Perbandingan Algoritma AES128 dengan SHA256 dalam Kecepatan Enkripsi Pengiriman Data

Rina Setiani*[1], Eldiva Tegar Imananda[2], Wahyu Enggar Wicaksono[3]
Muhammad Aziz Baihaqi[4], Jeki Kuswanto[5]
Universitas Amikom Yogyakarta
e-mail: [1]rinasetiani@students.amikom.ac.id, [2]tegareldiva@students.amikom.ac.id,
[3]wahyuenggar@students.amikom.ac.id, [4]baihaqiaziz9@students.amikom.ac.id,
[5]jeki@.amikom.ac.id
*Penulis Korespondensi

Diterima: 23 Juli 2023; Direvisi: 3 Juli 2024; Disetujui: 3 Juli 2024

Abstrak

Internet of Things bekerja untuk menyediakan berbagai layanan yang menghubungkan sensor dengan aktuator. Pada masa sekarang, layanan IoT telah banyak digunakan diberbagai bidang. Dengan pertumbuhan eksponensial IoT menyebabkan meningkatnya masalah ancaman keamanan. Untuk memastikan keamanan data tersebut, diperlukan algoritma untuk mengenkripsi data. Algoritma yang paling umum dalam melakukan enkripsi adalah Advanced Encryption Standard (AES)[4]. AES adalah salah satu metode yang digunakan untuk mencapai privasi dan kerahasiaan data yang ditransfer melalui berbagai jaringan komputer. Selain AES, terdapat pula SHA256 sebagai algoritma enkripsi. Dalam implementasinya algoritma SHA256 digunakan untuk integritas data dan otentikasi pesan. Metode yang digunakan dalam meneliti perbandingan enkripsi menggunakan algoritma AES (Advanced Encryption Standard) dengan SHA-256 (Secure Hash Algorithm 256-bit), dapat mengadopsi model NDLC (Network Development Life Cycle). Baik AES maupun SHA-256 adalah algoritma yang kuat dan mapan untuk enkripsi data, dan kecepatan keduanya hampir sama dengan perbedaan yang kecil dalam ukuran milisekon.

Kata kunci: *Internet of Things, AES128, SHA256, Enkripsi, Dekripsi*

Abstract

The Internet of Things works to provide various services that connect sensors with actuators. At present, IoT services have been widely used in various fields. With the exponential growth of IoT, the problem of security threats is increasing. To ensure data security, an algorithm is needed to encrypt data. The most common algorithm for encrypting is the Advanced Encryption Standard (AES)[4]. AES is one of the methods used to achieve privacy and confidentiality of data transferred over various computer networks. Apart from AES, there is also SHA256 as an encryption algorithm. In its implementation, the SHA256 algorithm is used for data integrity and message authentication. The method used in researching encryption comparisons using the AES (Advanced Encryption Standard) algorithm with SHA-256 (256-bit Secure Hash Algorithm), can adopt the NDLC (Network Development Life Cycle). Both AES and SHA-256 are strong and well-established algorithms for data encryption, and they are almost identical in speed with a small difference in milliseconds.

Keywords*: Internet of Things, AES128, SHA256, Enkripsi, Dekripsi*

1. PENDAHULUAN

Internet of Things bekerja untuk menyediakan berbagai layanan yang menghubungkan sensor dengan aktuator. Pada masa sekarang, layanan IoT telah banyak digunakan diberbagai bidang. Dengan pertumbuhan eksponensial IoT menyebabkan meningkatnya masalah ancaman keamanan[1]. Ada beberapa alasan yang membuat perangkat yang terhubung ke internet rentan terhadap ancaman dan serangan tersebut yaitu penyerang dapat memiliki akses fisik ke perangkat karena sebagian besar beroperasi tanpa campur tangan manusia. Kemudian penyerang dapat eavesdrop perangkat karena mereka terhubung ke jaringan nirkabel di antara mereka sendiri dan perangkat tidak mendukung algoritma keamanan yang rumit.

Dikarenakan hal tersebut, untuk memastikan keamanan IoT, terdapat berbagai macam hal yang harus terpenuhi, pertama *confidentiality* yaitu melindungi data dari akses yang tidak sah, kemudian *Safety* untuk memastikan bahwa data tidak diubah kecuali mendapatkan izin dan terakhir a*vailabilit*y untuk memastikan akses data sesuai kebutuhan [3].

