Analisis Kinerja Algoritma Kriptografi Kandidat *Advanced Encryption Standard* (AES) pada *Smartphone*

Ahmad Farisi
STMIK GI MDP; Jl. Rajawali No.14 Palembang, 0711-376400
Program Studi Sistem Informasi, STMIK GI MDP, Palembang
e-mail: ahmadfarisi@mdp.ac.id

Abstract

Rijndael algorithm is the choosen algorithm for AES algorithm. However several previous studies have found the performance of rijndael algorithm on smartphones was under the other AES candidate algorithm such as serpent and twofish. This study compares the performance of rijndael, twofish, and serpent algorithms (the three major AES candidate algorithms) applied to smartphones. This study uses 3 devices with Android operating system and 3 devices with iOS operating system in the experiment. The experiments conducted in this study measure the performance of time, memory usage, and CPU percentage in the process of encryption and decryption performed using different key characters such as alphabet, numeric, alphanumeric, and alphanumeric and symbol. From 720 experiments conducted, this study found that serpent has the fastest time for encryption (0,025 ms) and decryption (0,022 ms). While the most efficient use of memory (0,032 KB) for encryption and decryption is done by twofish and rijndael. And the most efficient percentage of CPU usage (0,025 % for encryption and 0,038% for decryption) is done by serpent. The results of this experiment show that the performance of rijndael on smartphone is not better than other AES candidate algorithms, although the rijndael is the choosen AES algorithm.

Keywords— *AES, rijndael, serpent, twofish, encryption, decryption.*

Abstrak

Algoritma rijndael adalah algoritma yang terpilih sebagai algoritma AES. Namun beberapa penelitian terdahulu menemukan kinerja algoritma rijndael berada di bawah algoritma kandidat AES lainnya seperti serpent dan twofish dalam kasus enkripsi dan dekripsi yang dilakukan pada smartphone. Penelitian ini membandingkan kinerja algoritma rijndael, twofish, dan serpent (tiga besar algoritma kandidat AES) yang diterapkan pada smartphone. Penelitian ini menggunakan 3 device dengan sistem operasi Android dan 3 device dengan sistem operasi iOS dalam percobaannya. Percobaan yang dilakukan dalam penelitian ini mengukur kinerja waktu, penggunaan memory, dan persentase CPU dalam proses enkripsi dan dekripsi yang dilakukan menggunakan karakter kunci yang berbeda-beda seperti alfabet, numerik, alfanumerik, dan alfanumerik dan simbol. Dari 720 kali percobaan yang dilakukan, penelitian ini menemukan bahwa algoritma serpent memiliki waktu tercepat untuk proses enkripsi (0,025 ms) dan dekripsi (0,022 ms). Sementara penggunaan memory yang paling efisien (0,032 KB) untuk proses enkripsi dan dekripsi dilakukan oleh algoritma rijndael dan twofish. Dan persentase penggunaan CPU yang paling efisien (0,025 % untuk proses enkripsi dan 0,038% untuk proses dekripsi) dilakukan oleh algoritma serpent. Hasil percobaan ini menunjukkan bahwa dalam penerapannya pada smartphone, kinerja algoritma rijndael tidak lebih baik dari algoritma kandidat AES lainnya, meskipun algoritma rijndael adalah algoritma AES yang terpilih.

Kata kunci— *AES, rijndael, serpent, twofish, enkripsi, dekripsi.*

1. PENDAHULUAN

National Institute of Standards and Technology (NIST) pada Oktober 2000 mengumumkan bahwa rijndael adalah algoritma yang terpilih sebagai *Advanced Encryption Standard* (AES) menggantikan *Data Encryption Standard* (DES) yang dianggap tidak aman lagi sebagai standar kriptografi kunci simetri [1]. Pemilihan algoritma AES tersebut dilakukan berdasarkan kriteria kategori dan panjang kunci, ukuran blok, dan rancangan algoritma yang dianggap memiliki tingkat keamanan paling tinggi [2]. Beberapa algoritma lainnya yang menjadi kandidat di peringkat 5 besar secara berturut-turut setelah rijndael adalah algoritma serpent, twofish, RC6, dan MARS.

