Analisis Performansi Algoritma Small AES Menggunakan Arduino UNO, Studi Kasus : Pemantauan Suhu

1st Muhammad Naufal Rabbani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
muhnaufalr@student.telkomuniversity.ac.id

2nd Farah Afianti
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
farahafi@telkomuniversity.ac.id

***Abstrak* — Keamanan data dalam konteks Internet of Things (IoT) sangat penting, terutama dalam aplikasi pemantauan suhu. Algoritma kriptografi seperti Advanced Encryption Standard (AES) sering digunakan, namun algoritma yang lebih ringan seperti Small AES perlu dipertimbangkan. Penelitian ini bertujuan menerapkan dan mengevaluasi performa Small AES pada platform Arduino dengan studi kasus pemantauan suhu. Evaluasi dilakukan dengan membandingkan Small AES dengan algoritma SPECK berdasarkan kecepatan enkripsi dan dekripsi, penggunaan memori, dan Bit Avalanche Test. Hasil penelitian menunjukkan bahwa Small AES tidak lebih baik dari SPECK dalam hal kecepatan enkripsi dan dekripsi, tetapi Small AES unggul dalam penggunaan memori yang lebih sedikit dibandingkan SPECK. Dari hasil Bit Avalanche Test, Small AES lebih baik untuk data berukuran besar sedangkan SPECK lebih baik untuk data berukuran kecil.**

***Kata kunci*— AES, Bit Avalanche Test, IoT, Pemantauan Suhu, Small AES, SPECK**

I. PENDAHULUAN

Keamanan data sangat penting dalam dunia teknologi informasi yang terus berkembang. Salah satu cara untuk melindungi data sensitif dari akses yang tidak sah adalah dengan enkripsi data. Advanced Encryption Standard (AES) adalah algoritma enkripsi yang populer dan aman yang digunakan di seluruh dunia [1]. Algoritma AES telah digunakan dalam berbagai aplikasi, seperti komunikasi nirkabel, perbankan online, dan banyak lagi [2], [3]. Namun, kinerja enkripsi AES juga menjadi perhatian penting dalam beberapa aplikasi, terutama di Internet of Things (IoT). Perangkat IoT memiliki sumber daya yang terbatas dalam hal daya dan komputasi. Oleh karena itu, diperlukan versi AES yang lebih ringan, seperti Small AES, yang dapat berjalan pada perangkat IoT dengan sumber daya terbatas.

Pemantauan suhu adalah salah satu aplikasi yang sering ditemukan dalam perangkat IoT. Pemantauan suhu dapat digunakan dalam berbagai industri, seperti pertanian, penyimpanan makanan, dan rumah pintar [4], [5]. Data suhu dapat diserang untuk dimanipulasi datanya menggunakan jenis serangan Data Manipulation [6], [7]. Implementasi Small AES pada perangkat IoT untuk pemantauan suhu dapat menjadi solusi yang efektif untuk mengamankan data suhu. Penelitian ini membahas implementasi Small AES pada perangkat IoT, khususnya menggunakan platform Arduino, yang dikenal dengan sumber daya terbatasnya.Performa Small AES pada perangkat ini akan dianalisis dengan mempertimbangkan kecepatan enkripsi dan memori yang digunakan. Penelitian ini juga akan mencakup studi kasus pemantauan suhu sebagai contoh aplikasi praktis.

Penelitian akhir ini berfokus pada evaluasi dan penerapan Small AES dalam konteks Internet of Things (IoT), khususnya pada platform Arduino yang memiliki keterbatasan sumber daya. Permasalahan utama yang akan diteliti adalah bagaimana mengukur dan menganalisis performansi Small AES pada perangkat arduino. Dua pertanyaan kunci yang akan dijawab dalam penelitian ini adalah pertama, bagaimana cara mengukur dan menganalisis performansi Small AES pada Arduino UNO, terutama dalam hal kecepatan enkripsi dan dekripsi, serta penggunaan memori. Kedua, bagaimana performansi Small AES jika dibandingkan dengan algoritma enkripsi ringan lainnya. Penelitian ini akan mengimplementasikan algoritma enkripsi Small AES serta setidaknya satu algoritma kriptografi ringan lainnya. Parameter yang akan digunakan untuk mengevaluasi performa algoritma-algoritma ini adalah kecepatan enkripsi dan dekripsi, konsumsi memori di perangkat Arduino UNO serta analisis perbandingan plaintext dengan ciphertext menggunakan metode *Bit Avalanche Test* [8].

Penelitian ini bertujuan untuk mengimplementasikan Small AES pada perangkat Arduino UNO, untuk menganalisis performansinya dengan fokus pada faktor-faktor seperti kecepatan enkripsi dan dekripsi, penggunaan memori perangkat, serta perbandingan plaintext dan ciphertext yang dihasilkan dengan menggunakan metode *Bit Avalanche Test*. Selain itu, penelitian ini juga bertujuan untuk melakukan perbandingan performansi Small AES dengan algoritma SPECK [9]. Algoritma SPECK dipilih sebagai pembanding karena algoritma ini menggunakan fungsi sederhana di setiap putarannya, berbanding terbalik dengan Small AES yang di setiap putarannya menggunakan berbagai fungsi yang kompleks [9]. Tujuan dari perbandingan ini adalah untuk mengidentifikasi kelebihan dan kelemahan masing-masing algoritma, sehingga dapat memberikan wawasan yang lebih baik dalam pemilihan algoritma enkripsi yang tepat untuk pengaplikasiannya pada perangkat IoT.

