Media of Computer science (MCs)
Vol.x, No.x, July xxxx, pp. 121~134
ISSN (print): 3063-4822, ISSN (online): 3063-4997
DOI: 10.69616/mcs.v2i2.258

Implementation of the Advanced Encryption Standard (AES) Algorithm on Access Code QR Codes in a Smart Door System

Ihsan Azhar Rhamadan*, Kasliono, Syamsul Bahri
Faculty of Mathematics and Natural Sciences , FMIPA Untan, Pontianak, Indonesia
e-mail: ***h1051211115@student.untan.ac.id**, kasliono@siskom.untan.ac.id,
syamsul.bahri@siskom.untan.ac.id

Abstract

This research aims to design a smart door system based on QR codes with increased security using the AES-256-CBC algorithm. With AES-256-CBC encryption, the access code data sent from the device to the server is no longer in plaintext form, making it more secure from Man-in-the-Middle (MitM) attacks such as sniffing. The system utilizes ESP32-CAM to read QR codes and NodeMCU ESP32 to encrypt access data using a key hashed with SHA-256 and a 16-byte IV before being sent to the server for verification. Performance testing shows that the average delay increased from 66.4 ms to 67.46 ms after AES implementation, with a maximum value of 82 ms which is still in accordance with the TIPHON standard. On the throughput side, without AES the average is only 5.23 kbps due to shorter data, while with AES it increases to 7.30 kbps with a peak of 8 kbps due to the longer ciphertext size. The response time test recorded an average of 3830.76 ms and a maximum of 6086 ms, spanning the process from sending the code to opening the solenoid. Despite variations due to internet connection quality, delay spikes do not significantly impact system performance, indicating that the system remains secure and stable for IoT-based access control.

Keywords—*AES-256-CBC, QR Code, Encryption, Delay, Throughput*

Abstrak

Penelitian ini bertujuan merancang sistem pintu cerdas berbasis kode QR dengan peningkatan keamanan menggunakan algoritma AES-256-CBC. Dengan enkripsi AES-256-CBC, data kode akses yang dikirim dari perangkat ke server tidak lagi dalam bentuk plaintext, sehingga lebih aman dari serangan Man-in-the-Middle (MitM) seperti sniffing. Sistem memanfaatkan ESP32-CAM untuk membaca kode QR dan NodeMCU ESP32 untuk mengenkripsi data akses memakai kunci yang di-hash dengan SHA-256 dan IV 16 byte sebelum dikirim ke server untuk verifikasi. Pengujian performa menunjukkan rata-rata delay meningkat dari 66,4 ms menjadi 67,46 ms setelah implementasi AES, dengan nilai tertinggi 82 ms yang masih sesuai standar TIPHON. Pada sisi throughput, tanpa AES rata-rata hanya 5,23 kbps karena data lebih pendek, sedangkan dengan AES meningkat menjadi 7,30 kbps dengan puncak 8 kbps akibat ukuran ciphertext yang lebih panjang. Uji response time mencatat rata-rata 3830,76 ms dan maksimum 6086 ms, mencakup proses pengiriman kode hingga pembukaan solenoid. Meskipun ada variasi akibat kualitas koneksi internet, lonjakan delay tidak berdampak signifikan terhadap kinerja sistem, menunjukkan bahwa sistem tetap aman dan stabil untuk kontrol akses berbasis IoT.

Kata kunci—*AES-256-CBC, QR Code, Enkripsi, Delay, Throughput*

Received November 6th,2025; Revised February 1st, 2026; Accepted February 2nd, 2026

1. PENDAHULUAN

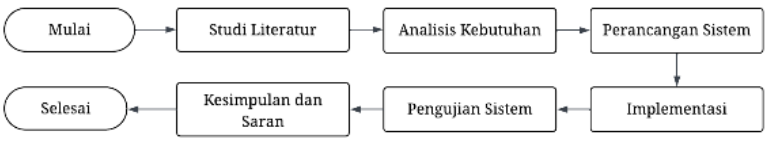
Sistem kontrol akses berbasis QR Code merupakan solusi modern meningkatkan keamanan untuk mengelola hak akses pada lingkungan fisik seperti pintu kantor, ruang kerja, atau fasilitas penting lainnya [1]. Keamanan memungkinkan setiap orang menjalankan rutinitas sehari-hari dengan baik karena memberikan kenyamanan dan ketenangan [2]. Hal ini karena penggunaan kunci konvensional sudah rentan terhadap perampokan, seperti pembobolan dan penduplikatan kunci oleh individu yang tidak bertanggung jawab [3]. Namun, pengiriman kode akses tanpa enkripsi masih berisiko terkena serangan *Man-in-the-Middle* (MitM). Oleh karena itu, dibutuhkan penguatan keamanan dengan algoritma enkripsi seperti AES-256-CBC yang memiliki kunci 256 bit. Penelitian ini menggunakan ESP32-CAM untuk membaca QR Code dan NodeMCU ESP32 untuk mengirim data terenkripsi ke *server*, dengan fokus pada evaluasi performa serta peningkatan keamanan sistem setelah integrasi AES-256-CBC.

Penelitian sebelumnya yang berjudul *"Sistem Pintu Cerdas dengan QR Code berbasis Internet of Things sebagai Penerapan Edge Computing"* [4] mengembangkan sistem pintu cerdas berbasis IoT menggunakan QR Code sebagai metode akses, dengan penerapan teknologi *Edge Computing* untuk mengurangi ketergantungan terhadap *Cloud Server* dan menekan latensi. Sistem tersebut memproses verifikasi QR Code secara lokal di perangkat *edge* sehingga respons sistem menjadi lebih cepat. Namun, meskipun QR Code memberikan kemudahan akses, teknologi ini masih rentan terhadap serangan seperti *Man-in-the-Middle* (MitM) akibat tidak adanya mekanisme autentikasi bawaan [5]. Untuk mengatasi celah ini, diperlukan penerapan algoritma enkripsi seperti *Advanced Encryption Standard* (AES) guna menjaga kerahasiaan dan integritas data. Penelitian lain berjudul *"Penerapan Algoritma Advanced Encryption Standard (AES) Untuk Pengamanan File Pada Aplikasi Berbasis Web"* [6]menunjukkan bahwa AES mampu melindungi data dari serangan *sniffing* dan manipulasi, dengan implementasi berbasis PHP dan MySQL yang memungkinkan proses enkripsi-dekripsi *file* secara *online*. Sementara itu, studi *"Peningkatan Keamanan Data End-to-End Smart Door Menggunakan Advanced Encryption Standard"* [7]berfokus pada pengamanan sistem *Smart Door* berbasis IoT terhadap potensi penyusupan dan *masquerading* dengan menerapkan AES untuk enkripsi *end-to-end* serta validasi integritas data menggunakan *Truncated Decimal-converted SHA-1 checksum*. Berdasarkan berbagai penelitian tersebut, terlihat bahwa penerapan AES efektif meningkatkan keamanan sistem akses digital, termasuk pada penerapan kontrol pintu cerdas berbasis QR Code.

Berdasarkan dari penelitian sebelumnya, maka diangkat sebuah penelitian yang berjudul "Implementasi Algoritma *Advanced Encryption Standard* (AES) pada Kode Akses QR Code di Sistem Pintu Cerdas". Penelitian ini bertujuan untuk mengeksplorasi potensi implementasi algoritma *Advanced Encryption Standard* (AES) 256 bit untuk meningkatkan keamanan sistem pintu cerdas menggunakan QR Code.

2. METODE PENELITIAN

Metode penelitian adalah prosedur atau cara-cara yang dilakukan untuk mengumpulkan dan menganalisis data serta menjawab pertanyaan atau hipotesis yang telah ditetapkan dalam penelitian [8], [9], [10], [11]. Adapun isi dari metode penelitian yang dilakukan yaitu diagram alir, studi literatur, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, kesimpulan dan saran. Diagram alir dari penelitian dapat dilihat pada Gambar *1*.