Sebagian besar penelitian terdahulu melakukan analisis terhadap kinerja algoritma rijndael yang diuji pada berbagai *platform* dan *device*, dan dibandingkan dengan beberapa algoritma lainnya. Sementara sebagian lainnya mengusulkan pendekatan-pendekatan dan modifikasi-modifikasi terhadap algoritma rijndael tersebut. Hal ini dilakukan karena beberapa penelitian menemukan kinerja rijndael berada di bawah algoritma kandidat AES lainnya seperti serpent dan twofish dalam kasus enkripsi dan dekripsi yang dilakukan pada beberapa perangkat bergerak dengan beberapa sistem operasi, walaupun rijndael merupakan algoritma terpilih untuk AES.

Seperti penelitian [3] yang melakukan optimasi terhadap algoritma AES untuk diterapkan pada *smartphone*. Optimasi pada penelitian ini dilakukan atas dasar penelitian sebelumnya yang dilakukan oleh [4] yang menyatakan bahwa algoritma AES belum bekerja baik pada perangkat dengan arsitektur processor ARM. Sehingga penelitian ini melakukan optimasi dalam bentuk mengurangi penggunaan ‘for’ dalam beberapa logika perulangan dalam AES, menggantikan *call* yang dilakukan terhadap *function* dengan *macro*. Hasil dari penelitian ini menunjukkan optimasi yang dilakukan berhasil melakukan enkripsi dan dekripsi dengan waktu yang lebih cepat.

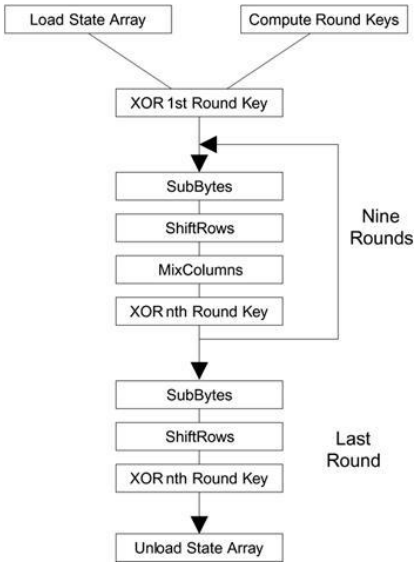
Penelitian lainnya yang melakukan analisis terhadap kinerja dari beberapa algoritma dalam ilmu kriptografi adalah penelitian [5] yang melakukan melakukan kombinasi terhadap algoritma RSA dan algoritma Diffie-Hellman untuk kemudian dibandingkan performanya dengan algoritma AES (Rijndael), DES, 3DES, RC2, dan Blowfish. Parameter pembanding dalam penelitian ini adalah CPU *time*, *memory*, dan *battery power*. Hasil dari penelitian ini menunjukkan bahwa kombinasi yang dibangun dari algoritma RSA dan algoritma Diffie-Hellman memiliki kinerja yang lebih unggul dibandingkan algoritma AES (rijndael), DES, 3DES, RC2, dan blowfish.

Penelitian lainnya yang juga melakukan perbandingan kinerja dari algoritma rijndael, serpent, dan twofish adalah penelitian [6] yang membandingkan *benchmark* (MB/s), waktu (ms), penggunaan *memory* (KB), dan persentase penggunaan *processor* (%) pada perangkat bergerak. Adapun perangkat bergerak yang diuji adalah perangkat bergerak dengan sistem operasi android (*smartphone*) dan windows (*tablet*). Hasil pengujian ini menunjukkan kinerja algoritma twofish lebih unggul dari pada algoritma lainnya berdasarkan uji *benchmark*. Sementara hasil pengujian waktu menunjukkan serpent lebih cepat daripada algoritma yang lain. Algoritma rijndael yang merupakan standar algoritma AES unggul pada pengujian persentase penggunaan CPU yang lebih rendah. Sedangkan hasil pengujian penggunaan *memory* menunjukkan kinerja algoritma twofish lebih unggul.

Berdasarkan studi literatur yang telah dilakukan, penelitian ini melanjutkan penelitian [6] yang melakukan analisis terhadap kinerja algoritma rijndael, twofish, dan serpent (tiga besar algoritma kandidat AES) yang diterapkan pada *smartphone*. Namun penelitian ini menambahkan beberapa *device* dan sistem operasi dalam percobaannya agar data uji menjadi lebih representatif. Penelitian ini penting dilakukan untuk menemukan algoritma kandidat AES mana yang memiliki kinerja terbaik dalam penerapannya pada *smartphone*, sehingga dapat membantu para pengembang aplikasi *smartphone* dalam merancang aplikasi dengan keamanan yang tinggi dan sumberdaya yang efisien.

