PENGARUH KOMPONEN ALGORITMA AES TERHADAP HASIL UJI *STRICT AVALANCHE CRITERION* (SAC) DARI ALGORITMA AES

Novita Angraini[1)], Muhammad Wibisono[2)], Nugroho Jati[3)]

Badan Siber dan Sandi Negara
Jl. Harsono RM no 70 Ragunan Pasarminggu Jakarta Selatan 12550
Email : novita.angraini@bssn.go.id, muhammad.wibisono@bssn.go.id, nugroho.jati@bssn.go.id

Abstrak

Advanced Encryption Standard atau yang lebih dikenal dengan nama Algoritma AES merupakan suatu standar algoritma dan digunakan secara luas pada berbagai aplikasi kriptografi. Algoritma AES terdiri dari empat komponen (operasi) utama, yaitu Substitution Box (S-box), Shiftrows, Mixcolumn, dan Addroundkey. Pada penelitian kali ini akan dilakukan dengan menghilangkan penggunaan masing-masing komponen algoritma AES secara satu-persatu. Setelah itu, dianalisis pengaruh dari menghilangkan penggunaan komponen tersebut dengan uji SAC dan dibandingkan dengan hasil uji SAC algoritma AES asli. Hasilnya pada pengujian SAC dengan variable bebas plainteks, hanya algoritma AES tanpa Addroundkey yang tetap memenuhi uji SAC. Sedangkan pada pengujian SAC dengan variable bebas kunci, semua algoritma AES tetap memenuhi uji SAC. Hal ini dikarenakan terdapat pengaruh dari proses pembangkitan kunci yang digunakan algoritma AES.

Kata kunci: *Algoritma AES, uji SAC, Substitution Box (S-box), Shiftrows, Mixcolumn, Addroundkey.*

1. Pendahuluan

Pada era saat ini, teknologi komunikasi berkembang sangat pesat. Teknologi komunikasi merupakan teknologi yang mendukung pertukaran informasi. Komunikasi itu sendiri bisa diartikan sebagai proses proses pertukaran informasi dan data antara satu dengan yang lainnya. Ancaman terhadap teknologi komunikasi atau sistem informasi serta pola serangan terhadap sistem keamanan yang dibangun, secara umum berkembang lebih cepat jika dibandingkan oleh metode keamanan untuk mengantisipasi ancaman tersebut. Oleh karena itu, teknik untuk menjaga kerahasiaan data sangat dibutuhkan seiring berkembangnya teknik informasi dan komunikasi. Hal ini dapat diatasi dengan menggunakan teknik enkripsi yang merupakan bagian dari kriptografi. Kriptografi merupakan studi matematika yang berhubungan dengan aspek pengamanan informasi seperti kerahasiaan (*confidentiality*), integritas data (*data integrity*), otentikasi entitas (*entity authentication*), dan otentikasi keaslian data (*data origin authentication*) [1].

Sistem kriptografi dibagi menjadi tiga bagian, yaitu: sistem simetrik, asimetrik, dan fungsi hash. Sistem simetrik sendiri terbagi menjadi dua, yaitu *stream cipher* dan *Block Cipher*. *Block Cipher* adalah suatu fungsi yang memetakan *n*-bit blok teks terang ke *n*-bit blok teks sandi dengan *n* merupakan panjang blok [1].

Pada umumnya algoritma *Block Cipher* harus memenuhi konsep difusi dan konfusi. Konfusi adalah penggunaan transformasi penyandian dengan tujuan mempersulit mencari hubungan statistik antara *ciphertext* dan *plaintext* dengan cara menyembunyikan ciri-ciri statistik *plaintext*. Salah satu komponen dalam *Block Cipher* yang menggunakan prinsip konfusi adalah *substitution box* (*S-box*). Difusi adalah menyebarkan statistik dari *plaintext* ke dalam struktur statistik yang melibatkan kombinasi yang panjang dari bit-bit dalam *plaintext*. Difusi dapat dilakukan melalui permutasi [2]. Pengujian yang dilakukan untuk keseluruhan algoritma *Block Cipher* dapat menggunakan uji *Strict Avalanche Criterion* (SAC).