Gambar 1 Diagram Alir Penelitian

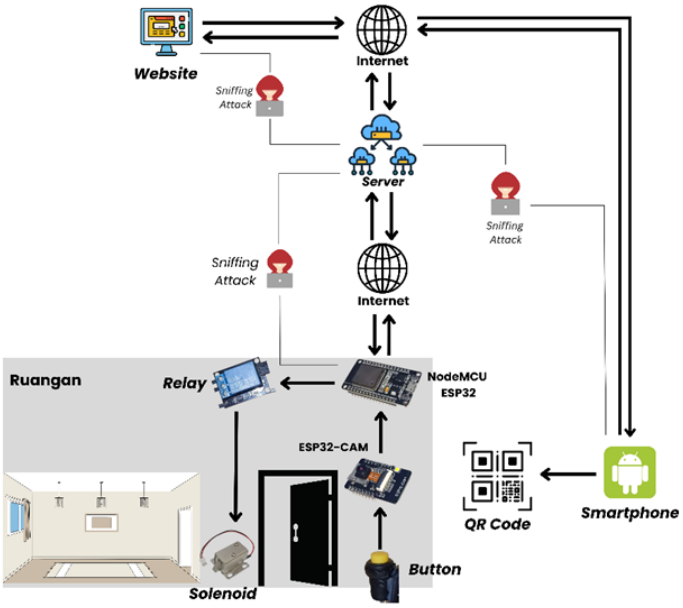
Penelitian ini diawali dengan studi literatur yang bertujuan mengumpulkan referensi mengenai sistem pintu cerdas berbasis QR Code, keamanan data, serta penerapan algoritma kriptografi AES-256-CBC dari berbagai sumber seperti jurnal ilmiah, buku, dan artikel terkait. Selanjutnya dilakukan analisis kebutuhan untuk menentukan komponen yang diperlukan, baik perangkat keras seperti NodeMCU ESP32, ESP32-CAM, *relay*, solenoid, *buzzer*, dan tombol, maupun perangkat lunak seperti Arduino IDE, HTTP, MySQL, dan PHP. Pada tahap perancangan, sistem dibangun dengan mengintegrasikan NodeMCU ESP32 dan ESP32-CAM, di mana data hasil pemindaian QR Code dienkripsi menggunakan algoritma AES-256-CBC sebelum dikirim ke *server* melalui protokol HTTP agar aman dari serangan *Man-in-the-Middle* (MitM). Implementasi sistem dilakukan dengan ESP32-CAM sebagai pemindai QR Code dan NodeMCU ESP32 sebagai pengolah serta pengirim *ciphertext* ke *server* untuk verifikasi, kemudian *server* memberikan respons untuk mengaktifkan solenoid atau *buzzer*. Tahap pengujian dilakukan untuk mengevaluasi kinerja sistem, termasuk memastikan data terenkripsi selama transmisi, serta mengukur *delay*, *throughput*, dan respons *time* dari proses verifikasi hingga pintu terbuka. Tahap akhir penelitian berisi kesimpulan yang menjawab rumusan masalah berdasarkan hasil pengujian serta saran untuk pengembangan sistem di masa mendatang.

2.3 Perancangan Sistem

Sistem ini menggunakan NodeMCU yang terhubung dengan ESP32-CAM sebagai mikrokontroler. Data kode akses dari sensor dikirimkan secara *real-time* ke *server* melalui protokol HTTP ke *database* MySQL dalam bentuk terenkripsi menggunakan algoritma AES-256 dengan mode *Cipher Block Chaining* (CBC). Dengan demikian, pertukaran data antara perangkat dan *server* lebih terlindungi dari serangan *Man-in-the-Middle* (MitM) seperti *sniffing.*

2. 3.1 Arsitektur Sistem

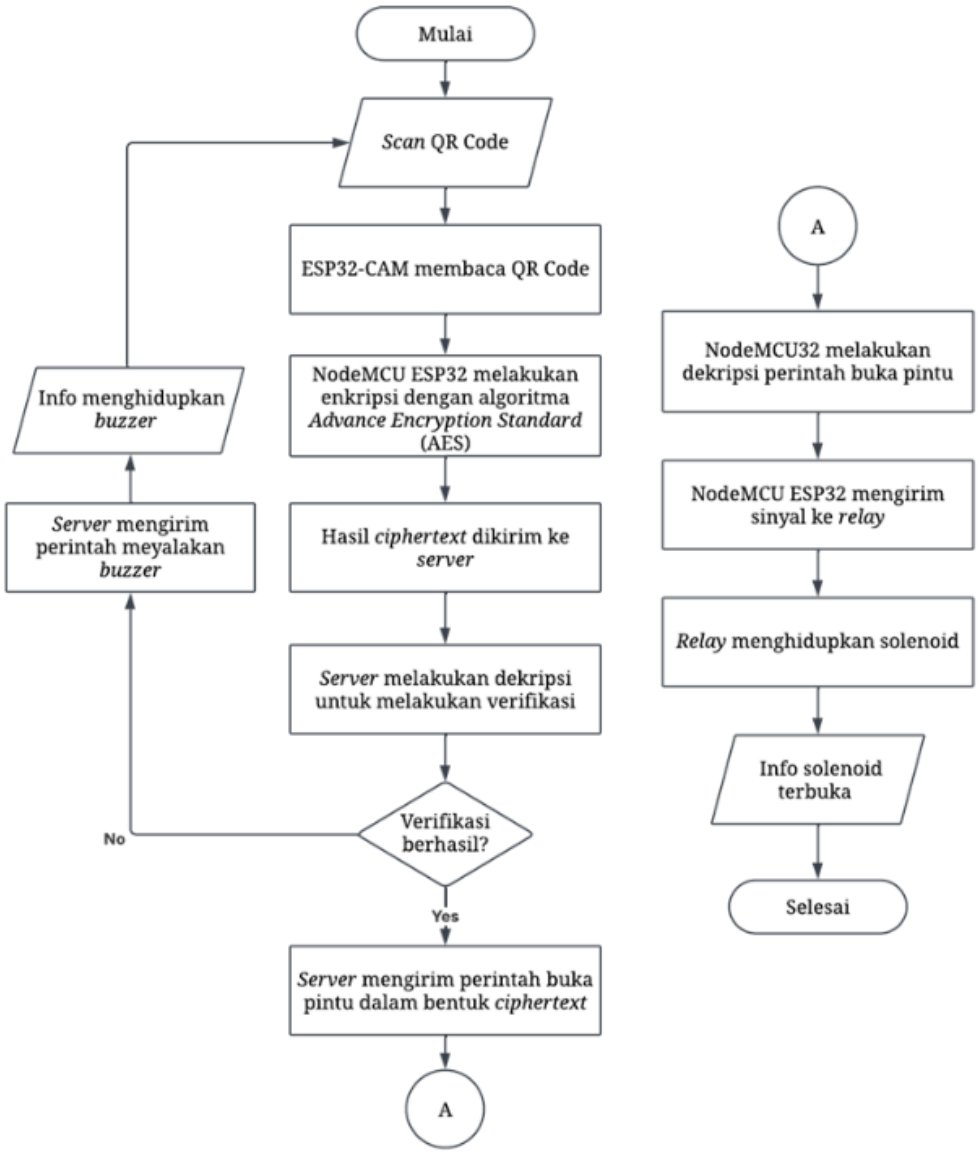
Sistem yang dibuat menggunakan NodeMCU ESP32 yang terhubung juga dengan ESP32-CAM sebagai mikrokontroler. Data kode akses yang didapatkan dari sensor dikirimkan ke *server* dengan menggunakan protokol HTTP(*Hypertext Transfer Protocol*) secara *real-time* ke *database* MySQL dalam bentuk yang sudah terenkripsi. Algoritma *Advanced Encryption Standard* (AES) itu sendiri menggunakan versi 256 bit. Dengan begitu, data kode akses yang dikirim antara perangkat dan *server* dapat lebih terlindungi dari serangan *Man-in-the-Middle* (MitM), seperti *sniffing*. Mode yang digunakan juga adalah CBC sehingga menggunakan penambahan kunci IV. Arstitektur sistem yang dimana *sniffing* dapat melakukan penyerangan dapat dilihat pada Gambar *2*.