2. TINJAUAN PUSTAKA

2.1 Rijndael



Gambar 1. Tahapan Rijndael [7]

Algoritma rijndael merupakan algoritma kunci simetri yang terpilih sebagai AES. Algoritma ini mendukung kriptografi dengan panjang kunci 128 bit sampai dengan 256 bit dengan step 32 bit. Ukuran blok dapat dipilih secara independen dan setiap blok dienkripsi sejumlah putaran tertentu.

AES atau rijndael dikelompokkan berdasarkan panjang kunci yang digunakan. Masing-masing kelompok memiliki jumlah putaran yang berbeda untuk setiap tahapannya.

Tabel 1. Kelompok AES [1]

Kelompok AES	**Panjang Kunci**	**Jumlah Putaran**
AES-128	128 bit	10
AES-192	192 bit	12
AES-256	256 bit	14

Tahapan-tahapan rijndael secara umum pada blok dan panjang kunci 128 bit adalah sebagai berikut.

1. *AddRoundKey.* Proses ini merupakan *initial round* yang melakukan XOR antara *plaintext* dengan *chipper key*.
2. Putaran sebanyak 10 kali meliputi proses berikut ini.

   a. *SubBytes*. Proses ini mensubstitusi *byte* dengan menggunakan tabel *S-box.* Adapun tabel *S-box* terlampir.

   b. *ShiftRows*. Proses ini melakukan pergeseran 1 kali ke kiri untuk baris 2 dari *state*, pergeseran 2 kali ke kiri untuk baris 3 dari *state*, dan pergeseran 3 kali ke kiri untuk baris 4 dari *state*.

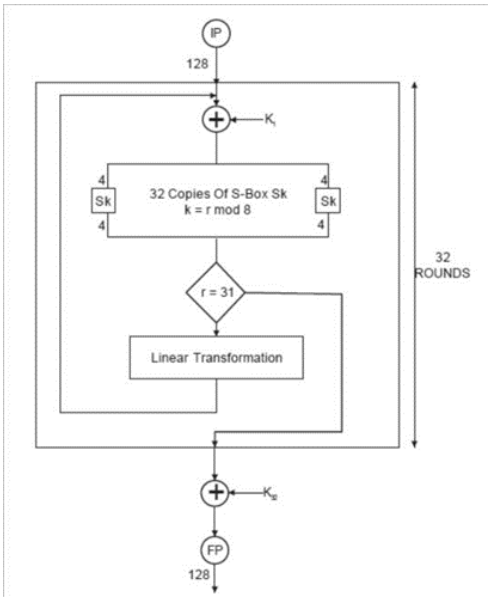
   c. *MixColumns*. Proses ini mengalikan kolom *state* satu persatu dengan *modular multiplication* terhadap matriks berikut.

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	

   d. *AddRoundKey*. Proses ini melakukan XOR antara *state* dengan *round key*.

3. Proses terakhir merupakan putaran terakhir atau *final round* meliputi *SubBytes*, *ShiftRows*, dan *AddRoundKey*

2.2 Serpent



Gambar 2. Tahapan Serpent [8]

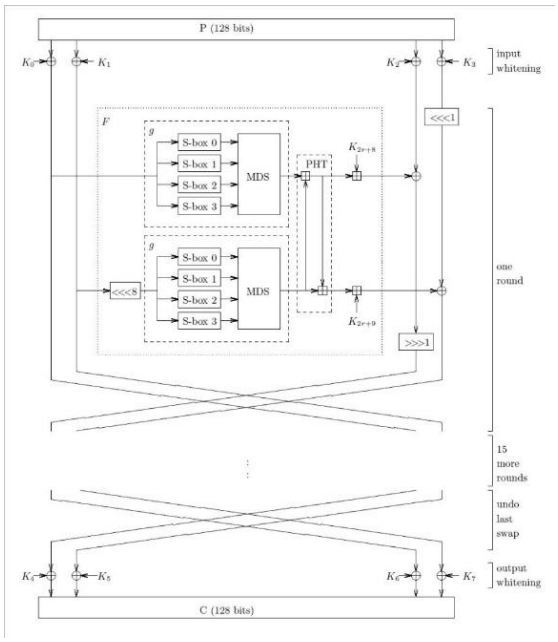
Algoritma Serpent adalah algoritma *chipper block* dengan 32 putaran jaringan *substitution permutation* (SP) yang beroperasi pada empat word 32 bit, yang berarti ukuran bloknya adalah 128 bit. Untuk komputasi internal, semua nilai direpresentasikan dalam *little-indian*, di mana word pertama adalah *least-significant word*, dan word terakhir adalah *most-significant word*. Secara eksternal, setiap blok dituliskan sebagai plain *hexadecimal* 128 bit .