Algoritma *Block Cipher* Rijndael yang diajukan oleh Joan Daemen dan Vincent Rijmen terpilih sebagai AES karena kombinasi keamanan, unjuk kerja, efisiensi, kemampuan untuk diimplementasikan dan fleksibilitas dinilai jauh lebih baik dibandingkan finalis lainnya. Selanjutnya NIST mempublikasikan FIPS PUB 197 yang menyatakan bahwa algoritma AES sebagai algoritma standar pada tanggal 26 November 2001. Algoritma AES terdiri dari empat komponen (operasi) utama, yaitu *Substitution Box (S-box), Shiftrows, Mixcolumn, dan Addroundkey* [3]. Keempat komponen tersebut memiliki kontribusi berbeda pada kekuatan dan keacakan serta hasil uji algoritma AES.

Oleh karena itu, pada penelitian kali ini akan dilakukan pengujian SAC bagi algoritma AES dengan menghilangkan masing-masing komponen Algoritma AES secara satu-persatu agar dapat terlihat pengaruh dari masing-masing komponen terhadap algoritma AES.

Tujuan penelitian ini bertujuan untuk mengetahui pengaruh masing-masing komponen terhadap algoritma AES berdasarkan hasil uji SAC.

Berikut ini adalah landasan teori yang digunakan pada penelitian ini.

A. *Strict Avalanche Criterion* (SAC)

Sebuah fungsi $f: \{0,1\}^n \rightarrow \{0,1\}^n$ memenuhi SAC jika untuk semua *i* dan *j* $\in$ (1,2, … , *n*), perubahan bit *input* ke-*i* akan mengubah bit *output* ke-*j* dengan probabilitas tepat ½ [4].

Dari penjelasan tersebut dapat diformulasikan menjadi persamaan (1):

$$\frac{1}{2^n} W\left(a_j^{e_i}\right) = \frac{1}{2} \text{……(1)}$$

Dari persamaan (1) dapat dimodifikasi untuk menentukan parameter SAC, $k_{SAC}(i,j)$, menjadi persamaan (2):

$$k_{SAC}(i,j) = \frac{1}{2^n} W\left(a_j^{e_i}\right) = \frac{1}{2} \text{ …… (2)}$$

$k_{SAC}(i,j)$ berada pada rentang [0,1] dan merupakan probabilitas perubahan bit *output* ke-*j* ketika bit *input* ke-*i* diubah. Jika probabilitas $k_{SAC}(i,j)$ tidak sama dengan ½ untuk setiap pasangan (*i*, *j*), maka tidak memenuhi SAC. Untuk nilai *n* yang besar akan sulit untuk mendapatkan nilai yang tepat memenuhi SAC. Oleh karena itu pada pengujian SAC penelitian ini, algoritma yang dikatakan memenuhi uji SAC adalah algoritma yang memiliki nilai uji SAC sama atau mendekati nilai uji SAC algoritma AES asli.

B. *Advanced Encryption Standard* (AES)

Algoritma *Block Cipher* Rijndael yang diajukan oleh Joan Daemen dan Vincent Rijmen terpilih sebagai *Advanced Encryption Standard* (AES) pada tahun 2001. Algoritma ini adalah algoritma simetrik standar berbasis *Block Cipher* yang mengenkripsi 128-bit blok *input* menjadi 128-bit blok *output*. Algoritma AES menggunakan panjang kunci yang beragam, yaitu 128, 192, dan 256 bit. Berdasarkan panjang kuncinya, AES dikelompokkan menjadi AES-128, AES-192, dan AES-256. AES menggunakan jumlah *round* Nr beragam bergantung dari panjang kunci yang digunakan [3]. Tabel 1 merupakan perbandingan jumlah *round* pada AES berdasarkan panjang kunci yang digunakan:

Tabel 1 Kombinasi kunci-blok-*round* AES

	Key Length (Nk words)	*Block Size (Nb words)*	*Number of Rounds (Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

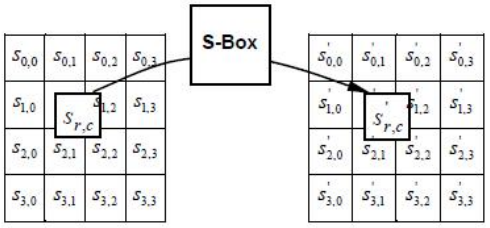
Untuk memperoleh *cipher* pada algoritma AES, pada blok *input* dilakukan operasi *Addroundkey*(), kemudian dilakukan transformasi berdasarkan fungsi *round* sebanyak 10, 12 atau 14 kali (tergantung panjang kunci yang digunakan). Adapun transformasi dalam fungsi *round* AES sebagai berikut [3]:

a) Substitusi *byte* menggunakan *S-box* (*S-box*).
b) *Shifting rows* dari *state array* (*ShiftRow*)
c) *Mixing data* dalam setiap kolom dari *state array* (*Mixcolumn*)
d) Penambahan *round key* ke *state* (*Addroundkey*)

Untuk *round* terakhir sedikit berbeda dengan *Nr-1 round* pertama yaitu tidak terdapat transformasi *Mixcolumn*().