Gambar 2 Arsitektur Sistem

2. 3.2 Perancangan Sistem Verifikasi Kode Akses

Kode akses yang didapat dilakukan enkripsi dengan algoritma *Advanced Encryption Standard* (AES) pada NodeMCU ESP32 sebelum dikirim ke *server* untuk di verifikasi dan data yang dikirimkan ke *server* menggunakan protokol HTTP. NodeMCU ESP32 adalah sistem berdaya rendah pada seri *chip* (SoC) yang memiliki *Wi-Fi* dan dual mode *Bluetooth* [12]. Di *server*, kode akses yang dalam bentuk *ciphertext* tersebut didekripsi kembali. Sehingga kode akses dalam bentuk *plaintext* yang akan dicocokkan dengan data yang ada di *database*. Apabila verifikasi berhasil, *server* akan mengirimkan perintah ke *relay* untuk membuka solenoid. Sedangkan, jika verifikasi gagal, *server* mengirimkan perintah untuk menghidupkan *buzzer* seperti pada diagram alir Gambar 3.

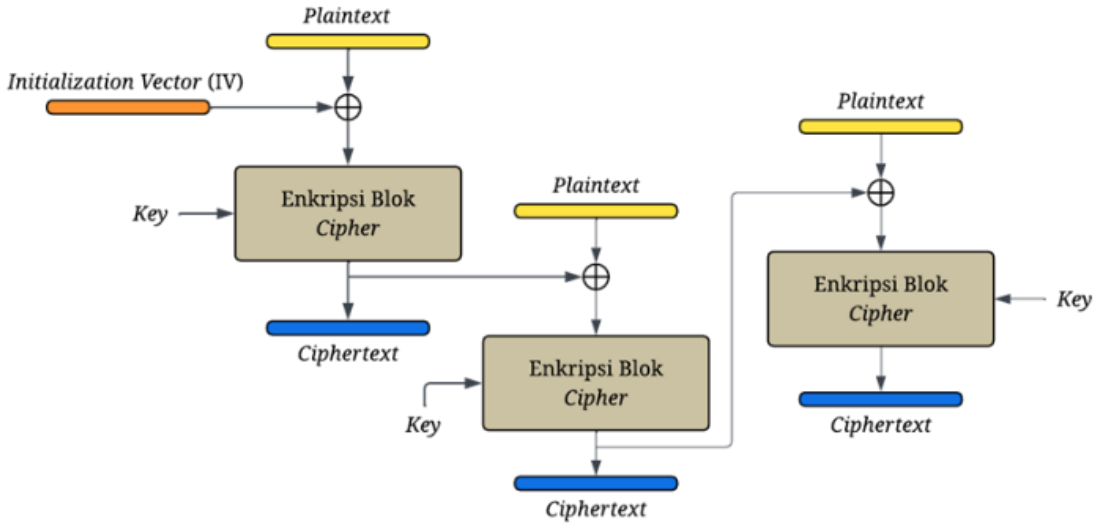


Gambar 3 Proses Verifikasi Kode Akses

2. 3.4 Perancangan Algoritma Advanced Encryption Standard (AES)

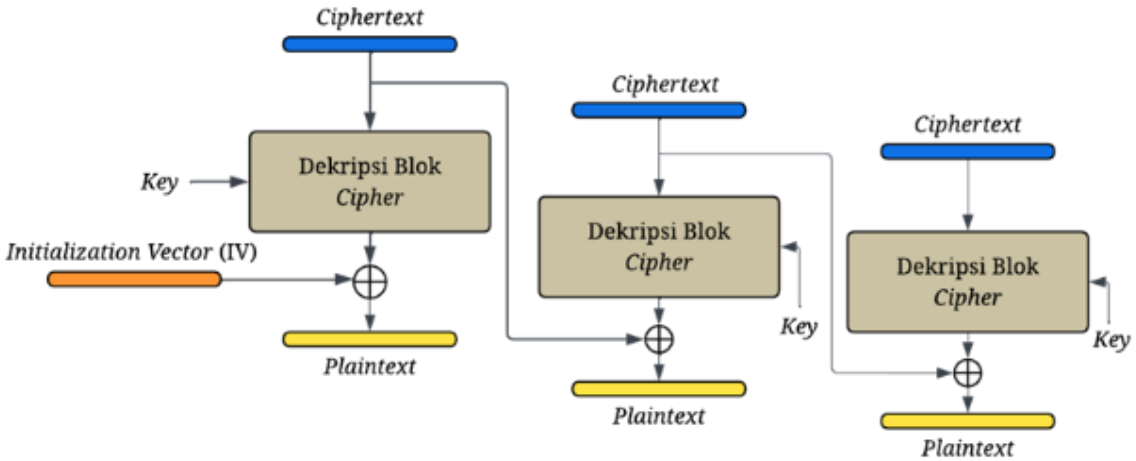
Pada sistem yang dibuat, menggunakan algoritma AES-256 dengan mode *Cipher Block Chaining* (CBC) untuk proses enkripsi dan dekripsi nya. *Advanced Encryption Standard* (AES) adalah algoritma kriptografi yang dapat mengenkripsi dan mendekripsi data dengan panjang kunci yang berbeda, seperti 128 bit, 192 bit, dan 256 bit. Algoritma ini memiliki empat jenis

transformasi *byte* dalam proses enkripsinya, yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey [13]. Pada sistem yang dibuat, menggunakan algoritma AES-256 untuk proses enkripsi dan dekripsi nya. AES tetap tahan terhadap serangan pencarian kunci menyeluruh dengan teknologi saat ini. Rijndael dan AES menggunakan permutasi dan substitusi serta sejumlah *cipher* atau putaran berulang yang dimana setiap putaran menggunakan kunci internal unik [14]. Proses enkripsi dengan algoritma AES dimulai dari saat *admin* memberikan akses ke *user* yang dimana dikirim barengan dengan info akses. Sehingga enkripsi terjadi sebelum data akses tersebut dikirim ke *server*. Algoritma AES juga diterapkan pada *server* untuk melakukan dekripsi dari *ciphertext* yang dikirim dari *admin*. Tidak hanya proses dekrip, namun pada *server* juga terdapat proses enkripsi untuk mengirim kode akses ke pengguna untuk mendapatkan QR Code. Algoritma kriptografi AES dengan mode CBC memanfaatkan nilai *Initialization Vector* (IV) yang disesuaikan dengan ukuran masing-masing blok *plaintext* yang akan diproses sehingga membantu menjaga keamanan dan kerahasiaan data selama transmisi [15]. Proses enkripsi dengan mode CBC dapat dilihat pada Gambar 4.



Gambar 4 Proses Enkripsi AES pada mode CBC

Selain itu, terdapat proses dekripsi pada AES, diimplementasikan dengan arah yang berlawanan dengan proses enkripsi sehingga menghasilkan *inverse cipher*. Transfromasi yang dilakukan menjadi *inverse ShiftRows*, *inverse SubBytes*, *inverse MixColums*, dan *AddRoundKey*. Proses dekripsi dalam mode CBC dapat dilihat pada Gambar 5.