Untuk memastikan keamanan data tersebut, diperlukan algoritma untuk mengenkripsi data. Algoritma yang paling umum dalam melakukan enkripsi adalah *Advanced Encryption Standard* (AES) [4]. AES adalah salah satu metode yang digunakan untuk mencapai privasi dan kerahasiaan data yang ditransfer melalui berbagai jaringan komputer. Algoritma ini mencakup struktur khusus untuk enkripsi dan dekripsi data sensitif [5]. AES memungkinkan untuk diimplementasikan dalam perangkat lunak atau perangkat keras. Enkripsi AES-128 dapat diimplementasikan pada ESP32 yang dirancang untuk mendukung pekerjaan proyek Internet of things[6].

Selain AES, terdapat pula SHA256 sebagai algoritma enkripsi. Dalam implementasinya algoritma SHA256 digunakan untuk integritas data dan otentikasi pesan [7]. SHA-256 dirancang oleh *The National Institute of Standars and Technology* (NIST) pada tahun 2002. SHA-256 menerima input data dengan panjang maksimal 264 bit dan menghasilkan output berupa message digest atau nilai hash dengan panjang 256 bit [8]. SHA-256 menggunakan beberapa fungsi logika diantaranya operasi AND, OR, XOR, SHIFT, dan ROTATE. SHA-256 beroperasi pada MD4, MD5, dan SHA-1 algoritma [9].

Dalam mengirim data, dapat menggunakan HTTP protocol. HTTP mendukung permintaan/respons arsitektur Web RESTful. Serupa dengan CoAP, HTTP menggunakan *Universal Resource Identifier* (URI) [10]. Server mengirimkan data melalui URI dan klien menerima data melalui URI tertentu. HTTP adalah protokol berbasis teks dan tidak menentukan ukuran *heade*r dan muatan pesan, melainkan bergantung pada server web atau teknologi pemrograman. HTTP menggunakan TCP sebagai protokol *transport default* dan TLS/SSL untuk keamanan. Dengan demikian, komunikasi antara client dan server bersifat *connection-oriented* [11].

2. METODOLOGI PENELITIAN

Metode yang digunakan dalam meneliti perbandingan enkripsi menggunakan algoritma AES (*Advanced Encryption Standard*) dengan SHA-256 (*Secure Hash Algorithm* 256-bit), dapat mengadopsi model NDLC (*Network Development Life Cycle*) yang telah dijelaskan sebelumnya dan menyesuaikannya dengan fokus pada perbandingan enkripsi ini[12]. Berikut adalah deskripsi tahap-tahap dalam NDLC yang dikhususkan untuk penelitian perbandingan enkripsi AES dan SHA-256 [13], [14].

2.1. Analisis

Pada tahap analisis, tim peneliti merumuskan pertanyaan penelitian, tujuan, dan kebutuhan dalam membandingkan enkripsi menggunakan AES dan SHA-256. Hal tersebut termasuk mendefinisikan data apa yang akan dienkripsi, apa kunci enkripsi yang akan digunakan,

dan konteks penggunaan enkripsi misalnya, penyimpanan data, pengiriman data melalui jaringan, dan sebagainya.

2.2. Desain

Pada tahap desain, peneliti merancang arsitektur sistem yang akan digunakan untuk implementasi enkripsi AES dan SHA-256. Diantaranya menentukan bagaimana data akan dienkripsi dan dekripsi menggunakan masing-masing algoritma. Selain itu, juga mempertimbangkan parameter dan opsi konfigurasi yang relevan untuk setiap algoritma, seperti mode enkripsi, ukuran kunci, dan padding.