Paper ini akan disusun dalam 4 bab setelah pendahuluan, yaitu studi terkait, rancangan dan implementasi program, hasil pengujian, dan kesimpulan. Bagian studi terkait menjelaskan berbagai penelitian atau studi yang berkaitan dengan penelitian ini. Rancangan dan implementasi program memberikan gambaran mengenai bagaimana

algoritma Small AES bekerja dan bagaimana alur implementasi Small AES ke Arduino UNO yang menerapkan sensor pemantauan suhu. Hasil pengujian menampilkan dan menjelaskan hasil analisis performansi penerapan algoritma Small AES, kemudian membandingkannya dengan algoritma SPECK. Adapun bagian kesimpulan akan memuat kesimpulan dari hasil pengujian dan analisis hasil pengujian serta saran untuk penelitian lebih lanjut.

II. PENELITIAN TERKAIT

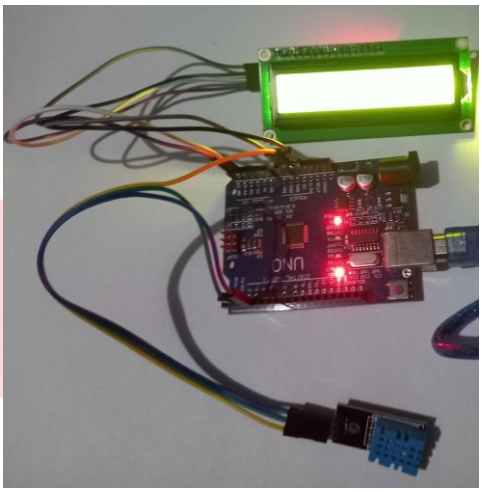
Sudah ada beberapa penelitian telah mengilustrasikan sejumlah pendekatan dalam mengoptimalkan dan meningkatkan performansi Advanced Encryption Standard (AES) dalam konteks perangkat terbatas, khususnya Arduino dan perangkat IoT. Dalam penelitian oleh Mohammad dan Abdullah [10], dilakukan penelitian untuk meningkatkan performa AES dengan mengurangi konsumsi daya algoritma dan meningkatkan kinerja kriptografi pada perangkat terbatas dengan sumber daya terbatas. Penelitian lain oleh Sukiatmodjo [11] fokus pada perbandingan antara Data Encryption Standard (DES) dan Advanced Encryption Standard (AES) dalam hal kecepatan eksekusi dan konsumsi daya pada perangkat Arduino.

Selain itu, dalam penelitian oleh Dutta, dkk [12] penekanan diberikan pada perbandingan antara solusi perangkat keras dan perangkat lunak, modifikasi strategi Mix-Column/S-box, serta berbagai serangan yang dapat mempengaruhi keamanan IoT. Sementara itu, dalam tinjauan oleh Hamza dan Kumar [13], penelitian lebih umum membahas algoritma kriptografi seperti DES, AES, dan RSA, baik dalam konteks kriptografi simetris maupun asimetris. Adapun dalam penelitian oleh Fotovvat, dkk [14], dilakukan perbandingan kinerja dari 32 algoritma kriptografi ringan dengan menerapkannya pada perangkat Raspberry Pi 3, Raspberry Pi Zero W, dan IMX233.

Dalam penelitian lainnya, Fadhil, dkk[15] mengimplementasikan algoritma Lightweight AES pada perangkat Raspberry Pi 3 dengan menggunakan sensor suhu dan sensor pendeteksi api. Perbedaan penelitian tersebut dengan penelitian ini adalah pada penelitian ini hanya digunakan sensor suhu dan pengimplementasiannya menggunakan perangkat Arduino UNO yang kemudian dibandingkan performansinya dengan algoritma SPECK, sedangkan pada penelitian tersebut menggunakan perangkat Raspberry Pi 3 dan lebih berfokus pada pengimplementasian Lightweight AES saja. Sementara itu, El Sobky, dkk[16] melakukan penelitian untuk meningkatkan keamanan dan kompleksitas Substitution Box (S-box) dalam algoritma Advanced Encryption Standard (AES) dengan menggunakan pendekatan yang berbeda dari metode konvensional. Adam, dkk [17]dalam penelitiannya mencoba untuk mengombinasikan algoritma Mini-AES dengan algoritma Geoencryption untuk memberikan keamanan yang lebih tinggi. Adapun penelitian oleh Singha, dkk [18] melakukan tinjauan komprehensif dan kronologis terhadap perkembangan teknik desain Substitution-box (S-box) dalam Advanced Encryption Standard (AES) serta mengevaluasi berbagai countermeasures yang telah dikembangkan untuk melindungi AES dari serangan side-channel, khususnya Power Analysis Attacks (PAAs).

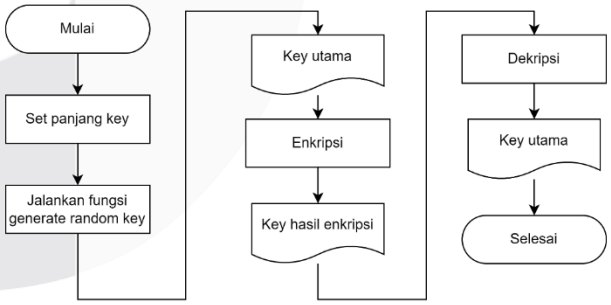
Beberapa penelitian terkait ini membentuk landasan kuat bagi pengembangan analisis performansi Small AES di Arduino. Oleh karena itu, penelitian ini bertujuan untuk mendalami performansi Small AES pada perangkat Arduino dalam konteks IoT, lalu membandingkannya dengan setidaknya satu algoritma kriptografi ringan lainnya.

III. RANCANGAN DAN IMPLEMENTASI PROGRAM



GAMBAR 1.
Setup Penelitian

Pada *GAMBAR 1* ditampilkan setup dari penelitian ini yang menggunakan perangkat Arduino UNO, Sensor DHT 11, dan LCD 16x2. Sistem yang dibangun pada setup tersebut memiliki dua skenario. Skenario pertama adalah skenario dimana arduino melakukan proses enkripsi, setelah itu data yang telah dienkripsi dikirim ke komputer untuk di dekripsi. Adapun skenario kedua adalah komputer mengirimkan sebuah teks kepada arduino yang telah dienkripsi, kemudian pada arduino dilakukan dekripsi untuk menampilkan teks pada LCD.