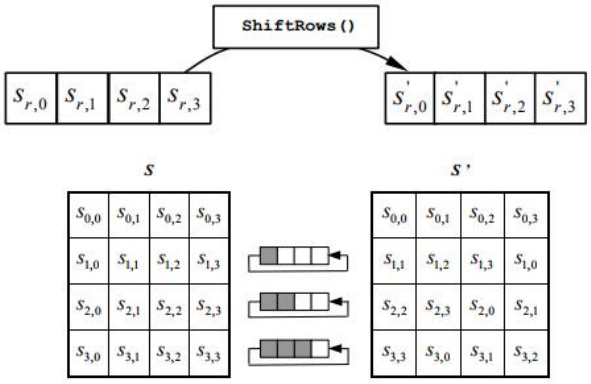
Transformasi penyandian yang digunakan dalam algoritma AES adalah sebagai berikut [3]:

1) Transformasi *SubBytes()*
   Transformasi *SubBytes()* adalah substitusi *byte nonlinear* yang mengoperasikan setiap *byte* dari *state* menggunakan tabel substitusi (*S-box*) yang *invertible*.
   Proses transformasinya digambarkan pada gambar 1 di bawah ini.



Gambar 1. *Transformasi SubBytes()*

2) Transformasi *ShiftRow()*
   Shiftrows merupakan suatu transformasi yang merotasi *bytes* pada tiga baris terakhir dari *state* sebanyak beberapa *bytes*. Proses transformasi *Shiftrows* digambarkan pada gambar 2 di bawah ini.



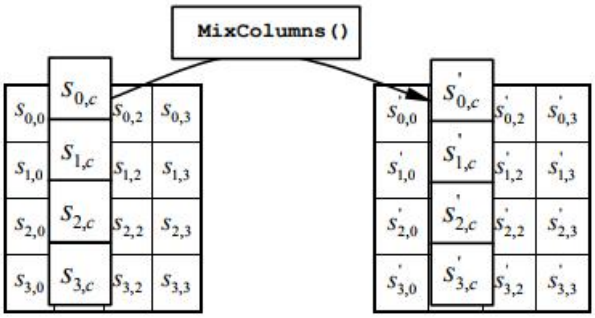
Gambar 2. *Transformasi ShiftRow()*

3) Transformasi *Mixcolumns()*
   Transformasi *Mixcolumns*() beroperasi pada *state* kolom per kolom yang direpresentasikan dalam bentuk perkalian matriks pada GF(2^8). Persamaan (3) merupakan proses perkalian matriks pada *Mixcolumn*.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

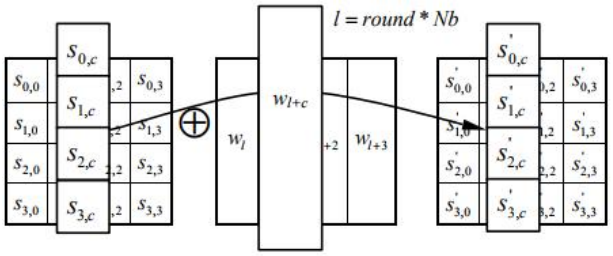
untuk $0 < c \leq Nb$ .....(3)

Proses transformasi *Mixcolumns*() digambarkan pada gambar 3 berikut:



Gambar 3. *Transformasi Mixcolumns()*

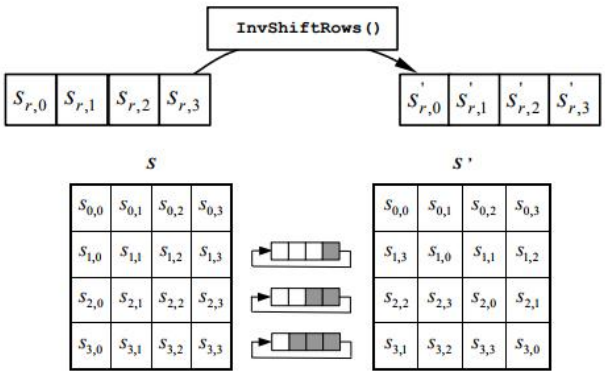
4) Transformasi *Addroundkey()*
Transformasi *Addroundkey()* merupakan transformasi *round key* dengan *state* menggunakan operasi XOR sederhana. Untuk transformasi *Addroundkey()* digambarkan pada gambar 4.