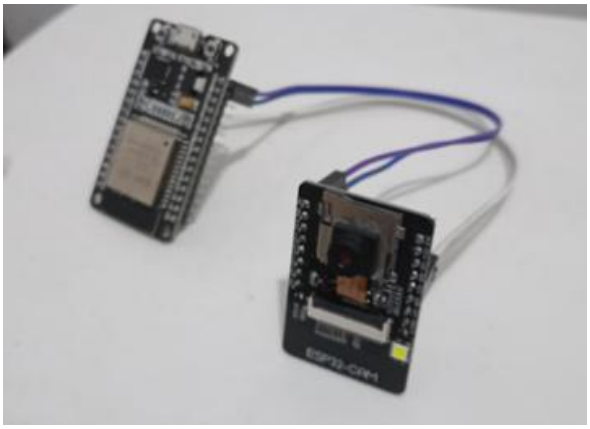


Gambar 5 Proses Dekripsi AES pada Mode CBC

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Perangkat Keras

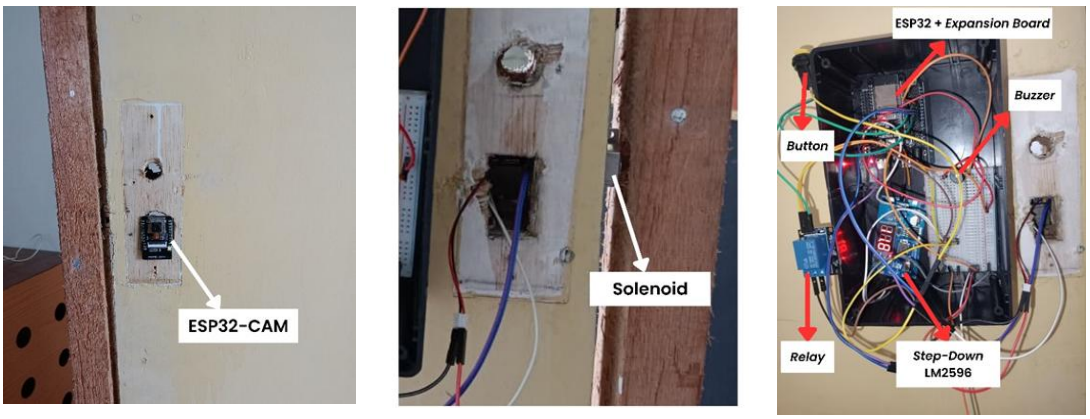
Pada tahap implementasi perangkat keras ini, komponen-komponen perangkat keras dihubungkan seperti ESP32 dan ESP32-CAM. Perangkat ESP32-CAM berfungsi sebagai pemindaian kode akses pada QR Code yang dimana akan dikirim ke NodeMCU ESP32. ESP32 sebagai kontrol utama pada sistem. Proses yang terjadi pada ESP32 merupakan proses enkripsi menggunakan algoritma AES pada kode akses yang diterima. Hasil implementasi sistem untuk membaca QR Code dapat ditunjukkan pada Gambar 6.



Gambar 6 Sistem Pembacaan QR Code

Selanjutnya terdapat implementasi sistem pembukaan solenoid apabila verifikasi berhasil. Implementasi ini bertujuan untuk memastikan bahwa perintah untuk membuka solenoid yang dimana jika sesuai, maka akan mengirim perintah ke *relay* untuk membuka kunci solenoid. Jika respons baliknya salah atau tidak sesuai yang dimana karena verifikasi gagal, maka akan menghidupkan *buzzer*.

Selain itu, adapun implementasi perangkat keras keseluruhan yang dimana NodeMCU ESP32 berfungsi sebagai perangkat utama untuk melakukan kontrol dalam sistem yang dibuat, seperti menerima perintah dari *button*, memberikan perintah ke *buzzer*, *relay*, dan melakukan komunikasi ke *server*. Namun, apabila ingin membuka pintu tanpa QR Code, dapat menekan tombol (*button*) yang berfungsi menghidupkan langsung solenoid untuk membuka pintu dari dalam ruangan. Hasil implementasi keseluruhan sistem untuk membaca QR Code dapat ditunjukkan pada Gambar 7.



(a) (b) (c)

Gambar 7 (a) Posisi ESP32-CAM pada Pintu, (b) Posisi Solenoid pada Pintu, (c) Keseluruhan Sistem pada Bagian Belakang Pintu

3.2 Implementasi Perangkat Lunak

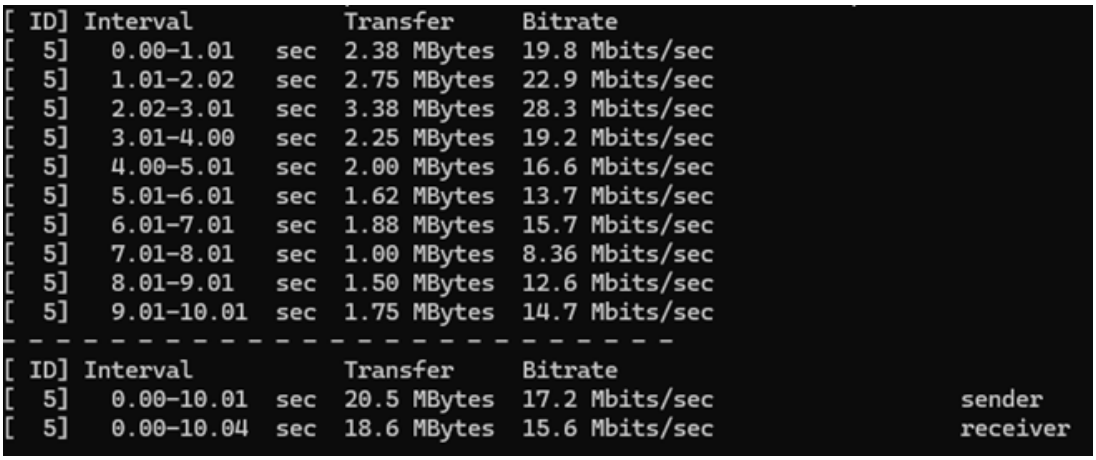
Implementasi perangkat lunak merupakan salah satu bagian terpenting dalam sistem, yaitu pengkodean sistem enkripsi dan dekripsi menggunakan algoritma *Advanced Encryption Standard* (AES) dengan mode CBC, komunikasi antar ESP32, dan komunikasi dengan *server*. Selain itu, ada juga perhitungan *delay* dan *throughput* yang terjadi pada *server* serta respons *time* pada keseluruhan sistem yang dibuat. Adapun pengembangan aplikasi berbasis *web app* untuk pemantauan dan pemberian akses ke pengguna yang dimana untuk sisi pengguna bergfungsi sebagai penerimaan sebuah kode QR untuk dilakukan pemindaian.

3. 2.1 Komunikasi ESP32 dengan Server

Komunikasi antara ESP32 dan *server* dilakukan menggunakan protokol HTTP POST melalui *library* HTTPClient. Proses dimulai saat ESP32 menerima data QR code dari ESP32-CAM melalui serial (Serial2), lalu membersihkan data menggunakan *trim*(). Data tersebut dienkripsi dengan algoritma AES-256-CBC menggunakan fungsi encryptAES256_CBC(), lalu dikodekan dalam format base64 dan dikirim ke *server* dalam bentuk JSON. Setelah pengiriman, sistem mencatat *delay* dan *throughput* sebagai indikator kinerja komunikasi. ESP32 kemudian mengirimkan ulang data ke *server*, mencakup *ciphertext*, nilai *delay*, dan *throughput* untuk dianalisis lebih lanjut di sisi *server*. Setelah sudah melakukan verifikasi di *server* maka akan dikirim respons balik dari *server* untuk pengambilan keputusan pada ESP32. Yang dimana nantinya ESP32 memberikan perintah ke *relay* jika berhasil dan menghidupkan *buzzer* jika gagal.