Serpent mengenkripsi plainteks P 128 bit menjadi cipherteks C 128 bit dalam 32 putaran dengan kontrol dari 33 sub-kunci 128 bit K0,...,K32. Panjang kunci sama dengan kandidat AES lainnya, yaitu 128, 192,dan 256 bit.

Dari Gambar 2 di atas, tiga komponen utama dari algoritma Serpent adalah sebagai berikut.

1. Inisial Permutasi (IP)
2. 32 Putaran yang terdiri dari operasi pertukaran, kecuali di putaran terakhir dan pencampuran data pada transformasi linear.
3. Final Permutasi (FP)

2.3 Twofish



Gambar 3. Tahapan Twofish [9]

Twofish merupakan salah satu kandidat AES. Algoritma ini didesain oleh Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, dan Niels Ferguson dari Laboratorium Counterpane System. Berbeda dengan Rijndael yang dikelompokkan berdasarkan panjang kuncinya dan jumlah putaran untuk masing-masing panjang kunci, Twofish dilakukan dengan 16 putaran.

Twofish merupakan block cipher 128 bit yang menerima kunci dengan panjang yang fleksibel sampai 256 bit sama dengan Rijndael dan Serpent. Cipher pada algoritma Twofish ini menggunakan cukup banyak metode dalam implementasinya. Adapun metode-metode tersebut meliputi jaringan Feistel (Feistel *Network*), *S-Box*, Matriks MDS, transformasi Pseudo-Hadamard, *whitening*, dan *key schedule*.

Gambar 3 di atas adalah skema dari *block cipher* Twofish. Twofish menggunakan sebuah struktur seperti jaringan Feistel 16 putaran dengan tambahan *whitening* terhadap input dan output.

2.4 Penelitian Terdahulu

Penelitian [3] melakukan optimasi terhadap algoritma AES yang diterapkan pada *smartphone*. Optimasi pada penelitian ini dilakukan atas dasar penelitian sebelumnya yang dilakukan oleh [4] yang menyatakan bahwa algoritma AES belum bekerja baik pada perangkat dengan arsitektur *processor* ARM, sementara sebagian besar perangkat bergerak dengan sistem operasi android merupakan perangkat-perangkat dengan arsitektur *processor* ARM. Sehingga penelitian ini melakukan optimasi dalam bentuk mengurangi penggunaan 'for' dalam beberapa logika perulangan dalam AES, menggantikan *call* yang dilakukan terhadap *function* dengan *macro*. Hasil dari penelitian ini menunjukkan optimasi yang dilakukan berhasil melakukan enkripsi dan dekripsi dengan waktu yang lebih cepat.

Penelitian lainnya yang melakukan analisis terhadap kinerja dari beberapa algoritma dalam ilmu kriptografi adalah penelitian [5]. Penelitian ini melakukan kombinasi terhadap algoritma RSA dan algoritma Diffie-Hellman untuk kemudian dibandingkan performanya dengan algoritma AES (Rijndael), DES, 3DES, RC2, dan Blowfish. Parameter pembanding dalam penelitian ini adalah CPU *time*, *memory*, dan *battery power*. Hasil dari penelitian ini menunjukkan kombinasi algoritma RSA dan algoritma Diffie-Hellman memiliki kinerja yang lebih unggul dibandingkan AES (Rijndael), DES, 3DES, RC2, dan Blowfish.

Penelitian lainnya yang juga melakukan perbandingan kinerja dari algoritma rijndael, serpent, dan twofish adalah penelitian [6] yang membandingkan *benchmark* (MB/s), waktu (ms), penggunaan *memory* (KB), dan persentase penggunaan *processor* (%) pada perangkat bergerak. Adapun perangkat bergerak yang diuji adalah perangkat bergerak dengan sistem operasi android (*smartphone*) dan windows (*tablet*). Hasil pengujian ini menunjukkan kinerja algoritma twofish lebih unggul dari pada algoritma lainnya berdasarkan uji *benchmark*. Sementara hasil pengujian waktu menunjukkan serpent lebih cepat daripada algoritma yang lain. Algoritma rijndael yang merupakan standar algoritma AES unggul pada pengujian persentase penggunaan CPU yang lebih rendah. Sedangkan hasil pengujian penggunaan *memory* menunjukkan kinerja algoritma twofish lebih unggul.

