LOGIC: Jurnal Penelitian Informatika Vol. 2, Issue 2, pp. 46–51 (2024)
doi: http://doi.org/10.25124/logic.v2i2.8814

RESEARCH ARTICLE

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

Muhammad Naufal Rabbani and Farah Afianti *

Fakultas Informatika, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: 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.

**Key words:** AES, Bit Avalanche Test, IoT, Pemantauan Suhu, Small AES, SPECK

## Pendahuluan

Keamanan data sangat penting dalam dunia teknologi informasi yang terus berkembang. Salah satu cara untuk 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. 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 [8]. 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. 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.

© The Author 2024. Published by Telkom University.

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.

## Tinjauan Pustaka

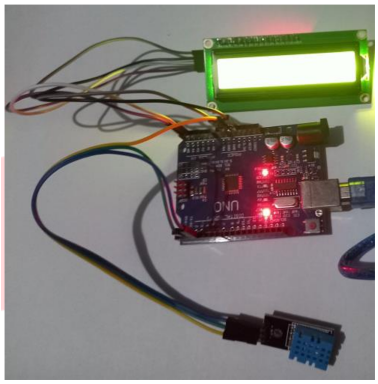
Sudah ada beberapa penelitian telah mengilustrasikan sejumlah pendekatan dalam mengoptimalkan dan meningkatkan performansi *Advanced Encryption Standard* (AES) dalam konteks perangkat terbatas, khususnya Arduino dan perangkat IoT. Dalam penelitian oleh Mohammad dan Abdullah [8], dilakukan penelitian untuk meningkatkan performa AES dengan mengurangi konsumsi daya algoritma dan meningkatkan kinerja kriptografi pada perangkat terbatas dengan sumber daya terbatas. Penelitian lain oleh Sukiatmodjo [9] 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 [10], penelitian lebih umum membahas algoritma kriptografi seperti DES, AES, dan RSA, baik dalam konteks kriptografi simetris maupun asimetris. Adapun dalam penelitian oleh Fotovvat, dkk [11], 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 [12] 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 [13] dalam penelitiannya mencoba untuk mengombinasikan algoritma Mini-AES dengan algoritma Geoencryption untuk memberikan keamanan yang lebih tinggi. Adapun penelitian oleh Singha, dkk [14] 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).

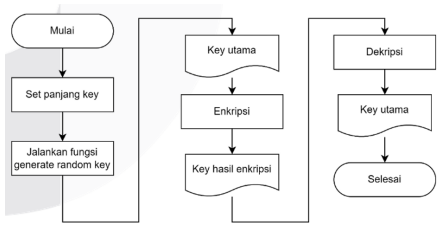
Beberapa penelitian terkait ini membentuk landasan kuat bagi pengembangan analisis performansi Small AES di Arduino. Oleh karena itu, penelitian ini bertujuan untuk mendalami performansi Small AES pada perangkat Arduino dalam konteks IoT, lalu membandingkannya dengan setidaknya satu algoritma kriptografi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

## Hasil dan Pembahasan

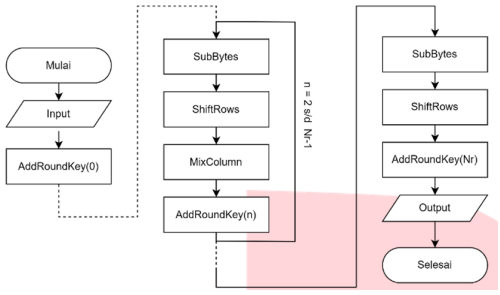
Pada gambar 1 ditampilkan setup dari penelitian ini yang menggunakan perangkat Arduino UNO, Sensor DHT 11, dan LCD 16x2. Sistem yang dibangun pada setup tersebut memiliki dua skenario. Skenario pertama adalah skenario dimana arduino melakukan proses enkripsi, setelah itu data yang telah dienkripsi dikirim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



**Gambar 1.** Setup Penelitian.




**Gambar 2.** Inisialisasi Kunci.



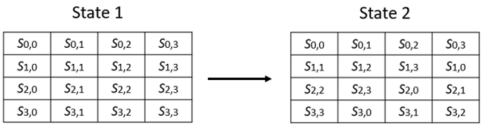

**Gambar 3.** Alur Enkripsi Small AES.

kepada arduino yang telah dienkripsi, kemudian pada arduino dilakukan dekripsi untuk menampilkan teks pada LCD.

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.

### Proses Enkripsi

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



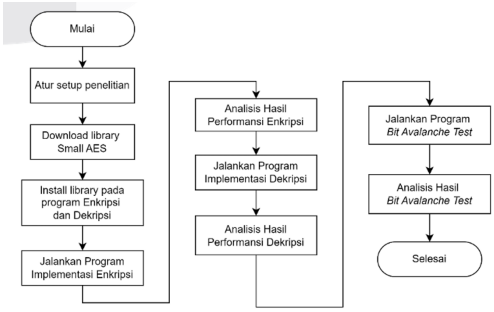

**Gambar 4.** Alur Enkripsi Small AES.

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.