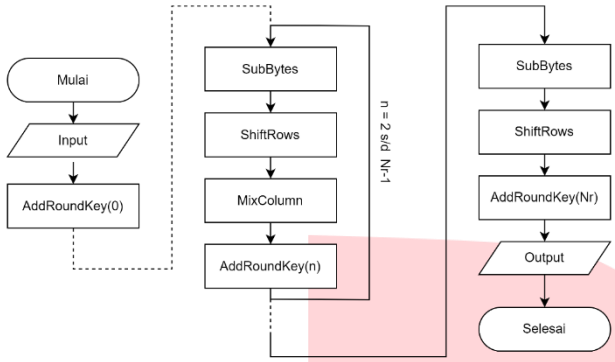


GAMBAR 2.
Inisialisasi Kunci

Sebelum melakukan proses enkripsi, diperlukan sebuah kunci yang nantinya akan digunakan sebagai kunci utama untuk mengakses pesan atau teks yang telah dienkripsi. Pada GAMBAR 2, proses inisialisasi kunci dimulai dengan menentukan panjang kunci yang akan digunakan. Karena pada penelitian ini akan menggunakan algoritma Small AES 256, maka panjang kunci yang akan digunakan adalah 32 byte. Setelah menentukan panjang kunci yang akan digunakan, selanjutnya akan dijalankan sebuah fungsi untuk menghasilkan sebuah kunci acak. Kunci acak adalah kunci

dinamis dalam bentuk byte yang akan dihasilkan secara acak dan terjadi di setiap iterasi program. Alasan peneliti menggunakan kunci yang dinamis adalah untuk menambah keamanan data. Dari fungsi menghasilkan kunci acak tersebut didapatkan sebuah kunci utama yang akan digunakan untuk proses enkripsi dan dekripsi.

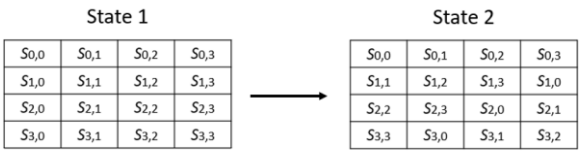
A. Proses Enkripsi



GAMBAR 3.
Alur Enkripsi Small AES

Pada dijelaskan proses enkripsi pada Small AES melibatkan serangkaian operasi yang diulang sejumlah putaran tertentu, di mana setiap putaran terdiri dari tiga operasi utama, yaitu AddRoundKey, SubBytes, ShiftRows, dan MixColumn [19]. Perbedaan utama antara AES Small dengan AES pada umumnya adalah proses ekspansi kunci pada Small AES dilakukan secara bertahap pada setiap round nya tanpa menyimpan keseluruhan jadwal kunci ke dalam memori. Karena jadwal kunci tidak tersimpan di memori, hal ini dapat berdampak ke performa algoritma Small AES.

Proses dimulai dengan operasi AddRoundKey(0), di mana blok data di-XOR dengan subkunci pertama dari jadwal kunci enkripsi [19]. Subkunci ini diperoleh dari ekspansi kunci enkripsi yang melibatkan operasi bitwise XOR, substitusi, dan pergeseran byte. Selanjutnya, blok data melewati operasi SubBytes, di mana setiap byte diganti dengan nilai yang sesuai dari S-Box [20], [21]. Fungsi utama S-Box adalah menggantikan setiap byte dalam blok data dengan byte lainnya berdasarkan suatu substitusi yang telah ditentukan.



GAMBAR 4.
Operasi ShiftRows

Setelah operasi SubBytes, blok data mengalami operasi ShiftRows, di mana setiap baris dirotasi ke kiri sesuai dengan nomor barisnya. Pada *GAMBAR 4* dijelaskan bagaimana pada Small AES 256, blok data 4x4 dirotasi sehingga baris pertama tidak bergeser, baris kedua mengalami pergeseran satu posisi ke kiri, yang berarti byte terkiri dipindahkan ke posisi paling kanan, baris ketiga mengalami pergeseran dua posisi ke kiri, sehingga dua byte sebelah kiri ditukar dengan dua byte sebelah kanan, dan baris keempat mengalami pergeseran tiga posisi ke kiri, yang pada dasarnya memiliki efek yang sama dengan menggeser satu posisi ke kanan [19].

Langkah berikutnya adalah MixColumns, di mana setiap kolom blok data diubah menggunakan matriks MDS (Matrices Diffusion Layer) yang berfungsi untuk memberikan efek difusi. Difusi adalah proses di mana bit-bit dalam input memengaruhi banyak bit dalam output. Ini bertujuan agar perubahan kecil dalam plaintext atau kunci menghasilkan perubahan besar dan kompleks dalam ciphertext, membuat pola lebih sulit dikenali dan dieksploitasi oleh penyerang[22]. Bentuk umum dari matriks MDS adalah sebagai berikut :

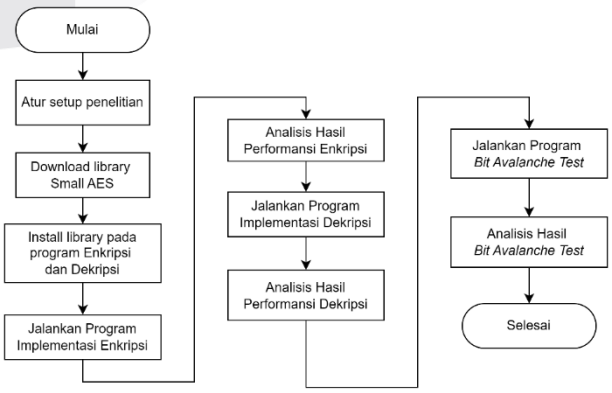
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \quad (1)$$