3.3 Pengujian Pengiriman Kode Akses

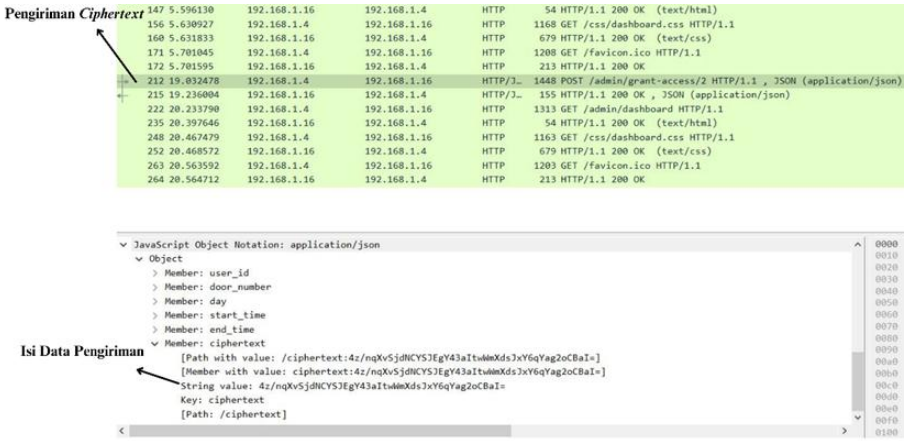
Pada sistem pintu cerdas, *admin* memberikan izin akses kepada *user* dalam bentuk data hari, jam, dan kode akses. Data ini dienkripsi di sisi *frontend* menjadi *ciphertext* sebelum dikirim ke *server*. Setelah diterima, *server* mendekripsinya menjadi *plaintext* untuk disimpan di *database* sebagai acuan verifikasi akses selanjutnya. Sebelum dilakukannya pengujian, perlu di cek terlebih dahulu berapa *bandwidth* yang digunakan selama proses pengujian berlangsung dengan menggunakan *tool* Iperf3. *Bandwidth* yang digunakan selama pengujian berlangsung dapat dilihat pada Gambar 8.



[ID]	Interval		Transfer	Bitrate	
[5]	0.00-1.01	sec	2.38 MBytes	19.8 Mbits/sec	
[5]	1.01-2.02	sec	2.75 MBytes	22.9 Mbits/sec	
[5]	2.02-3.01	sec	3.38 MBytes	28.3 Mbits/sec	
[5]	3.01-4.00	sec	2.25 MBytes	19.2 Mbits/sec	
[5]	4.00-5.01	sec	2.00 MBytes	16.6 Mbits/sec	
[5]	5.01-6.01	sec	1.62 MBytes	13.7 Mbits/sec	
[5]	6.01-7.01	sec	1.88 MBytes	15.7 Mbits/sec	
[5]	7.01-8.01	sec	1.00 MBytes	8.36 Mbits/sec	
[5]	8.01-9.01	sec	1.50 MBytes	12.6 Mbits/sec	
[5]	9.01-10.01	sec	1.75 MBytes	14.7 Mbits/sec	
[ID]	Interval		Transfer	Bitrate	
[5]	0.00-10.01	sec	20.5 MBytes	17.2 Mbits/sec	sender
[5]	0.00-10.04	sec	18.6 MBytes	15.6 Mbits/sec	receiver

Gambar 8 *Bandwidth* yang digunakan saat Pengujian

Dapat diketahui total transfer yang dari *client* sebesar 20,5 Mbps dari 10 data yang dikirim dengan rata-rata yang didapatkan sebesar 17,2 Mbps. Dengan total yang diterima oleh *server* sebesar 18,6 Mbps dimana rata-ratanya adalag 15,6 Mbps. Selanjutnya, dilakukan pengujian pengiriman kode akses. Pengujian ini dilakukan dengan melakukan *sniffing* menggunakan *tools wireshark* pada windows pada saat *admin* memberikan izin akses ke *user*. Hasil *sniffing* pada saat pengiriman kode akses dapat dilihat pada Gambar *9*.



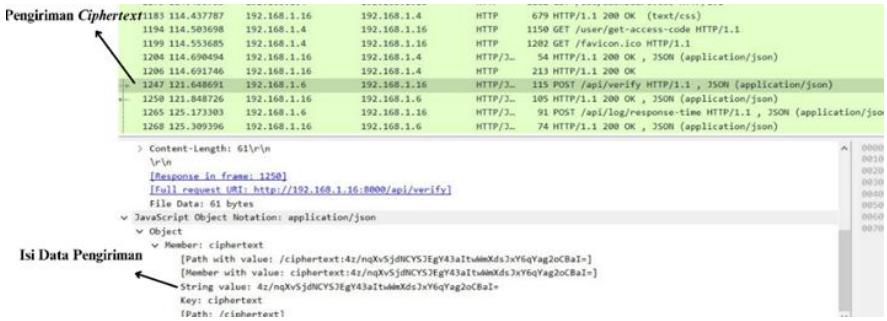
Gambar 9 Hasil *Sniffing* pada Pengiriman Kode Akses

Dapat dilihat bahwa data yang dikirim dari sisi *admin* ke *server* berupa *ciphertext* dari kode aksesnya. Sehingga ini membuktikan bahwa data yang di kirim ke *server* sudah dienkripsi dan terlindungi dari serangan *sniffing* yang hanya ingin mengintip kode akses untuk membuka pintu. Selanjutnya, pengujian dilakukan dengan pengguna membawa atau menunjukkan QR Code ke arah modul kamera ESP32-CAM sehingga modul dapat menangkap gambar dan melakukan proses *decoding*. Proses pemindaian QR pada ESP32-CAM dapat dilihat pada Gambar 10.

(a) (b)

Gambar 10 (a) Mendapatkan QR Code, (b) Melakukan *Scan* pada ESP32-CAM

Setelah *server* menerima kode akses yang berbentuk *ciphertext, server* akan mendekripsi ulang untuk mengambil *plaintext* dari kode akses tersebut untuk melakukan verifikasi. Percobaan *sniffing* dalam pengiriman kode akses dari NodeMCU ESP32 ke *server* dapat dilihat pada Gambar 11.



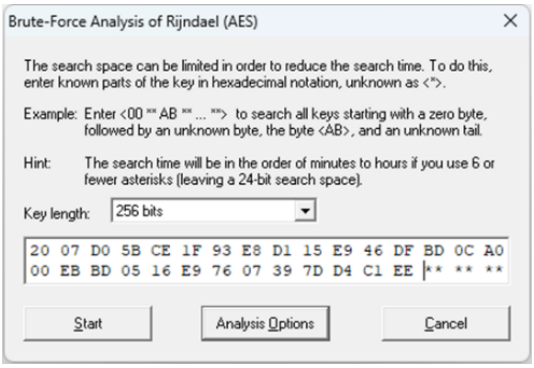
Gambar 11 Pengiriman Kode Akses ke *Server* untuk Verifikasi

Hal ini menunjukkan bahwa informasi kode akses yang dikirimkan tidak dapat langsung dibaca atau dimanfaatkan secara langsung oleh pihak yang tidak berwenang. Karena berbentuk *ciphertext*, pihak yang mencoba melakukan penyadapan harus terlebih dahulu melakukan proses dekripsi untuk memperoleh *plaintext* atau data asli dari kode akses tersebut. Adapun tabel hasil pengujian penyerangan saat pengiriman kode akses terdapat pada Tabel 1.