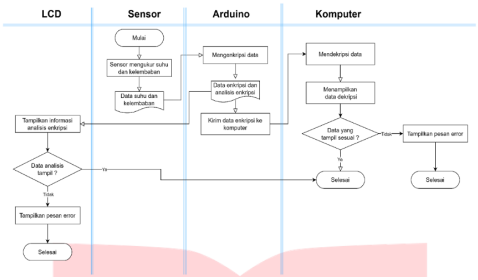
Setelah operasi SubBytes, blok data mengalami operasi ShiftRows, di mana setiap baris dirotasi ke kiri sesuai dengan nomor barisnya. Pada GAMBAR 4 dijelaskan bagaimana pada Small AES 256, blok data 4x4 dirotasi sehingga baris pertama tidak bergeser, baris kedua mengalami pergeseran satu posisi ke kiri, 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: Berikut adalah format LaTeX untuk matriks yang sesuai dengan gambar:

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

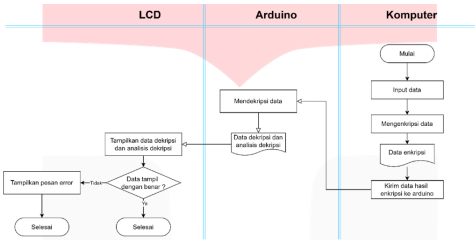
Matriks MDS adalah matriks invertibel dengan elemen dari bidang Galois GF(28). Setiap kolom diubah dengan matriks MDS, yang meningkatkan difusi dan kompleksitas enkripsi. GF sendiri adalah singkatan dari Galois Field atau Bidang Galois. Dalam konteks kriptografi dan matematika diskrit, Bidang Galois 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(28) karena mempunyai 28 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.




**Gambar 5.** Alur Implementasi Small AES .




**Gambar 6.** Alur Implementasi Enkripsi.




**Gambar 7.** Alur Implementasi Deskripsi.

### 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 5 dijelaskan bagaimana alur implementasi Small AES pada Arduino UNO untuk penelitian ini.

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.

### 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.

### Implementasi Deskripsi

Pada *Error*! *Reference source not found*. dijelaskan bagaimana skenario dekripsi dilakukan. Pertama, program di komputer mengenkripsi sebuah teks yang nantinya akan ditampilkan pada LCD oleh arduino. Pada program enkripsi ini plaintext akan di*input*kan 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 di*input*kan pada program enkripsi tadi. Selain itu, ada juga informasi analisis dekripsi yang tampil. Namun, jika data yang ditampilkan tidak sesuai dengan data yang di*input*kan pada program, maka akan tampil sebuah pesan *error*.

### Evaluasi

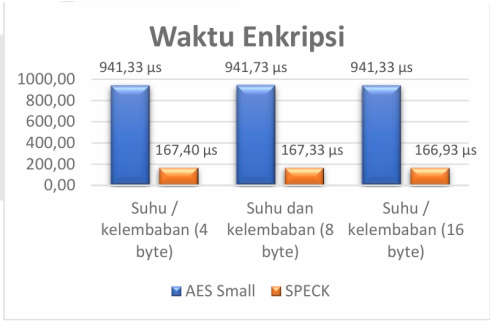
#### Skenario Pengujian

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:

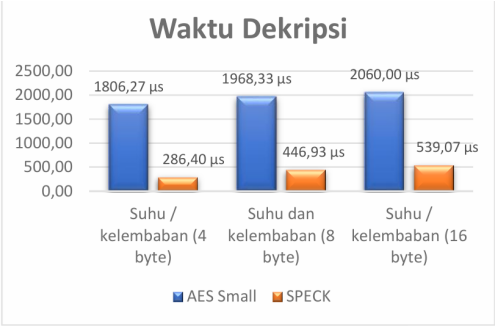
$$\text{Skor Avalanche Test} = \frac{\text{Jumlah bit yang berbeda antara ciphertext dan plaintext}}{\text{Jumla htotal bit dalam ciphertext}} \quad (2)$$

**Table 1.** Hasil percobaan Known-plaintext Attack

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




**Gambar 8.** Alur Implementasi Deskripsi.




**Gambar 9.** Statistik Waktu Dekripsi.

**Table 2.** Statistik Bit Avalanche Test

| | 4 *Byte* | | 8 B *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 % |

**Table 3.** Statistik Penggunaan Memori

| | Enkripsi | | Deskripsi | |
|---|---|---|---|---|
| | **Snall 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 |