Penelitian lainnya dengan judul "*SMS Security for Smartphone*" [10] mengkombinasikan algoritma AES dan RC4 untuk mengamankan pengiriman SMS pada *smartphone*. Adapun proses enkripsi dan dekripsi pada penelitian ini dilakukan menggunakan AES, namun proses pembangkitan kunci dilakukan dengan menggunakan RC4. Rancangan algoritma tersebut kemudian diuji pada beberapa *smartphone* dengan beragam ukuran RAM dan CPU. Hasilnya menunjukkan setiap proses enkripsi dan dekripsi berhasil dilakukan dengan waktu kurang dari satu detik.

Penelitian [11] melakukan perbandingan terhadap kinerja beberapa algoritma kunci simetri, seperti DES, 3DES, RC4, blowfish dan AES (rijndael) pada kasus keamanan SMS. Parameter pembanding kinerja dalam penelitian ini adalah *encryption time*, *decryption time*, *throughput*, *CPU process time*, dan *memory utilization*. Hasil dari penelitian ini menunjukkan algoritma AES dan blowfish memiliki kinerja terbaik, namun AES lebih unggul dari blowfish dalam penggunaan *memory* yang lebih efisien.

Terdapat penelitian lainnya yang mengusulkan pendekatan kriptografi dalam SMS pada *smartphone* dengan sistem operasi android. Penelitian tersebut mengkombinasikan ECDSA (Elliptic Curve Digital Signature Algorithm), ECDH (Elliptic Curve Diffie-Hellman) dan AES [12]. Penelitian ini menggunakan AES dan ECDH untuk melakukan proses enkripsi dan dekripsi, sementara ECDSA digunakan untuk proses otentikasi saat terjadi pertukaran kunci publik. Hasil dari penelitian ini menunjukkan pendekatan yang diusulkan berhasil melakukan

proses enkripsi dan dekripsi dengan tiga proses utama, yaitu pembangkitan kunci, pembangkitan *signature*, dan verifikasi *signature*.

3. METODE PENELITIAN

3.1 Skenario Percobaan

Penelitian ini melakukan serangkaian percobaan untuk menganalisis kinerja algoritma kandidat AES (rijndael, twofish, dan serpent). Parameter kinerja yang dianalisis dalam penelitian ini adalah sebagai berikut.

1. Waktu (ms)
2. Penggunaan *memory* (KB)
3. Persentase penggunaan CPU (%).

Device dan sistem operasi yang digunakan untuk melakukan percobaan dalam penelitian ini adalah sebagai berikut.