Tabel 1. Pengujian Serangan *Sniffing* pada Pengiriman Kode Akses

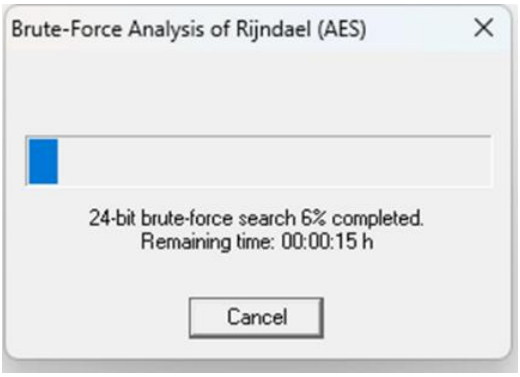
Id	*Plaintext*	*Ciphertext*
104	ihsanazhar-2_01Kam07:45-17:45	4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=
105	ihsanazhar-2_01Kam07:45-17:45	4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=
106	ihsanazhar-2_01Kam07:45-17:45	4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=
107	ihsanazhar-2_01Kam07:45-17:45	4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=
108	ihsanazhar-2_01Kam07:45-17:45	4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=

Selanjutnya dilakukan pengujian terhadap *ciphertext* yang didapatkan menggunakan aplikasi tambahan, yaitu Cryptool 1. Pengujian dilakukan pada *key* yang dimana 3 *byte* terakhir dikosongkan untuk dilakukan *bruteforce* pada *byte* kosong tersebut. Pemberian *key* dengan 3 *byte* terakhir kosong dapat dilihat pada Gambar 12.



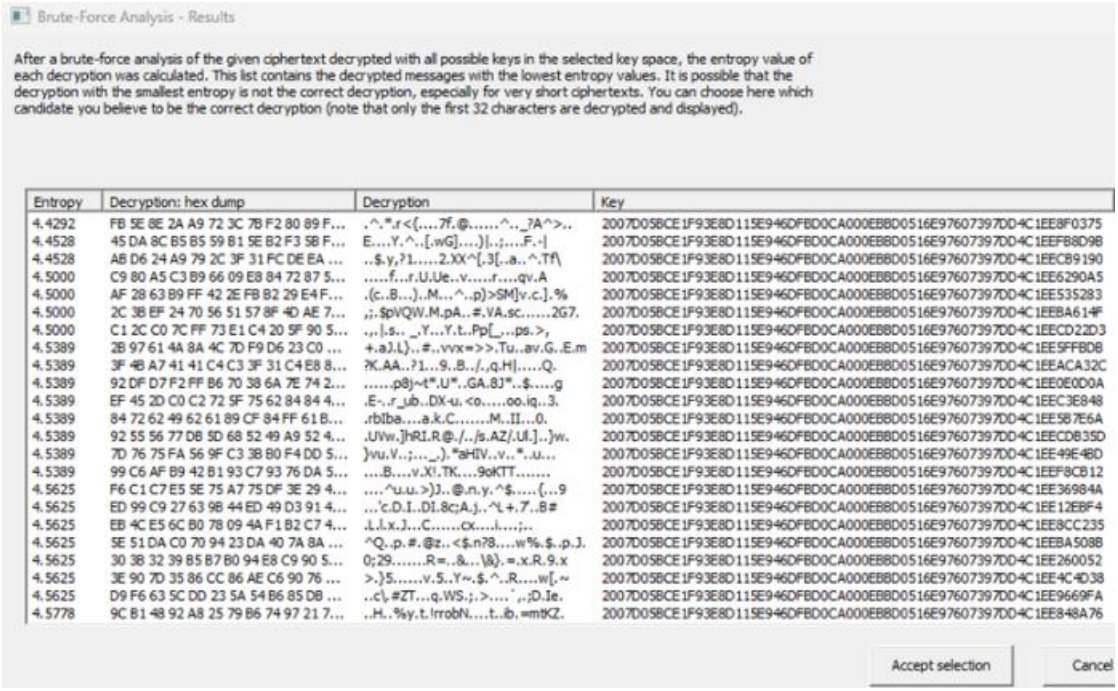
Gambar 12 Pemberian *Key* dengan 3 *Byte* Kosong diakhir

Dapat dilihat bahwa pemberian *key* tidak sepenuhnya lengkap atau diketahui *byte*-nya, yang dimana 3 *byte* terakhir dikosongkan dengan tanda "*" yang berarti tidak diketahui. Hal ini dilakukan untuk mengurangi waktu yang dibutuhkan dalam melakukan *bruteforce* pada *ciphertext* yang didapatkan. Sehingga waktu yang diperlukan dapat dilihat pada Gambar *13*.



Gambar 13 Estimasi Waktu *Bruteforce* pada 3 *byte* akhir di *key*

Waktu yang dibutuhkan singkat, yaitu kurang lebih 15 detik karena *byte* kosong yang dicari tidak banyak sehingga mempersingkat waktu yang dibutuhkan mencari semua kemungkinan yang ada. Setiap penambahan *byte* yang kosong akan memperpanjang waktu yang dibutuhkan dari menit hingga berjam-jam. Hasil beberapa percobaan *bruteforce* pada 3 *byte key* terakhir yang dikosongkan pada *ciphertext* hasil *sniffing* dapat dilihat pada Gambar *14*.



Gambar 14 Hasil Pengujian *Bruteforce* dengan Cryptool 1

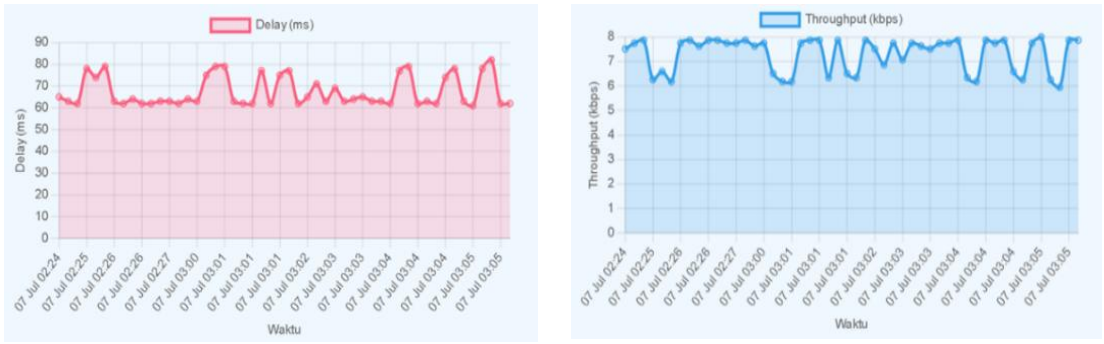
Dapat dilihat pada gambar bahwa banyak hasil yang gagal dalam 23 percobaan untuk mendapatkan *plaintext* asli dari *ciphertext* yang diretas. Sehingga upaya untuk memecahkan *ciphertext* tersebut menggunakan metode *bruteforce* pada beberapa *byte key* terbukti susah untuk berhasil. Adapun 5 data hasil percobaan *bruteforce* dengan Cryptool 1 pada Tabel 2.

Tabel 2. Hasil *Bruteforce* pada Sebuah *Ciphertext*

Ciphertext	***Plaintext***	**Waktu (detik)**
4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=	.^.*.r<....7f.@.....^..?A^>..	15
4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=	E....Y.^..[.wG]....)...;.....F.-|	15
4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=	..$.y,?1.....2.XX^[.3[..a..^.Tf	15
4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=	.....f...r.U.Ue..v.....r....qv.A	15
4z/nqXvSjdNCYSJEgY43aItwWmXdsJxY6qYag2oCBaI=	.(c..B...)..M...^..p)>SM]v.c.].%	15

3.4 Pengujian Delay dan Througput

Pengujian ini dilakukan untuk mengetahui bagaimana *delay* dan *throughput* yang terjadi pada *server* saat proses verifikasi berlangsung. Proses verifikasi tersebut melibatkan algoritma *Advanced Encryption System* (AES) saat melakukan dekripsi pada kode akses yang terekripsi. Setelah didapatkan *plaintext* dari kode akses, lalu di sesuaikan dengan yang ada di *database*. Hasil perhitungan *delay* dan *throughput* dapat dilihat pada Gambar 15.