Gambar 4. *Transformasi Addroundkey()*

Untuk invers penyandian (proses dekripsi) pada algoritma AES adalah sebagai berikut:

1) Transformasi *InvShiftRow*()
Transformasi *InvShiftRow*() merupakan invers dari transformasi *ShiftRow*(). *Bytes* pada tiga baris terakhir dari *state* dirotasi sejumlah *bytes* tertentu dan untuk baris pertama, *r = 0* tidak dilakukan pergeseran. Tiga baris terbawah dirotasi sebanyak Nb – *shift*(r,Nb) *bytes*, dimana nilai *shift*(r, Nb) bergantung pada jumlah baris. Proses transformasi *InvShiftRow*() digambarkan pada gambar 5 di bawah ini:



Gambar 5. *Transformasi InvShiftRow()*

2) Transformasi *InvSubByte*()
InvSubByte() merupakan invers dari transformasi substritusi *byte*, yang mana invers *S-box* diterapkan untuk setiap *byte* dati *state*.
3) Transformasi *InvMixcolumns*()
InvMixcolumns() merupakan invers dari transformasi *Mixcolumns*(). *InvMixcolumns*() mengoperasikan *state* kolom per kolom. Persamaan (4) merupakan proses perkalian matriks pada *InvMixcolumns*.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

untuk $0 < c \leq Nb$ .....(4)

4) Transformasi Invers *Addroundkey*()
Invers *Addroundkey*() sama halnya dengan transformasi *Addroundkey*(), karena hanya melibatkan operasi XOR.

Untuk penjadwalan kunci pada algoritma AES akan dijelaskan di bawah ini:
Kunci *round* diturunkan dari kunci penyandian melalui proses *key schedule*. Proses ini terdiri dari dua buah komponen, yaitu Ekspansi Kunci dan Pemilihan Kunci *Round*. Prinsip dasar dari proses ini adalah:

- Jumlah total dari bit kunci *round* sama dengan panjang blok *input* dikalikan dengan jumlah *round* ditambah 1 (misalnya, untuk panjang blok *input* sebesar 128-bit dengan jumlah *round* 10 *round*, maka total bit kunci *round* yand dibutuhkan adalah 1408 bit).
- Mengekspansi kunci penyandian.
- Kunci *round* diperoleh dari kunci yang telah diekspansi dengan cara berikut: Kunci *round* terdiri dari *Nb word* pertama, bagian kedua dari *Nb word* kedua dan demikian selanjutnya.

Ekspansi Kunci dari algoritma AES berdasarkan FIPS 197 [3] adalah sebagai berikut:
Algoritma AES menggunakan kunci penyandian, *K*, dan menjalankan rutin kunci ekspansi untuk menghasilkan *key schedule*. Ekspansi kunci secara keseluruhan menghasilkan Nb(Nr + 1) *word*. Algoritma tersebut memerlukan himpunan inisial dari Nb *word* dan setiap Nr *round* memerlukan Nb *word* dari data kunci. *Key schedule* yang dihasilkan terdiri dari sebuah *array* linear dengan panjang 4-*byte word*, dinotasikan oleh [*wi*], dengan nilai *i* berada di dalam interval $0 \leq i <$ Nb(Nr + 1).

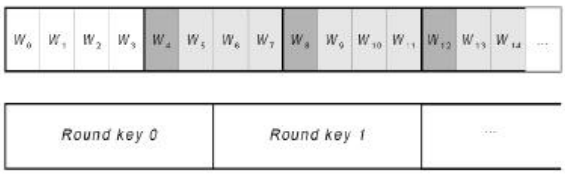
Ekspansi dari kunci *input* menjadi *key schedule* diproses dengan fungsi *SubWord*() dan *RotWord*(). *SubWord*() adalah sebuah fungsi yang memproses 4-*byte input word* dan mensubstitusikannya ke dalam *S-box* untuk menghasilkan sebuah *output word*. Fungsi *RotWord*() menggunakan *word* $[a_0, a_1, a_2, a_3]$ sebagai *input*, kemudian menerapkan permutasi siklik dan memberikan nilai balikan *word* $[a_1, a_2, a_3, a_0]$. *Array*

word konstanta *round*, Rcon[i], berisikan nilai $[x^{i-1},\{00\},\{00\},\{00\}]$, dengan x^{i-1} merupakan pangkat dari *x* (*x* dinotasikan sebagai {02}) di *field* GF(2^8).