Matriks MDS adalah matriks invertibel dengan elemen dari bidang Galois GF(2^8). Setiap kolom diubah dengan matriks MDS, yang meningkatkan difusi dan kompleksitas enkripsi. GF sendiri adalah singkatan dari Galois Field atau Bidang Galois. Dalam konteks kriptografi dan matematika diskrit, Bidang Galois adalah struktur aljabar yang digunakan untuk operasi aritmetika yang memiliki sifat tertentu. Dalam kasus AES, kita sering berurusan dengan Galois Field yang memiliki karakteristik dua, disimbolkan sebagai GF(2^8) karena mempunyai 2^8 elemen [19].

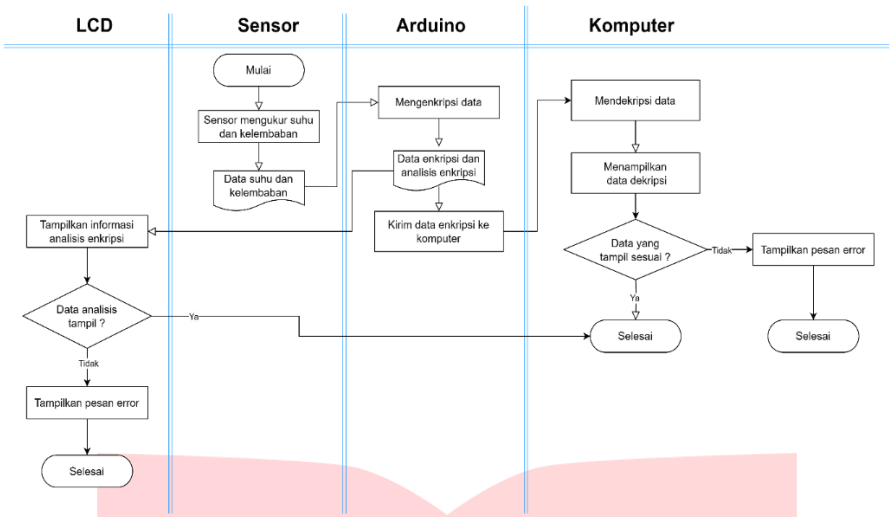
Selanjutnya, blok data melalui serangkaian operasi AddRoundKey(n) hingga AddRoundKey(Nr-1), di mana setiap kali blok data di-XOR dengan subkunci berikutnya dari jadwal kunci enkripsi. Pada putaran terakhir (Nr), operasi MixColumns diabaikan. Jumlah putaran (Nr) tergantung pada besar kunci yang digunakan. Pada penelitian ini, akan digunakan Small AES 256 dengan besar kunci 32 byte. Adapun jumlah putaran nya sebanyak 14 putaran, sama seperti AES pada umumnya.

B. Implementasi Small AES

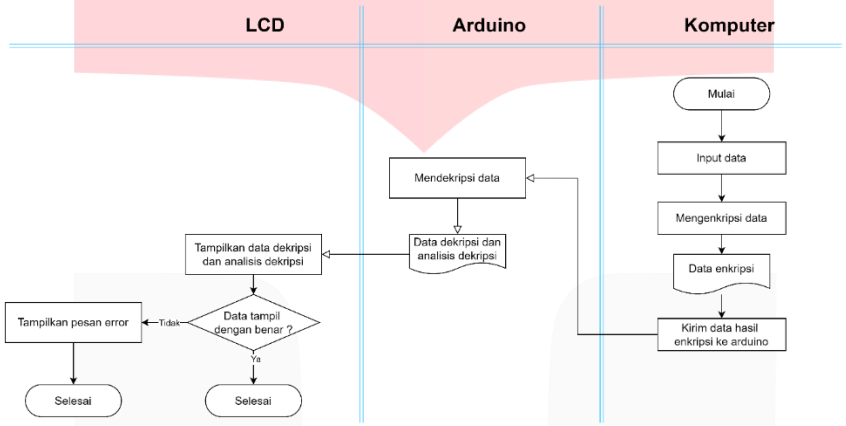
Untuk mengimplementasikan Small AES ke Arduino UNO, dibutuhkan sebuah library. Dalam implementasi penelitian ini, library kriptografi Arduino yang dikembangkan oleh Rhys Weatherly akan digunakan untuk mengimplementasikan kedua algoritma, yaitu Small AES dan SPECK. Library ini dipilih karena dapat menyediakan implementasi dan dokumentasi algoritma kriptografi pada perangkat Arduino yang cukup lengkap. Pada *GAMBAR 5* dijelaskan bagaimana alur implementasi Small AES pada Arduino UNO untuk penelitian ini.



GAMBAR 5.
Alur Implementasi Small AES



GAMBAR 6.
Alur Implementasi Enkripsi



GAMBAR 7.
Alur Implementasi Dekripsi

Sebelum menjalankan program implementasi enkripsi dan dekripsi, setup penelitian perlu diatur terlebih dahulu agar program bisa diupload dan dijalankan pada Arduino. Setelah itu, library perlu diunduh dan kemudian diinstall pada Arduino UNO sehingga bisa diimplementasikan pada program. Kemudian, program implementasi enkripsi dijalankan terlebih dahulu dan dianalisis hasil performansi enkripsinya. Setelah didapatkan data analisis yang dibutuhkan, program implementasi dekripsi dijalankan. Data performansi dekripsi kemudian dikumpulkan dan disimpan untuk dianalisis. Proses terakhir adalah menjalankan program *Bit Avalanche Test* untuk mengukur perbandingan perubahan setiap bit plaintext yang terjadi di ciphertext. Program tersebut akan menampilkan persentase skor perubahan yang terjadi. Dari data performansi dan skor *Bit Avalanche Test* inilah didapatkan informasi mengenai kelebihan dan kekurangan Small AES.