**Hasil dan Analisa Pengujian**

1. ***Known-plaintext Attack***

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. **Perbandiingan Performansi**

a. Waktu enkripsi dan Deskrripsi

Panjang data 8 *byte*. Sedangkan algoritma SPECK hanya membuthkan waktu 167.40 $\mu$s untuk panjang data 4 *byte*, 167.33 untuk panjang data 8 *byte* dan 166.93 $\mu$s 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 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 $\mu$s untuk panjang data 4, 1968.33 untuk panjang data 8 *byte* dan 2060 $\mu$s untuk panjang data 16 *byte* . Sedangkan SPECK hanya membutuhkan waktu 286.40 untuk panjang data 4 by$\mu$ste, 446.93 $\mu$s untuk panjang data 8 *byte* dan 539.07 $\mu$s untuk panjang data 16 *byte*.

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 $\mu$s sampai 774.4 $\mu$s. Sedangkan pada GAMBAR 9, didapatkan selisih waktu dekripsi antara Small AES dengan SPECK pada skenario pertama mencapai 1519.87 $\mu$s, skenario kedua 1541.4 $\mu$s, dan skenario ketiga mencapai 1520.93 $\mu$s.

b. Penggunaan Memori

Performa kedua algoritma dari segi penggunaan memori dapat dilihat pada 3. 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.

c. Bit Avalanche Test

Pada 2 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.

## 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.

## Daftar Pustaka


1. Abdullah AM, et al. Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data. Cryptography and Network Security. 2017;16(1):11.
2. Gaur M, Gupta R, Singh A. 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 Directions) (ICRITO); 2021. p. 1-5.
3. Dandekar AK, Pradhan S, Ghormade S. Design of AES-512 Algorithm for Communication Network. International Research Journal of Engineering and Technology (IRJET). 2016;3(5):438-43.
4. Chandramohan J, Nagarajan R, Satheeshkumar K, Ajithkumar N, Gopinath PA, Ranjithkumar S. Intelligent Smart Home Automation and Security System Using Arduino and Wi-Fi. International Journal of Engineering and Computer Science (IJECS). 2017;6(3):20694-8.
5. Mahan JR, Yeater KM. Agricultural Applications of a Low-Cost Infrared Thermometer. Future Journal (Assuming journal name). Tahun Tidak Diketahui.
6. Sikder AK, Petracca G, Aksu H, Jaeger T, Uluagac AS. A Survey on Sensor-Based Threats to Internet-of-Things (IoT) Devices and Applications. arXiv preprint. 2018;arXiv:1802.02041.
7. Panoff M, Dutta RG, Hu Y, Yang K, Jin Y. On Sensor Security in the Era of IoT and CPS. SN Computer Science. 2021;2(1):51.
8. Mohamed K, Pauzi MNM, Ali FHM, Ariffin S, et al. Analyse On Avalanche Effect In Cryptography Algorithm. In: European Proceedings of Multidisciplinary Sciences; 2022. .
9. Sukiatmodjo A. Speed and power consumption comparison between DES and AES algorithm in Arduino; 2019.
10. Hamza A, Kumar B. A review paper on DES, AES, RSA encryption standards. In: 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART); 2020. p. 333-8.
11. Fotovvat A, Rahman GME, Vedaei SS, Wahid KA. Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. IEEE Internet of Things Journal. 2020;8(10):8279-90.
12. Fadhil MS, Farhan AK, Fadhil MN. A lightweight AES algorithm implementation for secure IoT environment. Iraqi Journal of Science. 2021:2759-70.
13. Adam UA, Patel M. Enhancing the Security of Spin Framework by Combining Min AES with Geoencryption; 2020.
14. Singha TB, Palathinkal RP, Ahamed SR. Securing AES designs against power analysis attacks: a survey. IEEE Internet of Things Journal. 2023.
15. Cid C, Murphy S, Robshaw MJB. Small scale variants of the AES. In: International Workshop on Fast Software Encryption; 2005. p. 145-62.
16. Kumar A, Tejani S. 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. p. 17-27.
17. Daemen J, Rijmen V. The Design of Rijndael. vol. 2. Springer; 2002.
18. Kumar A, Tiwari N. Effective implementation and avalanche effect of AES. International Journal. Complete journal details and year are required for proper citation.
19. Wahid MNA, Ali A, Esparham B, Marwan M. A comparison of cryptographic algorithms: DES, 3DES, AES, RSA and blowfish for guessing attacks prevention. Journal Computer Science Applications and Information Technology. 2018;3(2):1-7.
20. Echeverri C. Visualization of the Avalanche Effect in CT2; 2017.
21. Astuti N, Arfiani I, Aribowo E. Analysis of the security level of modified CBC algorithm cryptography using avalanche effect. In: IOP Conference Series: Materials Science and Engineering; 2019. p. 12056.