(a) (b)

Gambar 15 (a) Hasil *Delay* pada Verifikasi di *Server,* (b) Hasil *Throughput* pada Verifikasi di *Server*

Grafik tersebut memperlihatkan bagaimana *delay* yang terjadi pada *server* saat melakukan verfikasi kode akses. Berdasarkan grafik tersebut, *delay* yang terjadi pada *server* cenderung stabil dengan rata-rata yang didapatkan adalah 67,46 ms. Hanya saja sempat terjadi lonjakan di beberapa bagian yang memiliki kisaran *delay* sebesar 74 ms hingga 82 ms. Sedangkan pada *throughput* yang terjadi cenderung stabil juga dengan rata-rata yang didapatkan sebesar 7,30 kbps. *Throughput* tertinggi yang didapatkan pada grafik adalah 8 kbps yang menandakan data yang berhasil dikirim di setiap waktunya cukup besar. Ini menandakan bahwa *server* dapat melayani banyak *request* dengan cepat dan bagus meskipun terdapat implementasi AES disistem. Tingkat stabil ini menandakan bahwa proses AES dapat optimal pada saat verifikasi di *server* terjadi. Adapun data dari grafik *delay* dan *throughput* dapat dilihat pada Tabel 3 dan Tabel 4.

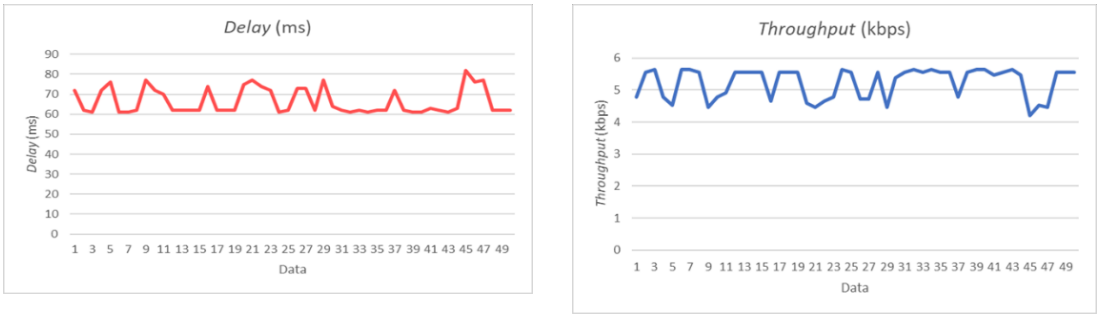
Tabel 3 Data *Delay* dalam Proses Verifikasi

Waktu	***User* Id**	**Aksi**	***Delay* (ms)**
2025-07-07 (16:58:41)	2	verify_qr	65
2025-07-07 (17:10:46)	2	verify_qr	80
2025-07-07 (17:11:52)	2	verify_qr	63
2025-07-07 (17:12:12)	2	verify_qr	83
2025-07-07 (17:12:54)	2	verify_qr	72

Tabel 4 Data *Throughput* dalam Proses Verifikasi

Waktu	***User* Id**	**Aksi**	***Throughput* (kbps)**
2025-07-07 (16:58:41)	2	verify_qr	7,51
2025-07-07 (17:10:46)	2	verify_qr	6,1
2025-07-07 (17:11:52)	2	verify_qr	7,75
2025-07-07 (17:12:12)	2	verify_qr	5,88
2025-07-07 (17:12:54)	2	verify_qr	6,78

Setelah itu, dilakukan juga pengujian proses verifikasi pada *server* yang dimana tanpa ada implementasi algoritma AES didalamnya. Sehingga kode akses yang dikirim ke *server* berupa teks asli atau *plaintext* yang bukan merupakan sebuah *ciphertext.* Maka dari itu, setelah diterima oleh *server*, *server* akan langsung melakukan pencocokan dengan yang ada di *database* tanpa melakukan proses dekripsi lagi terlebih dahulu. Grafik hasil perhitungan *delay* dan *throughput* pada *server* yang dimana tanpa ada implementasi algoritma AES dapat dilihat pada Gambar 16.



(a) (b)

Gambar 16 (a) Grafik *Delay* tanpa AES, (b) Grafik *Throughput* tanpa AES

Hasil pengujian menunjukkan adanya penurunan rata-rata *delay* dari 67,46 ms menjadi 66,4 ms setelah enkripsi dihilangkan. Dari sisi *throughput*, tanpa AES rata-rata hanya 5,23 kbps karena ukuran data lebih pendek dengan tanpa adanya enkripsi. Ini membuktikan bahwa proses verifikasi pada *server* dengan tanpa adanya implementasi algoritma AES, akan lebih cepat dibandingkan jika menggunakan AES. Data yang didapatkan dari perhitungan *delay* dan *throughput* pada *server* tersebut dapat dilihat pada Tabel 5 dan Tabel 6.

Tabel 5 Data *Delay* tanpa Implementasi AES

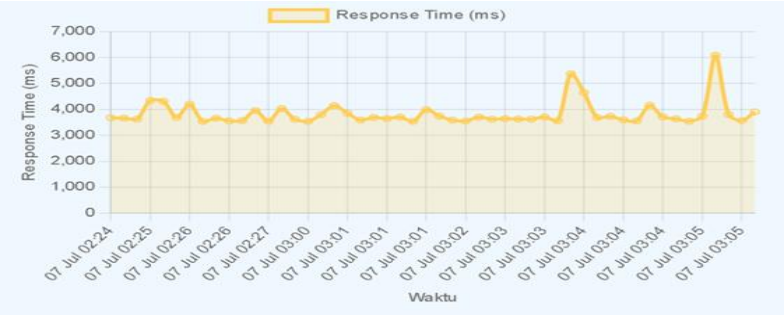
Waktu	*User* Id	Aksi	*Delay* (ms)
2025-08-25 (23:39:39)	2	verify_qr	76
2025-08-25 (23:39:43)	2	verify_qr	77
2025-08-25 (23:39:56)	2	verify_qr	62
2025-08-25 (23:40:11)	2	verify_qr	62
2025-08-25 (23:40:19)	2	verify_qr	62

Tabel 6 Data *Throughput* tanpa Implementasi AES

Waktu	*User* Id	Aksi	*Throughput* (kbps)
2025-08-25 (23:39:39)	2	verify_qr	4,53
2025-08-25 (23:39:43)	2	verify_qr	4,47
2025-08-25 (23:39:56)	2	verify_qr	5,55
2025-08-25 (23:40:11)	2	verify_qr	5,55
2025-08-25 (23:40:19)	2	verify_qr	5,55

3.5 Pengujian Respons Time

Pengujian ini bertujuan mengukur respons *time* sistem, dimulai saat ESP32 menerima data QR dari ESP32-CAM. Data tersebut dienkripsi dengan AES-256-CBC, lalu dikirim ke *server* untuk diverifikasi. *Server* kemudian mengirimkan respons terenkripsi (“VALID” atau “INVALID”) yang didekripsi kembali oleh ESP32. Jika valid, ESP32 mengaktifkan *relay* untuk membuka solenoid atau *buzzer*. Waktu dari penerimaan QR hingga solenoid terbuka digunakan untuk menghitung respons *time*. Grafik perhitungan respons *time* dapat dilihat pada Gambar *17*.



Gambar 17 Grafik Respons *Time*

Grafik tersebut menunjukkan bahwa respons *time* cenderung stabil juga dengan rata-rata yang didapatkan sebesar 3830,76 ms. Sempat terjadi lonjakan respons *time* pada data ke-47 pada grafik sebesar 6086 ms yang dimana tertinggi dalam 50 data terakhir tersebut. Adapun 5 data terakhir dari grafik respons *time* dapat dilihat pada Tabel 7.