C. Implementasi Enkripsi

Skenario enkripsi adalah skenario dimana arduino akan melakukan proses enkripsi, setelah itu data yang telah dienkripsi akan dikirim ke komputer untuk di dekripsi. Pada **Error! Reference source not found.** dijelaskan bagaimana skenario pertama akan dijalankan pada penelitian ini. Awalnya sensor akan mulai mengukur suhu dan kelembaban ruangan. Data suhu dan kelembaban ini kemudian akan dienkripsi oleh arduino menggunakan algoritma Small AES varian 256 bit. Dari hasil enkripsi tersebut didapatkanlah data enkripsi dan data analisis enkripsi. Data enkripsi kemudian dikirim ke komputer untuk dilakukan proses dekripsi. Data yang sudah didekripsi tersebut akan dicek apakah datanya sesuai atau tidak. Jika tidak, maka akan tampil pesan error. Adapun data analisis performansi akan dikirimkan ke LCD untuk ditampilkan. Jika data performansi yang ditampilkan tidak sesuai, maka akan tampil pesan error.

D. Implementasi Dekripsi

Pada **Error! Reference source not found.** dijelaskan b agaimana skenario dekripsi dilakukan. Pertama, program di komputer mengenkripsi sebuah teks yang nantinya akan ditampilkan pada LCD oleh arduino. Pada program enkripsi ini plaintext akan diinputkan untuk dienkripsi lalu dikirim ke arduino melalui komunikasi serial. Setelah enkripsi dilakukan, arduino akan melakukan dekripsi data lalu menampilkannya ke laya LCD. Lalu program melakukan pengecekan apakah data yang ditampilkan sesuai atau tidak. Proses dekripsi berhasil jika layar LCD yang dipasang pada arduino dapat menampilkan plaintext yang diinputkan pada program enkripsi tadi. Selain itu, ada juga informasi analisis dekripsi yang tampil. Namun, jika data yang ditampilkan tidak sesuai dengan data yang diinputkan pada program, maka akan tampil sebuah pesan error.

IV. EVALUASI

A. Skenario Pengujian

Metode pengujian akan terbagi menjadi tiga. Pengujian pertama dilakukan untuk menguji algoritma Small AES menggunakan salah satu metode *Cryptanalysis*, yaitu *Known-plaintext attack* [23]. Metode *Known-plaintext attack* merupakan jenis serangan kriptanalisis di mana penyerang memiliki akses ke satu atau lebih pasangan teks asli dan teks yang telah dienkripsi [23]. Dengan informasi ini, penyerang berusaha untuk menentukan kunci kriptografi yang digunakan untuk enkripsi. Pengujian ini berguna untuk mengetahui apakah implementasi Small AES pada perangkat Arduino sudah benar dan aman. Pada pengujian ini, dibuat sebuah program untuk menebak kunci yang digunakan untuk mengenkripsi data suhu. Untuk mencapai hal tersebut, program akan melakukan *brute force* untuk mencoba semua kemungkinan kunci yang digunakan dengan maksimal percobaan seratus ribu kali perulangan, satu juta kali, dan seratus juta kali perulangan.

Pengujian kedua dilakukan dengan mengoperasikan kedua algoritma selama tiga puluh kali secara berulang, dengan memberikan jeda selama satu detik setiap kali algoritma dijalankan. Tujuan dari pengujian ini adalah untuk mengevaluasi perbedaan performansi antara kedua algoritma tersebut. Pengujian kedua akan membandingkan hasil enkripsi plaintext ke ciphertext menggunakan metode *Bit Avalanche Test*. *Bit Avalanche Test* adalah metode yang digunakan untuk membandingkan perubahan kecil yang terjadi pada setiap bit plaintext saat proses enkripsi, yang diharapkan menghasilkan output yang berbeda secara signifikan [8], [24]. Penelitian yang dilakukan oleh Wahid M, dkk [25] dan Echeverri [26]juga menunjukkan bahwa *avalanche effect* menunjukkan performa dari suatu algoritma kriptografi. Menurut Astuti N, dkk [27] dalam penelitiannya, algoritma kriptografi yang aman seharusnya memiliki presentase efek lavina sekitar 50%. Oleh karena itu, salah satu fokus penelitian ini adalah untuk menentukan apakah Small AES dan SPECK memenuhi kriteria ini.

Pengujian kedua dan ketiga akan terbagi lagi menjadi tiga skenario. Skenario pengujian pertama hanya menggunakan salah satu data dari suhu dan kelembaban, sehingga data memiliki panjang 4 byte. Skenario pengujian kedua menggunakan data suhu dan kelembaban, sehingga data memiliki panjang 8 byte. Adapun skenario ketiga juga menggunakan salah satu data saja, tetapi data ini disimpan kedalam sebuah array hingga empat kali. Jadi jika data yang disimpan dalam array belum mencapai empat, maka dilakukan perulangan lagi untuk menambah data. Sehingga pada skenario tiga, total panjang data mencapai 16 byte.