2.3. Implementasi

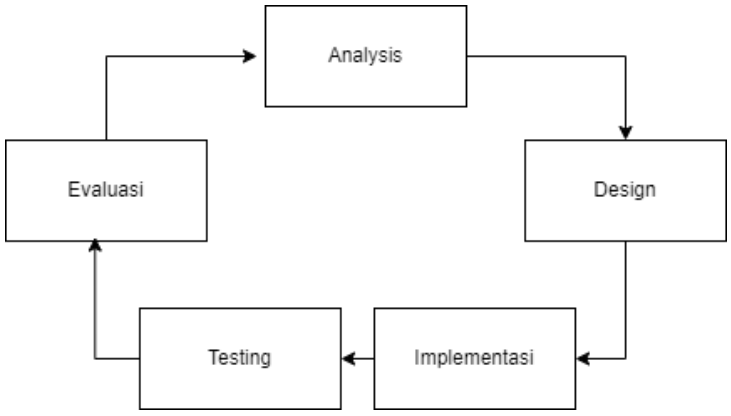
Tahap implementasi melibatkan penerapan desain yang telah dirancang sebelumnya. Peneliti mengimplementasikan enkripsi dan dekripsi menggunakan algoritma AES dan SHA-256 dalam bahasa pemrograman yang relevan, dalam hal ini adalah python. Data yang akan diuji dan perangkat lunak enkripsi dikonfigurasi sesuai dengan rencana dari tahap Analisis dan Desain.

2.4. Pengujian

Pada tahap pengujian, Peneliti melakukan serangkaian uji coba untuk membandingkan performa enkripsi dan dekripsi menggunakan AES dan SHA-256. Pengujian ini mencakup pengukuran waktu eksekusi, kinerja dalam mengenkripsi dan mendekripsi data berukuran berbeda, dan analisis keamanan kunci yang dihasilkan oleh masing-masing algoritma.

2.5. Evaluasi

Tahap evaluasi adalah saat peneliti menganalisis hasil dari pengujian dan perbandingan antara enkripsi AES dan SHA-256. Hasil pengujian dan analisis digunakan untuk membandingkan kekuatan dan kelemahan dari kedua algoritma. Tim juga mengevaluasi kecocokan masing-masing algoritma dengan kebutuhan dan tujuan penelitian. Hasil evaluasi ini akan membantu dalam menarik kesimpulan dan memberikan rekomendasi mengenai algoritma mana yang lebih cocok untuk kasus penggunaan tertentu.

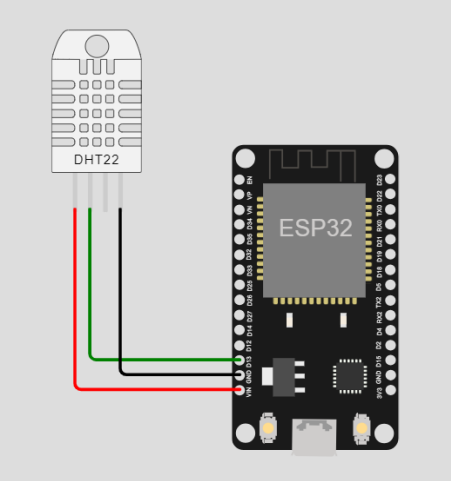


Gambar 1. Siklus NDLC

Gambar 1 diatas menggambarkan siklus proses metodologi yang digunakan. Model NDLC yang telah disesuaikan ini memungkinkan tim peneliti untuk menyelidiki secara sistematis perbandingan enkripsi menggunakan AES dan SHA-256 dengan pendekatan yang terstruktur dan ilmiah [15], [16]. Hal ini dapat membantu dalam mendapatkan wawasan yang lebih dalam tentang karakteristik dan performa masing-masing algoritma dalam konteks penelitian yang relevan [17].

3. HASIL DAN PEMBAHASAN

Dalam penelitian ini, menggunakan DHT22 sebagai sensor dan ESP32 sebagai *mikrokontroler*. *Wiring* untuk lebih jelasnya dapat dilihat pada gambar 2.



Gambar 2. *Wiring hardware*

Untuk membaca data sensor peneliti mendefinisikan function, untuk lebih jelasnya dapat dilihat pada gambar 3.

```
pin_dht22 = machine.Pin(13)

def read_dht():
    d = dht.DHT22(pin_dht22)
    d.measure()
    return d.temperature(), d.humidity()
```

Gambar 3. Function read DHT22

Kemudian, untuk memanggil fuction tersebut, peneliti memanfaatkan looping sehingga program akan terus terulang apabila terdapat kondisi True. Untuk lebih jelasnya dapat dilihat pada gambar 4.

```
while True:
  temperature, humidity = read_dht()

  print(f'Temp: {temperature} - Humi: {humidity}')
  time.sleep(2)
```