Nk *word* pertama dari kunci yang diekpansi diisi dengan kunci penyandian. Setiap *word* berikutnya, $w[i]$, sama dengan nilai XOR dari *word* sebelumnya, $w[i-1]$ dan posisi *word* Nk terawal, $w[i-Nk]$. Untuk *word* yang berada pada posisi yang merupakan kelipatan dari Nk, sebuah transformasi dilakukan pada $w[i-1]$ sebelum proses XOR, diikuti oleh XOR dengan konstanta *round*, Rcon[i]. Transformasi ini terdiri dari pergeseran siklik dari *bytes* di dalam *word* (*RotWord*()), diikuti dengan menerapkan tabel *lookup* untuk seluruh 4 *byte* dari *word* (*SubWord*()).

Ekspansi kunci untuk kunci penyandian 256 bit (Nk=8) sedikit berbeda dengan kunci penyandian dengan panjang 128 dan 192 bit. Jika Nk=8 dan nilai $i-4$ merupakan kelipatan dari Nk, maka *SubWord*() dilakukan terhadap $w[i-1]$ sebelum proses XOR dilakukan.

Pemilihan Kunci *Round*
Kunci *round* ke-*i* diperoleh dari word *buffer* kunci *round* *w*[Nb*i*] hingga *w*[Nb*(*i*+1)]. Gambar 6 merupakan ilustrasi dari pemilihan kunci *round*.



Gambar 6. *Pemilihan Kunci Round*

Metodologi Penelitian
Pengujian SAC pada masing-masing algoritma AES dilakukan dalam dua tahap. Tahap pertama adalah menghasilkan sebagai sampel yang digunakan sebanyak 20000 sebagai variabel bebas yang telah ditetapkan. Saat plainteks diperlakukan sebagai variabel bebas maka kunci sebagai variabel kontrol dibuat konstan dengan nilai nol. Hal ini dilakukan untuk mengetahui sifat difusi dari masing-masing algoritma AES. Demikian pula, ketika kunci diperlakukan sebagai variabel bebas maka plainteks sebagai variabel kontrol dibuat konstan dengan nilai nol. Hal ini dilakukan untuk mengetahui sifat konfusi dari masing-masing algoritma AES. Penggunaan nilai nol konstan pada variabel kontrol untuk menghilangkan pengaruh variabel kontrol. Output dari proses ini adalah nilai variabel bebas (cipherteks).

Tahap kedua adalah pengujian sampel yang telah dihasilkan dengan menggunakan uji SAC. Algoritma AES yang digunakan adalah algoritma AES-128 dengan jumlah *round* sebanyak 10 *round*. Algoritma yang dikatakan memenuhi uji SAC adalah algoritma yang memiliki nilai uji SAC sama atau mendekati nilai uji SAC algoritma AES asli.

2. Pembahasan

Berdasarkan hasil uji SAC yang telah dilakukan menggunakan variabel bebas plainteks, dapat dilihat bahwa algoritma AES asli dan algoritma AES tanpa *Addroundkey* yang dapat memenuhi uji SAC. Sedangkan pada algoritma AES tanpa *S-box*, tanpa *Shiftrows*, dan tanpa *Mixcolumn* tidak dapat memenuhi uji SAC. Hal ini dapat dilihat karena terdapat hasil uji SAC yang bernilai nol. Oleh karena itu pada pengujian SAC menggunakan variabel bebas plainteks, hanya algoritma AES asli dan algoritma AES tanpa *Addroundkey* yang mempunyai sifat difusi yang baik. Hasil pengujian SAC menggunakan variabel bebas plainteks dapat dilihat pada Tabel 2.