Dikarenakan terdapat tiga skenario pengujian, maka pada pengujian *Bit Avalanche Test* akan diperiksa panjang byte dari data yang di uji. Hal ini dikarenakan plaintext dipadding untuk panjang blok enkripsi, sehingga program perlu memeriksa seluruh ciphertext dan kemudian membandingkan setiap bit ciphertext dengan bit plaintext beserta paddingnya untuk skenario pertama dan kedua. Sehingga jika misalnya data plaintext hanya memiliki panjang 4 byte, maka program akan menambahkan padding ke plaintext kemudian membandingkannya dengan ciphertext melalui tes. Adapun rumus untuk menghitung skor Bit Avalanche Test adalah sebagai berikut:

$$Skor\ Avalanche\ Test = \frac{\substack{Jumlah\ bit\ yang\ berbeda \\ antara\ ciphertext \\ dan\ plaintext}}{\substack{Jumlah\ total\ bit \\ dalam\ ciphertext}} \quad (2)$$

B. Hasil dan Analisa Pengujian

1. *Known-plaintext Attack*

TABLE 1.
Hasil percobaan Known-plaintext Attack

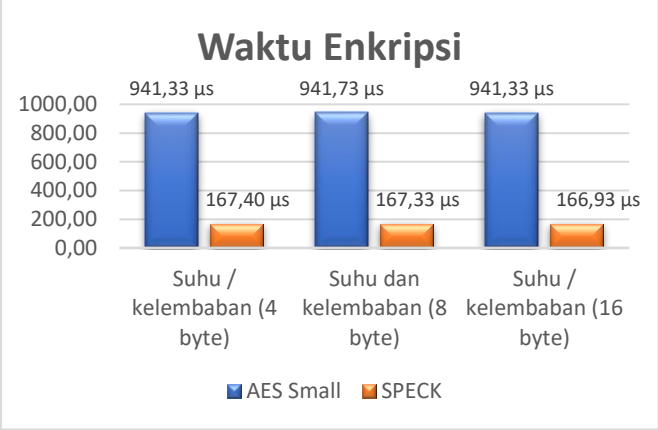
Percobaan	Waktu	Kunci Didapatkan ?
2^{16} (65536 x)	0.79 detik	Tidak
2^{20} (1048576 x)	13.20 detik	Tidak
2^{24} (16777216 x)	205.09 detik	Tidak

Pada table 1 ditampilkan bagaimana pada pengujian ini, implementasi Small AES sudah berhasil mengamankan data dari serangan *Known-plaintext attack*. Data tersebut juga menunjukkan bahwa metode *brute force* tidak dapat menembus keamanan algoritma Small AES. Panjang kunci Small AES yang diimplementasikan pada penelitian ini adalah 256 bit atau 32 byte, sehingga normalnya membutuhkan percobaan sebanyak 2^{256} kali untuk menemukan kunci utama menggunakan metode *brute force*. Namun karena keterbatasan kekuatan komputer, pengujian ini hanya dilakukan sebanyak tiga percobaan. Percobaan yang pertama sebanyak 2^{16} kali, kedua sebanyak 2^{20} kali, dan ketiga sebanyak 2^{24} kali.

Perlu diingat, pengujian ini dilakukan pada sebuah laptop dengan spesifikasi processor Intel Core i7-11800h dan RAM sebesar 16 GB. Sehingga waktu pengujian bisa saja berbeda di setiap komputer atau laptop yang digunakan.

2. Perbandingan Performansi

a. Waktu Enkripsi dan Dekripsi

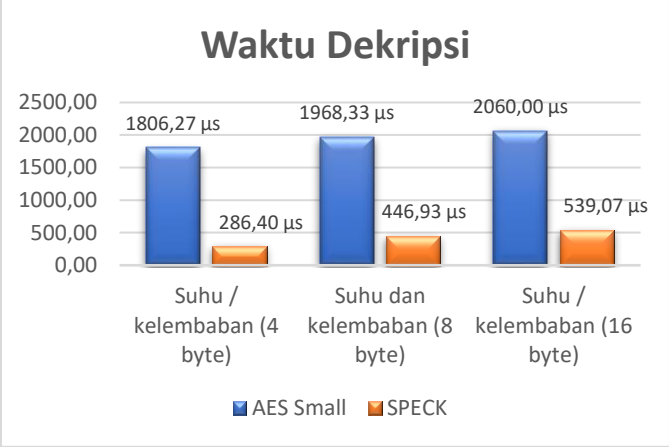


GAMBAR 8.
Statistik Waktu Enkripsi

Grafik batang pada *GAMBAR 8* menampilkan rata-rata waktu enkripsi yang dibutuhkan masing-masing algoritma kriptografi untuk mengenkripsi data suhu dan kelembaban. Dari ketiga skenario pengujian, Small AES menunjukkan hasil yang cukup tinggi dengan kisaran waktu 941.33 µs untuk panjang data 4 dan 16 byte, dan 941.73 µs untuk

panjang data 8 byte. Sedangkan algoritma SPECK hanya membutuhkan waktu 167.40 µs untuk panjang data 4 byte, 167.33 untuk panjang data 8 byte dan 166.93 µs untuk panjang data 16 byte. Hal ini menunjukkan Small AES membutuhkan lebih banyak waktu untuk melakukan enkripsi jika dibandingkan dengan SPECK.

Adapun dari segi dekripsi, SPECK kembali menunjukkan performa yang lebih baik jika dibandingkan dengan Small AES. Pada *GAMBAR 9* ditampilkan grafik rata-rata waktu dekripsi yang dibutuhkan setiap algoritma pada ketiga skenario yang dijalankan. Small AES kembali menunjukkan hasil yang cukup tinggi dengan kisaran waktu 1806.27 µs untuk panjang data 4, 1968.33 µs untuk panjang data 8 byte dan 2060 µs untuk panjang data 16 byte . Sedangkan SPECK hanya membutuhkan waktu 286.40 µs untuk panjang data 4 byte, 446.93 µs untuk panjang data 8 byte dan 539.07 µs untuk panjang data 16 byte.