Gambar 4. Looping untuk memanggil *fuction read* DHT22

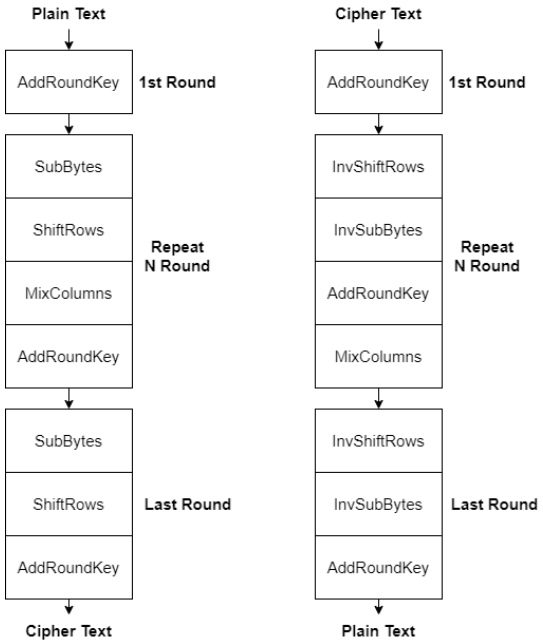
```
import hashlib
import utime
import uos
from ucryptolib import aes
from urandom import getrandbits
import utime
```

Gambar 5. Modul yang digunakan dalam enkripsi AES128 dan SHA256

Setelah DHT22 berhasil dibaca, maka akan dialakukan enkripsi supaya data yang terbaca tidak dapat dirubah selama dalam proses pengiriman data. Dalam kasus ini, enkripsi

menggunakan SHA256 serta AES128. Sebelum mendeklarasikan coding enkripsi, diharuskan untuk mengimport modul terlebih dahulu. Modul yang digunakan dapat dilihat pada Gambar 5.

Dalam proses enkripsi AES128, terdapat beberapa step yang harus dilewati, antara lain adalah *AddRoundKey*, *SubBytes*, *ShiftRows*, dan *MixColumns*. Untuk alur yang lebih jelasnya dapat dilihat pada gambar 6.



Gambar 6. Flowchart ekripsi dan dekripsi AES128

Kemudian dari alur tersebut dapat dirubah menjadi coding seperti pada Gambar 7.

```
def generate_key():
    return bytes([getrandbits(8) for _ in range(16)])

def encrypt(data, key):
    start_time = utime.ticks_us()

    iv = uos.urandom(16)

    MODE_CBC = 2
    cipher = aes(key, MODE_CBC, iv)

    print("Using AES{}-CBC cipher \n".format(len(key * 8)))

    ct_bytes = data + " " * (16 - len(data) % 16)

    encrypted = cipher.encrypt(ct_bytes)
    decipher = aes(key, MODE_CBC, iv)
    decrypted = decipher.decrypt(encrypted)
```

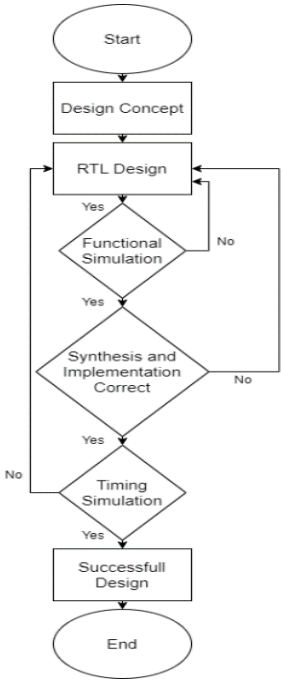
Gambar 7. Coding AES128

Dimana penjelasan dari gambar 7 tersebut yaitu generate_key(): Fungsi ini bertujuan untuk menghasilkan kunci enkripsi yang digunakan dalam algoritma AES. Kunci yang dihasilkan memiliki panjang 128-bit (16 byte). encrypt(data, key): Fungsi ini menerima dua argumen, yaitu data yang merupakan data yang akan dienkripsi dan key yang merupakan kunci enkripsi. Fungsi ini melakukan proses enkripsi dengan mode CBC menggunakan algoritma AES. Pertama, fungsi ini menginisialisasi sebuah vector inisialisasi (IV) dengan nilai acak (random) sepanjang 128-bit (16 byte) menggunakan fungsi uos.urandom(16). IV digunakan untuk memastikan bahwa blok pertama dari data yang dienkripsi tidak selalu sama meskipun data yang dienkripsi sama. Selanjutnya, fungsi menginisialisasi objek cipher dengan mode AES-CBC menggunakan kunci