Tabel 2. Perangkat yang Digunakan Dalam Percobaan

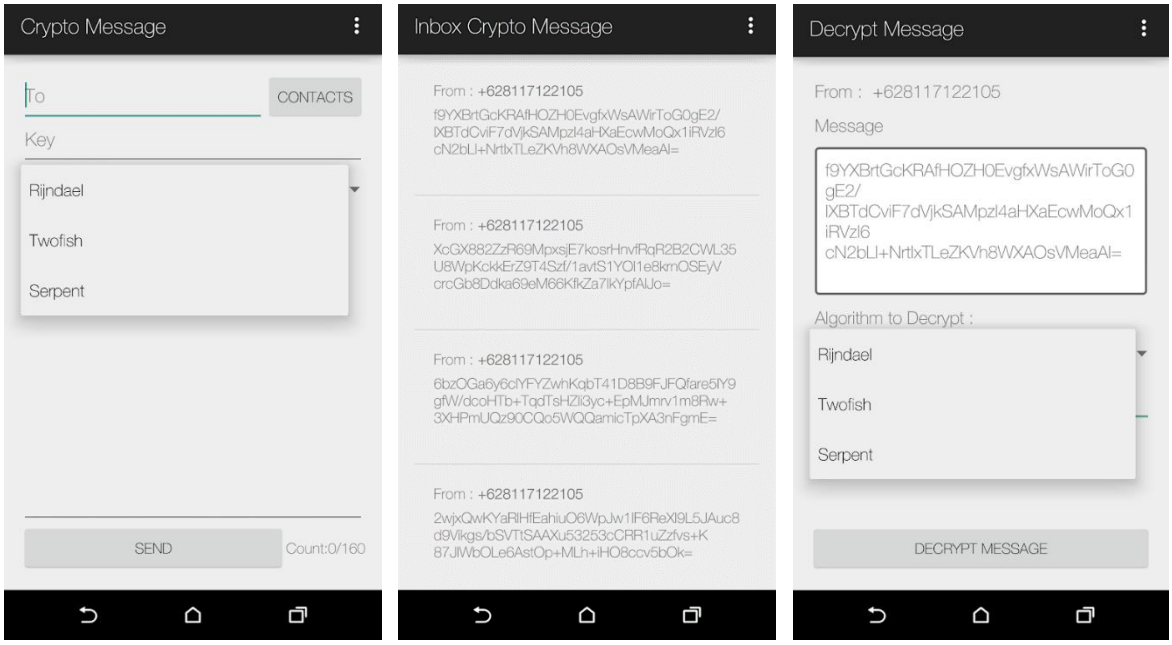
Devices Number	***Devices***	**Spesifikasi**	**Sistem Operasi**
1	Asus Zenfone 2	• CPU Quad-core 2.3 GHz • 4 GB RAM • *Internal memory* 32 GB	Android 6.0.1
2	Xiaomi Redmi 4A	• CPU Quad-core 1.4 GHz Cortex-A53 • 2 GB RAM • *Internal memory* 32 GB	Android 6.0.1
3	HTC One E8	• CPU Quad-core 2.5 GHz Krait 400 • 2 GB RAM • *Internal memory* 32 GB	Android 6.0.1
4	iPhone 6	• CPU Dual-core 1.4 GHz Typhoon (ARM v8-based) • 1 GB RAM DDR3 • *Internal memory* 64 GB	iOS 11.2.5
5	iPhone 6 Plus	• Dual-core 1.4 GHz Typhoon (ARM v8-based) • 1 GB RAM • *Internal memory* 64 GB	iOS 11.2.6
6	iPhone 7 Plus	• CPU Quad-core 2.34 GHz (2x Hurricane + 2x Zephyr) • 3 GB RAM • *Internal memory* 128 GB	iOS 11.2.6

Adapun percobaan yang dilakukan dalam penelitian ini menggunakan skenario percobaan sebagai berikut.

1. Percobaan 1-A : setiap *device* (6 *device*) melakukan 5 kali enkripsi, masing-masing *device* dan masing-masing proses enkripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfabet (90 percobaan).
2. Percobaan 1-B : setiap *device* (6 *device*) melakukan 5 kali dekripsi, masing-masing *device* dan masing-masing proses dekripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfabet (90 percobaan).

3. Percobaan 2-A : setiap *device* (6 *device*) melakukan 5 kali enkripsi, masing-masing *device* dan masing-masing proses enkripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter numerik (90 percobaan).
4. Percobaan 2-B : setiap *device* (6 *device*) melakukan 5 kali dekripsi, masing-masing *device* dan masing-masing proses dekripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter numerik (90 percobaan).
5. Percobaan 3-A : setiap *device* (6 *device*) melakukan 5 kali enkripsi, masing-masing *device* dan masing-masing proses enkripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfanumerik (90 percobaan).
6. Percobaan 3-B : setiap *device* (6 *device*) melakukan 5 kali dekripsi, masing-masing *device* dan masing-masing proses dekripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfanumerik (90 percobaan).
7. Percobaan 4-A : setiap *device* (6 *device*) melakukan 5 kali enkripsi, masing-masing *device* dan masing-masing proses enkripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfanumerik dan simbol (90 percobaan).
8. Percobaan 4-B : setiap *device* (6 *device*) melakukan 5 kali dekripsi, masing-masing *device* dan masing-masing proses dekripsi dilakukan dengan algoritma rijndael, twofish, dan serpent menggunakan kunci dengan karakter alfanumerik dan simbol (90 percobaan).

Untuk melakukan percobaan sesuai dengan skenario di atas, penelitian ini mengembangkan aplikasi yang dibangun secara *native* (android studio dan Xcode) pada *platform* android dan iOS yang diberi nama *Crypto Message*. Gambar 4 berikut ini menunjukkan beberapa *screenshot* dari aplikasi yang dibangun.



Gambar 4. *Screenshot* Aplikasi *Crypto Message*

3.2 Batasan Masalah

Penelitian ini membatasi permasalahan pada panjang kunci yang digunakan pada ketiga algoritma rijndael, serpent, dan twofish. Panjang kunci yang digunakan pada penelitian ini adalah 128 bit.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Percobaan Enkripsi

Percobaan enkripsi dilakukan pada percobaan 1-A, 2-A, 3-A, dan 4A sesuai dengan skenario percobaan yang telah dirancang. Hasil percobaan tersebut dirangkum pada Tabel 3 sebagai berikut.

