://dx.doi.org/10.30811/jaise.v1i2.2520.