(key) dan IV yang telah dihasilkan. Fungsi mencetak informasi tentang jenis kunci yang digunakan, yaitu AES dengan panjang kunci 128-bit. Data data di-padding dengan spasi (ASCII 32) sehingga panjangnya menjadi kelipatan 16 byte, karena AES bekerja dengan blok data 16-byte. Data yang telah dipadding kemudian dienkripsi dengan menggunakan kunci dan IV menggunakan metode cipher.encrypt(ct_bytes). Hasil enkripsi disimpan dalam variabel encrypted. Selanjutnya, fungsi menginisialisasi objek decipher dengan mode AES-CBC menggunakan kunci (key) dan IV yang sama yang digunakan pada saat enkripsi. Data yang telah dienkripsi (encrypted) kemudian didekripsi kembali menggunakan metode decipher.decrypt(encrypted). Hasil dekripsi disimpan dalam variabel decrypted. Hasil dari enkripsi AES128 dapat dilihat pada gambar 8.

Gambar 8. Hasil enkripsi AES128

Sedangkan SHA256 memiliki alur yang lebih sederhana, yaitu design concept, rtl design, functional simulation, synthesis and implementation correct, timing simulation, dan successful design. Untuk lebih jelasnya, alaur SHA256 dapat dilihat pada gambar 9 serta coding SHA256 dapat dilihat pada gambar 10.



Gambar 9. Flowchart SHA256

```
def encrypt(data):
    start_time = utime.ticks_us()

    sha256_hash = hashlib.sha256()
    sha256_hash.update(data.encode('utf-8'))
```

Gambar 10. Coding SHA256

Pada gambar 10 menjelaskan encrypt(data): Fungsi ini bertujuan untuk menghasilkan nilai hash dari data yang diberikan menggunakan algoritma SHA-256 (Secure Hash Algorithm 256-bit). SHA-256 adalah fungsi hash kriptografis yang mengambil data sebagai input dan menghasilkan hash dengan panjang 256-bit (32 byte) sebagai output. Dalam kode ini, proses hashing dilakukan pada data yang diberikan. Fungsi ini menerima satu argumen, yaitu data, yang merupakan data yang akan di-hash. Pertama, fungsi menginisialisasi objek sha256_hash dari modul hashlib untuk menggunakan algoritma SHA-256. Selanjutnya, sha256_hash.update(data.encode('utf-8')) digunakan untuk meng-update objek hash dengan data yang diberikan. Sebelumnya, data diubah menjadi representasi bytes menggunakan data.encode('utf-8'). Hasil dari enkripsi menggunakan algoritma SHA256 dapat dilihat pada gambar 11.

```
f575e34ea3f75bf5810dbacee717bb399a1e8ed8b55b357a61315ac275a14157
0.29
24.0 40.0
f575e34ea3f75bf5810dbacee717bb399a1e8ed8b55b357a61315ac275a14157
0.326
24.0 40.0
f575e34ea3f75bf5810dbacee717bb399a1e8ed8b55b357a61315ac275a14157
0.255
24.0 40.0
f575e34ea3f75bf5810dbacee717bb399a1e8ed8b55b357a61315ac275a14157
0.254
24.0 40.0
f575e34ea3f75bf5810dbacee717bb399a1e8ed8b55b357a61315ac275a14157
0.253
24.0 40.0
```