Tabel 3. Hasil Percobaan Enkripsi

Percobaan	Algoritma Rijndael			Algoritma Twofish			Algoritma Serpent		
	Waktu (ms)	*Memory* (KB)	CPU (%)	Waktu (ms)	*Memory* (KB)	CPU (%)	Waktu (ms)	*Memory* (KB)	CPU (%)
1-A	0,024	0,032	0,060	0,068	0,032	0,180	0,030	0.064	0,020
2-A	0,032	0,032	0,030	0,032	0,032	0,060	0,023	0,064	0,020
3-A	0,036	0,032	0,010	0,053	0,032	0,070	0,025	0,064	0,030
4-A	0,028	0,032	0,010	0,033	0,032	0,100	0,021	0,064	0,030
Rata-Rata	**0,030**	**0,032**	**0,028**	**0,047**	**0,032**	**0,103**	**0,025**	**0,064**	**0,025**

Berdasarkan Tabel 3 di atas, dari total 360 kali percobaan enkripsi menggunakan 6 *device*, didapatkan hasil bahwa dalam proses enkripsi, algoritma yang paling efisien dalam penggunaan waktu adalah algoritma serpent (0,025 ms) dibandingkan dengan algoritma algoritma twofish (0,047 ms) dan algoritma rijndael (0,030 ms). Sementara dalam penggunaan *memory*, algoritma twofish dan rijndael lebih efisien (0,032 KB) dibandingkan dengan algoritma serpent (0,064 KB). Dalam penggunaan CPU, algoritma yang paling efisien adalah algoritma serpent yang menggunakan 0,025 % CPU dibandingkan algoritma twofish yang menggunakan 0,103 % CPU dan algoritma rijndael yang menggunakan 0,028 % CPU.

4.2 Hasil Percobaan Dekripsi

Percobaan dekripsi dilakukan pada percobaan 1-B, 2-B, 3-B, dan 4B sesuai dengan skenario percobaan yang telah dirancang. Hasil percobaan tersebut dirangkum pada Tabel 4 sebagai berikut.

Tabel 4. Hasil Percobaan Dekripsi

Percobaan	Algoritma Rijndael			Algoritma Twofish			Algoritma Serpent		
	Waktu (ms)	*Memory* (KB)	CPU (%)	Waktu (ms)	*Memory* (KB)	CPU (%)	Waktu (ms)	*Memory* (KB)	CPU (%)
1-B	0,028	0,032	0,050	0,036	0,032	0,390	0,021	0,064	0,030
2-B	0,028	0,032	0,060	0,031	0,032	0,140	0,023	0,064	0,070
3-B	0,033	0,032	0,030	0,041	0,032	0,090	0,026	0,064	0,030
4-B	0,030	0,032	0,030	0,031	0,032	0,180	0,018	0,064	0,020
Rata-Rata	**0,030**	**0,032**	**0,043**	**0,035**	**0,032**	**0,200**	**0,022**	**0,064**	**0,038**

Berdasarkan Tabel 4 di atas, dari total 360 kali percobaan dekripsi menggunakan 6 *device*, didapatkan hasil bahwa dalam proses dekripsi, algoritma yang paling efisien dalam penggunaan waktu adalah algoritma serpent (0,022 ms) dibandingkan dengan algoritma algoritma twofish (0,035 ms) dan algoritma rijndael (0,030 ms). Sementara dalam penggunaan *memory*, algoritma twofish dan rijndael lebih efisien (0,032 KB) dibandingkan dengan algoritma serpent (0,064 KB). Dalam penggunaan CPU, algoritma yang paling efisien adalah algoritma serpent yang menggunakan 0,038 % CPU dibandingkan algoritma twofish yang menggunakan 0,200 % CPU dan algoritma rijndael yang menggunakan 0,043 % CPU.

5. KESIMPULAN

Dari serangkaian percobaan yang telah dilakukan untuk menguji kinerja 3 besar algoritma kandidat AES (rijndael, twofish, dan serpent) pada *smartphone*, waktu tercepat untuk proses enkripsi (0,025 ms) dan dekripsi (0,022 ms) dilakukan oleh algoritma serpent. Sementara penggunaan *memory* yang paling efisien (0,032 KB) untuk proses enkripsi dan dekripsi dilakukan oleh algoritma rijndael dan twofish. Dan persentase penggunaan CPU yang paling efisien (0,025 % untuk proses enkripsi dan 0,038% untuk proses dekripsi) juga dilakukan oleh algoritma serpent.