Tabel 2. Pengujian SAC dengan menggunakan variabel bebas Plainteks

No.	Algoritma	Nilai SAC (%)	
		Min	Maks
1	AES	48,67	51,31
2	AES tanpa *S-box*	0	1
3	AES tanpa *Shiftrows*	0	51,28
4	AES tanpa *Mixcolumn*	0	64,63
5	AES tanpa *Addroundkey*	48,53	51,55

Berdasarkan hasil uji SAC yang telah dilakukan menggunakan variabel bebas kunci, dapat dilihat bahwa semua algoritma AES yang diuji dapat memenuhi uji SAC. Uji SAC algoritma AES tanpa *S-box* memiliki nilai minimum 48,54% dan nilai maksimum 51,63%. Uji SAC algoritma AES tanpa *Shiftrows* memiliki nilai minimum 48,69% dan nilai maksimum 51,33%. Uji SAC algoritma AES tanpa *Mixcolumn* memiliki nilai minimum 48,59% dan nilai maksimum 51,27%. Uji SAC algoritma AES tanpa addrounkey memiliki nilai minimum 48,53% dan nilai maksimum 51,55%. Oleh karena itu pada pengujian SAC menggunakan variabel bebas kunci, semua algoritma yang diuji mempunyai sifat konfusi yang baik. Hasil pengujian SAC menggunakan variabel bebas kunci dapat dilihat pada Tabel 3.

Tabel 3. Pengujian SAC dengan menggunakan variabel bebas Kunci

No.	Algoritma	Nilai SAC (%)	
		Min	Maks
1	AES	48,64	51,52
2	AES tanpa *S-box*	48,54	51,63
3	AES tanpa *Shiftrows*	48,69	51.33
4	AES tanpa *Mixcolumn*	48,59	51,27
5	AES tanpa *Addroundkey*	48,53	51,55

Pada pengujian SAC yang telah dilakukan menggunakan variabel bebas kunci, dapat dilihat bahwa semua algoritma AES yang diuji dapat memenuhi uji SAC. Hal ini sangat berbeda dengan pengujian SAC yang menggunakan variabel bebas plainteks, bahwa hanya algoritma AES asli dan algoritma AES tanpa *Addroundkey* yang dapat memenuhi uji SAC. Ini dikarenakan terdapat pengaruh dari proses pembangkitan kunci dari algoritma AES. Pada pengujian SAC yang menggunakan variabel bebas plainteks, inputan untuk pembangkitan kunci yang digunakan hanya bernilai nol, sehingga kunci yang dihasilkan tidak acak. Sedangkan pada pengujian SAC yang menggunakan variabel bebas kunci, inputan untuk pembangkitan kunci yang digunakan bernilai random dan selalu berganti nilainya, sehingga kunci yang dihasilkan acak. Pembangkitan kunci pada algoritma AES sendiri memiliki beberapa proses, dapat dilihat pada FIPS 197 [3], sehingga dari kunci yang digunakan itu juga sangat berpengaruh terhadap keselurahan dari algoritma AES.

3. Kesimpulan

Berdasarkan hasil penelitian ini, dapat disimpulkan hal-hal sebagai berikut:

1. Pengujian SAC yang telah dilakukan menggunakan variabel bebas kunci, semua algoritma AES yang diuji dapat memenuhi uji SAC.
2. Pada pengujian SAC yang menggunakan variabel bebas plainteks, bahwa hanya algoritma AES asli dan algoritma AES tanpa *Addroundkey* yang dapat memenuhi uji SAC.
3. Perbedaan dari hasil uji SAC tersebut dikarenakan terdapat pengaruh dari proses pembangkitan kunci dari algoritma AES.

Daftar Pustaka

[1] Menezes, Alfred J., Paul C. Van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC press LLC : Boca Raton, 1997.
[2] Stallings, William, Cryptography and Network Security, Fifth edition, pearson education. 2011.
[3] NIST, *Federal Information Processing Standards Publication (FIPS) 197*, Springfield : National Institute of Standards and Technology (NIST), 2001.
[4] S. Kavut and M.D. Yucel, "On Some Cryptographic Properties of Rijndael", Middle East Technical University, 2000.

Biodata Penulis

Novita Angraini, memperoleh gelar Sarjana Sains Terapan Teknik Persandian (S.S.T.TP), Jurusan Teknik Persandian Sekolah Tinggi Sandi Negara, lulus tahun 2013. Saat ini menjadi pegawai negeri sipil di Badan Siber dan Sandi Negara.

Muhammad Wibisono, memperoleh gelar Sarjana Sains Terapan Teknik Persandian (S.S.T.TP), Jurusan Teknik Persandian Sekolah Tinggi Sandi Negara, lulus tahun 2014. Saat ini menjadi pegawai negeri sipil di Badan Siber dan Sandi Negara.

Nugroho Jati, Saat ini menjadi pegawai negeri sipil di Badan Siber dan Sandi Negara.

ISSN : 2302-3805