Gambar 11. Hasil enkripsi SHA256

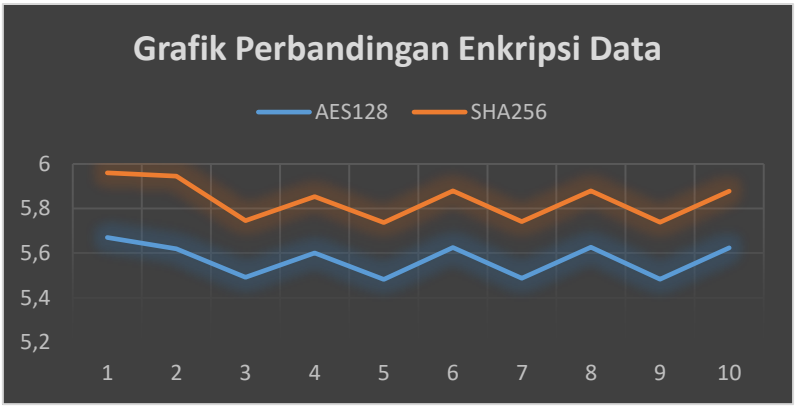
Dari penggunaan enkripsi AES128 dan SHA256 diatas, didapatkan perbedaan waktu enkripsi dari kedua algoritma tersebut. Algoritma SHA256 lebih cepat 5,306 ms dalam mengenkripsi data dibandingkan dengan algoritma AES128. Hal tersebut dapat dilihat dalam tabel 1.

Tabel 1. Perbandingan kecepatan mengenkripsi data dalam ms

AES128	SHA256
5,670	0,290
5,619	0,326
5,491	0,254
5,601	0,253
5,482	0,255
5,625	0,254
5,487	0,254

AES128	SHA256
5,626	0,253
5,483	0,255
5,624	0,254

Dari tabel 1 tersebut, maka didapatkan grafik yang dapat dilihat pada gambar 12.



Gambar 12. Grafik perbandingan enkripsi data

Dari grafik pada gambar 12 diatas dapat disimpulkan bahwa sebenarnya kecepatan kedua algoritma dalam mengenkripsi data hamper sama, hanya saja terdapat perbedaan dalam ukuran milisecon. Dari grafik di atas dapat disimpulkan bahwa sebenarnya kecepatan kedua algoritma dalam mengenkripsi data hampir sama, hanya saja terdapat perbedaan dalam ukuran milisekon. Algoritma AES dengan mode Cipher Block Chaining (CBC) cenderung membutuhkan waktu yang sedikit lebih lama dibandingkan dengan algoritma SHA-256 untuk melakukan enkripsi data. Namun, perbedaan waktu tersebut biasanya hanya berada pada skala milisekon, yang mungkin tidak signifikan untuk banyak aplikasi. Penting untuk dicatat bahwa kecepatan enkripsi dapat dipengaruhi oleh berbagai faktor, termasuk panjang data yang dienkripsi, tingkat kompleksitas algoritma, dan kualitas implementasi dari kedua algoritma tersebut. Dalam penggunaan nyata, pemilihan algoritma enkripsi harus mempertimbangkan keamanan, kecepatan, dan sumber daya perangkat yang tersedia.

Penting juga untuk menyadari bahwa keamanan adalah pertimbangan utama dalam pemilihan algoritma enkripsi. Meskipun AES dan SHA-256 adalah algoritma yang mapan dan banyak digunakan dalam dunia kriptografi, kemampuan mereka untuk melindungi data tergantung pada penggunaan kunci yang kuat dan implementasi yang benar. Dalam skenario tertentu, terutama ketika ada kebutuhan untuk mengenkripsi dan mendekripsi data dalam waktu nyata dengan sumber daya yang terbatas, mungkin ada pertimbangan untuk menggunakan algoritma yang lebih efisien dan khusus untuk lingkungan tersebut.

4. KESIMPULAN

Baik AES maupun SHA-256 adalah algoritma yang kuat dan mapan untuk enkripsi data, dan kecepatan keduanya hampir sama dengan perbedaan yang kecil dalam ukuran milisekon. Pemilihan algoritma harus dilakukan dengan mempertimbangkan keamanan, efisiensi, dan kebutuhan khusus dari sistem atau aplikasi yang digunakan.

DAFTAR PUSTAKA

[1] N. R. A. Abosata, A. H. Kemp, and M. Razavi, “Secure Smart-Home Application Based

on IoT- CoAP protocol," *2019 Sixth Int. Conf. Internet Things Syst. Manag. Secur.*, pp. 13–17, 2020.

[2] F. Z. Hamza, S. EL Aidi, A. Bajit, S. Beloualid, H. Chaoui, and A. Tamtaoui, "Advanced IoT Network Topologies to Optimize Medical Monitoring Platforms based on a Constrained and Secured IOT Application Protocol CoAP," *E3S Web Conf.*, vol. 351, pp. 3–7, 2022, doi: 10.1051/e3sconf/202235101011.