GAMBAR 9.
Statistik Waktu Dekripsi

Pada ketiga skenario, SPECK menunjukkan keunggulannya baik dari segi enkripsi maupun dekripsi. Dari *GAMBAR 8*, didapatkan selisih waktu enkripsi antara Small AES dengan SPECK pada ketiga skenario mencapai kisaran 773.93 µs sampai 774.4 µs. Sedangkan pada *GAMBAR 9*, didapatkan selisih waktu dekripsi antara Small AES dengan SPECK pada skenario pertama mencapai 1519.87 µs, skenario kedua 1541.4 µs, dan skenario ketiga mencapai 1520.93 µs.

b. Penggunaan Memori

TABLE 2.
Statistik Penggunaan Memori

	Enkripsi		Dekripsi	
	Small AES	SPECK	Small AES	SPECK
Suhu / kelembaban (4 byte)	957 bit	1138 bit	886 bit	1036 bit
Suhu dan kelembaban (8 byte)	669 bit	850 bit	893 bit	1043 bit
Suhu / kelembaban (16 byte)	985 bit	1166 bit	897 bit	1047 bit

Performa kedua algoritma dari segi penggunaan memori dapat dilihat pada *TABLE 2*. Penggunaan memori kedua algoritma stabil di ketiga skenario dan tidak ada perubahan selama tiga puluh kali percobaan. Dari tabel tersebut didapatkan informasi bahwa Small AES lebih baik dibandingkan SPECK dari segi penggunaan memori. Untuk enkripsi, algoritma Small AES menggunakan memori sebanyak 957 bit untuk panjang data 4 byte, 669 untuk panjang data 8 byte, dan 985 untuk panjang data 16 byte. Sedangkan algoritma SPECK menggunakan memori sebanyak 1138 bit untuk panjang data 4 byte, 850 untuk panjang data 8 byte, dan 1166 untuk panjang data 16 byte. Dari data tersebut kita bisa mendapatkan selisih penggunaan memori untuk enkripsi kedua algoritma tersebut mencapai 181 bit untuk ketiga skenario.

Adapun untuk dekripsi, Small AES menggunakan memori sebanyak 886 bit untuk panjang data 4 byte, 893 untuk panjang data 8 byte, dan 897 untuk panjang data 16 byte. Sedangkan SPECK menggunakan memori sebanyak 1036 bit untuk panjang data 4 byte, 1043 untuk panjang data 8 byte, dan 1047 untuk panjang data 16 byte. Sehingga didapatkan selisih penggunaan memori untuk dekripsi kedua algoritma tersebut mencapai 150 bit untuk ketiga skenario.

3. *Bit Avalanche Test*

TABLE 3.
Statistik Bit Avalanche Test

	4 Byte		8 Byte		16 Byte	
	Small AES	SPECK	Small AES	SPECK	Small AES	SPECK
Rata-rata skor	49.06 %	49.69 %	49.25 %	50.47 %	49.97 %	48.26 %
Modus	46.88 %	53.12 %	48.44 %	53.12 %	47.66 %	47.66 %
Skor tertinggi	65.62 %	62.50 %	59.38 %	64.06 %	62.50 %	53.91 %
Skor terendah	31.25 %	31.25 %	35.94 %	40.62 %	41.41 %	39.06 %

Pada *TABLE 3* ditampilkan hasil pengujian *Bit Avalanche Test*, yang menunjukkan kedua algoritma memiliki rata-rata skor yang hampir sama di ketiga skenario pengujian. Namun jika diperhatikan, perbandingan rata-rata skor Small AES dengan skor ideal yang diusulkan oleh Astuti N, dkk [27] yaitu 50%, lebih baik dibandingkan SPECK. Small AES memiliki perbandingan sebesar 0.94% untuk data 4 byte, 0.75% untuk data 8 byte dan 0.03% untuk data 16 byte. Sedangkan SPECK memiliki perbandingan sebesar 0.31% untuk data 4 byte, -0.47% untuk data 8 byte, dan 1.75% untuk data 16 byte. Dari perbandingan tersebut didapatkan informasi bahwa semakin tinggi ukuran data yang dienkripsi, maka semakin baik skor *Bit Avalanche Test* Small AES. Sedangkan jika semakin kecil ukuran data, maka skornya menjadi kurang baik. Hal ini berbanding terbalik dengan SPECK yang dimana semakin tinggi ukuran datanya, maka skornya akan menjadi kurang baik, dan jika ukuran data semakin kecil, maka skornya akan menjadi lebih baik.

V. KESIMPULAN

Algoritma Small AES berhasil diimplementasikan ke perangkat Arduino UNO dengan menggunakan skenario enkripsi dan dekripsi. Dari kedua skenario tersebut, ditetapkan parameter pengujian untuk mengetahui apakah implementasi Small AES telah berhasil, dan pengujian untuk

mengukur dan menganalisis performansi Small AES pada Arduino. Parameter pertama adalah pengujian *Known-plaintext Attack* untuk menebak kunci yang digunakan untuk enkripsi, parameter kedua adalah analisis performansi yang mengukur waktu pemrosesan dan memori yang digunakan, dan parameter ketiga adalah *Bit Avalanche Test* yang membandingkan perubahan kecil yang terjadi pada setiap byte plaintext saat proses enkripsi, yang diharapkan menghasilkan output yang berbeda secara signifikan. Adapun algoritma yang dipilih untuk menjadi pembanding adalah algoritma SPECK.

Dari pengujian pertama diketahu bahwa Small AES berhasil diimplementasikan dengan baik. Adapun dari ketiga skenario pengujian performansi, didapatkan hasil bahwa SPECK memiliki keunggulan dalam performansi kecepatan pemrosesan, baik dari segi enkripsi maupun dekripsi, jika dibandingkan dengan Small AES. Meskipun demikian, Small AES unggul dalam penggunaan memori perangkat dengan menggunakan memori yang lebih sedikit. Secara khusus, melalui pengujian *Bit Avalanche Test*, Small AES lebih baik untuk ukuran data yang cukup banyak, sedangkan SPECK lebih baik untuk ukuran data yang lebih kecil.