Hasil percobaan dari penelitian ini menyimpulkan bahwa algoritma serpent memiliki kinerja terbaik dalam proses enkripsi dan dekripsi pada *smartphone* dibandingkan algoritma twofish dan rijndael. Artinya dalam penerapannya pada smartphone, kinerja algoritma rijndael belum tentu lebih baik dari algoritma kandidat AES lainnya, meskipun algoritma rijndael adalah algoritma AES yang terpilih.

6. SARAN

Setelah melakukan serangkaian percobaan dalam penelitian ini, beberapa hal yang dapat dilakukan untuk penelitian selanjutnya adalah sebagai berikut.

1. Menambahkan beberapa *device* lagi sebagai percobaan tambahan dengan spesifikasi yang setara satu sama lain, khususnya dalam CPU dan RAM.
2. Pengujian kinerja akan lebih baik lagi jika percobaan penelitian yang serupa juga dilakukan pada algoritma AES dengan panjang kunci 192 dan 256 bit.
3. Penelitian selanjutnya juga dapat menguji kinerja algoritma kandidat AES dari sudut pandang pengembangan aplikasi yang digunakan dalam percobaan, misalnya aplikasi yang dikembangkan dengan *native android* dan *ios*, dan *hybrid*.

DAFTAR PUSTAKA

[1] R. Munir. 2006, *Kriptografi*, Informatika, Bandung.

[2] D. J Bernstein. 2012, "*AES : The Advanced Encryption Standard*," [Online], Available: https://competitions.cr.yp.to/aes.html, [Accessed: 02-Mar-2018].

[3] S. Liang, J. Liu, R. Zhang, and C. Wang. 2010, "*A Modified AES Algorithm for The Platform of Smartphone*," *Proc. - Int. Conf. Comput. Asp. Soc. Networks, CASoN'10*, pp. 749–752,

[4] H. Xiangyi and L. Tong. 2006, "*The Research of Dynamic Symmetric Cipher Algorithm*," *Netw. Secur. Technol. Appl.*, Col. 3.

[5] B. K. Mandal, D. Bhattacharyya, and S. K. Bandyopadhyay. 2013, "*Designing and Performance Analysis of A Proposed Symmetric Cryptography Algorithm*," *Proc. - 2013 Int. Conf. Commun. Syst. Netw. Technol. CSNT 2013*, pp. 453–461.

[6] A. O. Montoya, M. A. Muñoz, and S. T. Kofuji. 2013, "*Performance Analysis of Encryption Algorithms on Mobile Devices*," *47th Int. Carnahan Conf. Secur. Technol.*, pp. 1–6.

[7] Etutorials.org. 2010, "*Steps in the AES Encryption Process*," [Online]. Available: http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/Appendixes/Appendix+A.+Overview+of+the+AES+Block+Cipher/Steps+in+the+AES+Encryption+Process/. [Accessed: 22-Feb-2018].

[8] R. Anderson. 2001, "*SERPENT - A Candidate Block Cipher for The Advanced Encryption Standard*," *University of Cambridge*, [Online]. Available: http://www.cl.cam.ac.uk/~rja14/serpent.html. [Accessed: 23-Feb-2018].

[9] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, and C. Hall. 1998, "*Twofish : A 128-Bit Block Cipher*," *NIST AES Propos.*, vol. 15, no. 1, pp. 1–27,

[10] A. M. Sagheer, A. A. Abdulhameed, and M. A. Abduljabbar. 2013, "*SMS Security for Smartphone*," *Proc. - 2013 6th Int. Conf. Dev. eSystems Eng. DeSE 2013*, pp. 281–285.

[11] S. N. Karale, K. Pendke, and P. Dahiwale. 2015, "The Survey of Various Techniques & Algorithms for SMS Security," *ICIIECS 2015 - 2015 IEEE Int. Conf. Innov. Information, Embed. Commun. Syst.*

[12] M. H. Azaim, D. W. Sudiharto, and E. M. Jadied. 2016, "Design and Implementation of Encrypted SMS on Android Smartphone Combining ECDSA - ECDH and AES," *Proc. - APMediaCast 2016*, pp. 18–23.