Tabel 7 Data Respons *Time*

Algoritma	Waktu Proses	Ketelitian	Memori
2025-08-25 (23:39:39)	2	verify_qr	76
2025-08-25 (23:39:43)	2	verify_qr	77
2025-08-25 (23:39:56)	2	verify_qr	62
2025-08-25 (23:40:11)	2	verify_qr	62
2025-08-25 (23:40:19)	2	verify_qr	62

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan, dapat disimpulkan dari penelitian bahwa kode akses yang dikirimkan antar perangkat dan *server* tidak lagi dalam bentuk *plaintext*, melainkan *ciphertext*. Dengan hasil uji ketahanan sebanyak 23 kali menunjukkan bahwa upaya untuk meretas *ciphertext* yang membutuhkan waktu 15 detik tetap gagal mengembalikan *plaintext* asli. Sehingga membuat sistem lebih tahan dari serangan *Man-in-the-Middle* (MitM) seperti *sniffing*. Hal ini menunjukkan bahwa implementasi algoritma AES-256-CBC pada pengiriman kode akses QR Code berpengaruh dalam meningkatkan keamanan pengiriman data pada sistem pintu cerdas.

Penerapan algoritma AES pada kode akses turut memberikan dampak pada performa sistem, khususnya pada aspek *delay* dan *throughput*. Hasil pengujian menunjukkan adanya peningkatan rata-rata *delay* dari 66,4 ms menjadi 67,46 ms setelah enkripsi diterapkan. Dari sisi *throughput*, tanpa AES rata-rata hanya 5,23 kbps karena ukuran data lebih pendek, sedangkan dengan AES *throughput* meningkat menjadi 7,30 kbps dengan puncak mencapai 8 kbps akibat panjang *ciphertext* yang lebih besar. Meskipun terdapat kenaikan *delay* dan *throughput*, performa sistem tetap memadai untuk kebutuhan *real-time*. Hal ini membuktikan bahwa penambahan algoritma AES memang menambah waktu proses verifikasi di *server*, tetapi peningkatannya relatif kecil dan masih sesuai dengan standar TIPHON.

Selain itu, hasil pengujian respons *time* menunjukkan bahwa sistem mampu memproses permintaan akses mulai dari ESP32-CAM memperoleh kode akses hingga pembukaan solenoid dengan waktu yang relatif cepat dan stabil. Dari 50 kali pengujian, diperoleh rata-rata respons *time* sebesar 3830,76 ms dengan nilai tertinggi mencapai 6086 ms. Variasi respons *time* ini dipengaruhi oleh kondisi jaringan internet serta potensi *overheat* pada perangkat saat melakukan komunikasi antara ESP32 dengan *server*. Namun, lonjakan *delay* yang terjadi tidak berpengaruh signifikan terhadap performa keseluruhan, sehingga sistem tetap dapat berjalan dengan baik sesuai kebutuhan fungsionalnya.

5. SARAN

Namun, masih terdapat kelemahan dalam penelitian yang dilakukan, yaitu masih menggunakan protokol HTTP dibandingkan HTTPS yang dapat menambahkan lapisan keamanan dan QR Code yang tidak dinamis. Sehingga hal ini dapat menjadi celah dalam kemanan sistem yang dibuat. Sehingga dapat diubah menjadi menggunakan protokol HTTPS dan QR Code yang di *generate* secara random setiap beberapa waktu.

DAFTAR PUSTAKA

[1] M. R. Abdahu, U. Ristian, and H. Hasfani, “Implementasi Smart Helmet Cabinet pada Penyimpanan Helm Berbasis Mobile QR Code,” *J. Inform. J. Pengemb. IT*, vol. 9, no. 1,

pp. 78–85, 2024, https://doi.org/10.30591/jpit.v9i1.6593.
[2] A. Salam and S. B. Bhaskoro, "Sistem Keamanan Cerdas pada Kunci Pintu Otomatis menggunakan Kode QR," *Cybernatics*, vol. 5, no. 01, pp. 1–11, 2021, https://doi.org/10.29406/cbn.v5i01.2307.
[3] A. P. Muthoharum, B. Santoso, and L. Sunardi, "RANCANG BANGUN SMART SECURITY SYSTEM UNTUK KEAMANAN RUANG SERVER BERBASIS QR CODE PADA RUANG SERVER UNIVERSITAS BINA INSAN," *ESCAF*, pp. 1228–1235, 2023.
[4] M. P. HAKIM, U. RISTIAN, and S. SUHARDI, "Sistem Pintu Cerdas dengan QR Code berbasis Internet of Things sebagai Penerapan Edge Computing," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 11, no. 4, p. 907, 2023, https://doi.org/10.26760/elkomika.v11i4.907.
[5] F. Ferdiansyah, "Penggunaan Qr Code Berbasis Kriptografi Advanced Encryption Standard (AES) untuk Administrasi Rekam Medis," *J. Syntax Admiration*, vol. 2, no. 10, pp. 1870–1884, 2021, https://doi.org/10.46799/jsa.v2i10.325.
[6] F. Bibiola, T. U. Kalsum, and H. Alamsyah, "Penerapan Algoritma Advance Encryption Standard (AES) Untuk Pengamanan File Pada Aplikasi Berbasis WEB," *J. Surya Energy*, vol. 8, no. 1, pp. 35–51, 2023, https://doi.org/10.32502/jse.v8i1.6461.
[7] W. Adhiwibowo, A. M. Hirzan, and M. S. Suprayogi, "Peningkatan Keamanan Data End-To-End Smart Door Menggunakan Advanced Encryption Standard," *J. ELTIKOM*, vol. 6, no. 2, pp. 186–194, 2022, https://doi.org/10.31961/eltikom.v6i2.574.
[8] A. H. Nasrullah, A. M. Fajar, M. A. Taufiq, N. Rahmat, and F. Adiba, "Evaluation Of Fuzzy C-Means Method For District Clustering," *Media Comput. Sci.*, vol. 1, no. 2, pp. 117–128, 2024, https://doi.org/10.69616/mcs.v1i2.203.
[9] E. Suwanty, R. Hidayati, and K. Sari, "Implementation of Door Lock System Using Keyword and RFID," *J. Media Inf. Teknol.*, vol. 2, no. 2, pp. 53–64, 2025, https://doi.org/10.69616/mit.v2i2.240.
[10] M. D. Asrofa, S. Bahri, and K. Kasliono, "One-Time Pad Cryptography for Secure Data Transmission in IoT Smart Door Using QR Code," *J. Media Inf. Teknol.*, vol. 2, no. 2, pp. 133–148, 2025, https://doi.org/10.69616/mit.v2i2.248.
[11] M. R. Nur, A. Ranggareksa, F. Adiba, and A. Husna, "Clustering The Use Of Puskesmas Soreang Medicine Using The Fuzzy C-Means Method," *J. Heal. Innov. Technol.*, vol. 1, no. 1, pp. 1–8, 2025, https://doi.org/10.69616/johati.v1i1.226.
[12] I. Suharjo, "Prototype alat kendali otomatis penjemur pakaian menggunakan NodeMCU ESP32 dan Telegram Bot berbasis Internet of Things (IoT)," *J. Inf. Syst. Artif. Intell.*, vol. 1, no. 1, pp. 17–24, 2020.
[13] J. Handoyo and Y. M. Subakti, "Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (Aes)," *J. SITECH Sist. Inf. dan Teknol*, vol. 3, no. 2, pp. 143–152, 2020.
[14] D. Hulu, B. Nadeak, and S. Aripin, "Implementasi Algoritma AES (Advanced Encryption Standard) Untuk Keamanan File Hasil Radiologi di RSU Imelda Medan," *KOMIK (Konferensi*, vol. 4, pp. 78–86, 2020.
[15] A. Nugrahantoro, A. Fadlil, and I. Riadi, "Optimasi Keamanan Informasi Menggunakan Algoritma Advanced Encryption Standard (AES) Mode Chiper Block Chaining (CBC)," *J. Ilm. FIFO*, vol. 12, no. 1, p. 12, 2020, https://doi.org/10.22441/fifo.2020.v12i1.002.