Dengan demikian, dapat disimpulkan bahwa SPECK sedikit lebih baik dari sisi performansi waktu pemrosesan jika dibandingkan dengan Small AES. Namun jika fokus utama adalah penggunaan memori yang lebih efisien, maka Small AES lebih baik. Adapun dari tingkat keacakan hasil enkripsi, semakin besar ukuran data maka algoritma Small AES lebih baik dibandingkan SPECK.

Untuk penelitian lebih lanjut, diharapkan dapat mengembangkan dan menggunakan algoritma yang secara performansi lebih baik dan dapat menggunakan perangkat IoT atau sensor lainnya yang dapat diterapkan algoritma kriptografi dalam implementasinya.

REFERENSI

[1] A. M. Abdullah and others, "Advanced encryption standard (AES) algorithm to encrypt and decrypt data," *Cryptography and Network Security*, vol. 16, no. 1, p. 11, 2017.

[2] M. Gaur, R. Gupta, and A. Singh, "Use of AES Algorithm in development of SMS Application on Android Platform," in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, 2021, pp. 1–5.

[3] A. K. Dandekar, S. Pradhan, and S. Ghormade, "Design of AES-512 algorithm for communication network," *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 5, pp. 438–443, 2016.

[4] J. Chandramohan, R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. A. Gopinath, and S. Ranjithkumar, "Intelligent smart home automation and security system using Arduino and Wi-fi," *International Journal of Engineering And Computer Science (IJECS)*, vol. 6, no. 3, pp. 20694–20698, 2017.

[5] J. R. Mahan and K. M. Yeater, "Agricultural applications of a low-cost infrared thermometer," *Comput Electron Agric*, vol. 64, no. 2, pp. 262–267, 2008.

[6] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A survey on sensor-based threats to internet-of-things (iot) devices and applications," *arXiv preprint arXiv:1802.02041*, 2018.

[7] M. Panoff, R. G. Dutta, Y. Hu, K. Yang, and Y. Jin, "On sensor security in the era of IoT and CPS," *SN Comput Sci*, vol. 2, no. 1, p. 51, 2021.

[8] K. Mohamed, M. N. Mohammed Pauzi, F. H. Mohd Ali, S. Ariffin, and others, "Analyse On Avalanche Effect In Cryptography Algorithm," *European Proceedings of Multidisciplinary Sciences*, 2022.

[9] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proceedings of the 52nd annual design automation conference*, 2015, pp. 1–6.

[10] H. M. Mohammad and A. A. Abdullah, "Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 20, no. 3, pp. 551–560, 2022, doi: 10.12928/TELKOMNIKA.v20i3.23297.

[11] A. Sukiatmodjo, "Speed and power consumption comparison between DES and AES algorithm in arduino," UNIKA SOEGIJAPRANATA SEMARANG, 2019.

[12] I. K. Dutta, B. Ghosh, and M. Bayoumi, "Lightweight cryptography for internet of insecure things: A survey," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 475–481.

[13] A. Hamza and B. Kumar, "A review paper on DES, AES, RSA encryption standards," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2020, pp. 333–338.

[14] A. Fotovvat, G. M. E. Rahman, S. S. Vedaei, and K. A. Wahid, "Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes," *IEEE Internet Things J*, vol. 8, no. 10, pp. 8279–8290, 2020.

[15] M. S. Fadhil, A. K. Farhan, and M. N. Fadhil, "A lightweight aes algorithm implementation for secure iot environment," *Iraqi Journal of Science*, pp. 2759–2770, 2021.

[16] W. I. El Sobky, A. A. Isamail, A. S. Mohra, and A. M. Hassan, "Implementation Mini (Advanced Encryption Standard) by Substitution Box in Galois Field (2 4)," in *2021 International Telecommunications Conference (ITC-Egypt)*, 2021, pp. 1–4.

[17] U. A. Adam and M. Patel, "Enhancing the Security of Spin Framework by Combining Min AES with Geoencryption," 2020.

[18] T. B. Singha, R. P. Palathinkal, and S. R. Ahamed, "Securing AES designs against power analysis attacks: a survey," *IEEE Internet Things J*, 2023.

[19] S. Kromodimoeljo, "Teori dan aplikasi kriptografi," *SPK IT Consulting*, 2009.

[20] C. Cid, S. Murphy, and M. J. B. Robshaw, "Small scale variants of the AES," in *International Workshop on Fast Software Encryption*, 2005, pp. 145–162.

[21] A. Kumar and S. Tejani, "S-BOX Architecture," in *Futuristic Trends in Network and Communication Technologies: First International Conference, FTNCT 2018, Solan, India, February 9–10, 2018, Revised Selected Papers 1*, 2019, pp. 17–27.

[22] J. Daemen and V. Rijmen, *The design of Rijndael*, vol. 2. Springer, 2002.

[23] C. De Canniere, A. Biryukov, and B. Preneel, "An introduction to block cipher cryptanalysis," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 346–356, 2006.

[24] A. Kumar and N. Tiwari, "Effective implementation and avalanche effect of AES," *International Journal of Security, Privacy and Trust Management (IJSPTM)*, vol. 1, no. 3/4, pp. 31–35, 2012.

[25] M. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A comparison of cryptographic algorithms: DES, 3DES, AES, RSA and blowfish for guessing attacks prevention," *Journal Computer Science Applications and Information Technology*, vol. 3, no. 2, pp. 1–7, 2018.

[26] C. Echeverri, "Visualization of the Avalanche Effect in CT2," University of Mannheim, 2017.

[27] N. Astuti, I. Arfiani, and E. Aribowo, "Analysis of the security level of modified CBC algorithm cryptography using avalanche effect," in *IOP Conference Series: Materials Science and Engineering*, 2019, p. 12056.