[3] M. El-hajj, H. Mousawi, and A. Fadlallah, "Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform †," *Futur. Internet*, vol. 15, no. 2, pp. 1–5, 2023, doi: 10.3390/fi15020054.

[4] N. Kheshaifaty and A. Gutub, "Engineering Graphical Captcha and AES Crypto Hash Functions for Secure Online Authentication," *J. Eng. Res.*, 2021, doi: 10.36909/jer.13761.

[5] A. R. Chowdhury, J. Mahmud, A. R. M. Kamal, and M. A. Hamid, "MAES: Modified advanced encryption standard for resource constraint environments," *2018 IEEE Sensors Appl. Symp. SAS 2018 - Proc.*, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1109/SAS.2018.8336747.

[6] Y. Liu *et al.*, "Design of password encryption model based on AES algorithm," *Proc. 2019 IEEE 1st Int. Conf. Civ. Aviat. Saf. Inf. Technol. ICCASIT 2019*, pp. 385–389, 2019, doi: 10.1109/ICCASIT48058.2019.8973003.

[7] B. Barani Sundaram, M. K. Mishra, D. Thirumoorthy, U. Rastogi, and B. Pattanaik, "ZHLS Security Enhancement by integrating SHA256, AES, DH in MANETS," *J. Phys. Conf. Ser.*, vol. 1964, no. 4, 2021, doi: 10.1088/1742-6596/1964/4/042003.

[8] M. Husni *et al.*, "Security audit in cloud-based server by using encrypted data AES -256 and SHA-256," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 830, no. 3, 2020, doi: 10.1088/1757-899X/830/3/032015.

[9] H. Blankson and R. Chattamvelli, "A Symmetric Scheme for Securing Data in Cyber-Physical Systems/IoT Sensor-based Systems based on AES and SHA256," *Int. J. Comput. Appl.*, vol. 184, no. 24, pp. 12–17, 2022, doi: 10.5120/ijca2022922278.

[10] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," *2017 IEEE Int. Symp. Syst. Eng. ISSE 2017 - Proc.*, 2017, doi: 10.1109/SysEng.2017.8088251.

[11] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–29, 2019, doi: 10.1145/3292674.

[12] F. Naim, R. R. Saedudin, and U. Y. K. S. Hediyanto, "Analysis of Wireless and Cable Network Quality-of-Service Performance At Telkom University Landmark Tower Using Network Development Life Cycle (Ndlc) Method," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 4, pp. 1033–1044, 2022, doi: 10.29100/jipi.v7i4.3192.

[13] T. Sanjaya and D. Setiyadi, "1-10 Teknik Informatika; STMIK Bina Insani," *Rawa Panjang Bekasi Timur*, vol. 4, no. 1, p. 17114, 2019.

[14] D. Siswanto, G. Priyandoko, N. Tjahjono, R. S. Putri, N. B. Sabela, and M. I. Muzakki, "Development of Information and Communication Technology Infrastructure in School using an Approach of the Network Development Life Cycle Method," *J. Phys. Conf. Ser.*, vol. 1908, no. 1, 2021, doi: 10.1088/1742-6596/1908/1/012026.

[15] A. Mardiyono, W. Sholihah, and F. Hakim, "Mobile-based Network Monitoring System Using Zabbix and Telegram," *2020 3rd Int. Conf. Comput. Informatics Eng. IC2IE 2020*, pp. 473–477, 2020, doi: 10.1109/IC2IE50715.2020.9274582.

[16] K. Rianafirin and M. T. Kurniawan, "Design network security infrastructure cabling using network development life cycle methodology and ISO/IEC 27000 series in Yayasan Kesehatan (Yakes) Telkom Bandung," *Proc. 2017 4th Int. Conf. Comput. Appl. Inf.*

Process. Technol. CAIPT 2017, vol. 2018-January, pp. 1–6, 2018, doi: 10.1109/CAIPT.2017.8320681.

[17] F. Eko Nugroho and Y. Daniarti, "Rancang Bangun Qos (Quality of Service) Jaringan Wireless Local Area Network Menggunakan Metode Ndlc (Network Development Life Cycle) Di Pt Trimitra Kolaborasi Mandiri (3Kom)," *JIKA (Jurnal Inform.*, vol. 5, no. 1, p. 79, 2021, doi: 10.31000/jika.v5i1.3970.