IMPLEMENTASI KRIPTOGRAFI DENGAN MENGGUNAKAN METODE RC4 DAN AES-256 UNTUK MENGAMANKAN *FILE* DOKUMEN PADA PT VARNION *TECHNOLOGY* SEMESTA

Handrian Saputra Djong[1*], Siswanto Siswanto[2]

[1,2]Fakultas Teknologi Informasi, Teknik Informatika, Universitas Budi Luhur, Jakarta, Indonesia

Email: [1*]handriansptra@gmail.com, [2]siswanto@budiluhur.ac.id
(* : corresponding author)

Abstrak- PT. Varnion *Technology* Semesta merupakan perusahaan yang berkecimpung dalam bidang jasa teknologi informasi dan komunikasi. PT. Varnion mempunyai banyak *file* dokumen yang bersifat penting serta rahasia, yang tidak boleh diketahui oleh pihak yang tidak berkepentingan dikarenakan berhubungan dengan informasi pelanggan maupun informasi rahasia perusahaan sendiri seperti dokumen informasi pelanggan, dokumen rencana proyek, dokumen laporan teknis dan lainnya. Permasalahan yang dihadapi adalah *file* dokumen dapat dimengerti dan diketahui oleh pihak yang tidak berkepentingan dikarenakan perusahaan masih belum memiliki sistem pengamanan pada *file* dokumen dan upaya untuk mengatasi permasalahan tersebut yaitu dengan mengimplementasikan mekanisme keamanan seperti algoritma kriptografi yang bisa menyandikan isi *file* dokumen sehingga membuat *file* dokumen menjadi tidak bisa dipahami dan tidak bisa diketahui dengan mudah isinya oleh pihak yang tidak berkepentingan. Penelitian ini akan bertujuan membuat sebuah aplikasi pengamanan *file* dokumen yang mampu mengimplementasikan algoritma kriptografi dengan menggunakan metode RC4 dan AES-256 untuk mengamankan *file* dokumen agar *file* dokumen tidak bisa dipahami atau tidak bisa diketahui isinya oleh pihak yang tidak berkepentingan. Hasil dari penelitian yaitu aplikasi pengamanan *file* dokumen yang telah dibuat, mampu mengimplementasikan algoritma kriptografi dengan metode RC4 dan AES-256 pada *file* dokumen yang berekstensi *.txt, .pdf, .doc, .docx, .xls* dan juga *.xlsx* dengan ukuran maksimal 3MB. Berdasarkan percobaan enkripsi dan dekripsi terhadap sembilan sampel *file* dokumen, didapatkan hasil bahwa rata-rata ukuran *file* dokumen awal yaitu 571,17491 *KiloBytes* kemudian rata-rata ukuran *file* hasil enkripsi yaitu 571,18403 *KiloBytes* tidak berbeda jauh dengan rata-rata ukuran *file* hasil dekripsi yaitu 571,17491 *KiloBytes* (persentase perbedaan ukuran sekitar 0,000394531 % hingga 0,00075%) namun ukuran *file* dokumen yang sudah didekripsi serupa dengan ukuran *file* dokumen asli. Kontribusi penelitian ini adalah mengimplementasikan kriptografi dengan mengkombinasikan dua metode enkripsi kunci simetri yaitu RC4 dan AES-256 untuk mengamankan *file* dokumen.

Kata Kunci: Kriptografi, RC4, AES-256, Enkripsi, Dekripsi

IMPLEMENTATION OF CRYPTOGRAPHY USING RC4 AND AES-256 METHODS TO SECURE DOCUMENT FILES AT PT VARNION TECHNOLOGY SEMESTA

Abstract- *PT. Varnion Technology Semesta is a company engaged in the field of information and communication technology services. PT. Varnion has many important and confidential document files, which should not be known by unauthorized parties because they relate to customer information and the company's own confidential information such as customer information documents, project plan documents, technical report documents and others. The problem faced is that document files can be understood and known by unauthorized parties because the company still does not have a security system for document files and efforts to overcome these problems are by implementing security mechanisms such as cryptographic algorithms that can encode the contents of document files so as to make document files become cannot be understood and cannot be easily identified by unauthorized parties. This study aims to create a document file security application that is able to implement cryptographic algorithms using the RC4 and AES-256 methods to secure document files so that document files cannot be understood or cannot be known by unauthorized parties. The results of the research are document file security applications that have been created, able to implement cryptographic algorithms with the RC4 and AES-256 methods on document files with the extension .txt, .pdf, .doc, .docx, .xls and also .xlsx with a maximum size of 3MB. Based on the encryption and decryption experiments on nine sample document files, it was found that the average initial document file size was 571.17491 KiloBytes then the average encrypted file size was 571,18403 KiloBytes not much different from the average decrypted file size was 571,17491 KiloBytes (percentage difference in size is around 0.000394531 % to 0.00075%) but the file size of the decrypted document is similar to the file size of the original document. The contribution of this research is to implement cryptography by combining two symmetric key encryption methods, namely RC4 and AES-256 to secure document files.*

Keywords*: Cryptography, RC4, AES-256, Encryption, Decrypt*

1. PENDAHULUAN

Pesatnya perkembangan teknologi informasi dan juga komunikasi pada masa sekarang telah menunjukkan dampaknya bagi semua kalangan terutama dalam perihal memudahkan berbagai pekerjaan manusia seperti penggunaan dokumen digital yang memudahkan dalam mengelola data dan informasi, penggunaan jaringan internet yang memudahkan komunikasi ataupun pertukaran informasi dari jarak yang sangat jauh dan masih banyak lagi. Perkembangan tersebut di sisi lain juga menyebabkan munculnya ancaman kejahatan komputer atau kejahatan dunia maya yaitu kebocoran informasi, modifikasi, penyadapan hingga penyalahgunaan informasi oleh pihak yang tidak memiliki otoritas atas suatu data atau informasi yang bisa merugikan siapapun, baik individu, instansi hingga perusahaan sehingga membuat aspek keamanan menjadi faktor penting dan lebih diperhatikan. Dokumen adalah informasi rahasia yang sangat berharga dan penting bagi otoritas individu atau perusahaan manapun dan hanya boleh diketahui oleh pihak berkepentingan saja.

PT. Varnion *Technology* Semesta merupakan perusahaan yang berkecimpung dalam bidang jasa teknologi informasi dan komunikasi. PT. Varnion mempunyai banyak *file* dokumen bersifat penting serta rahasia, yang tidak boleh diketahui oleh pihak yang tidak berkepentingan dikarenakan berhubungan dengan informasi pelanggan maupun informasi rahasia perusahaan sendiri seperti dokumen informasi pelanggan, dokumen rencana proyek, dokumen laporan teknis dan lain sebagainya. Permasalahan yang dihadapi adalah *file* dokumen dapat diketahui dan dimengerti oleh pihak yang tidak berkepentingan dikarenakan perusahaan masih belum memiliki sistem pengamanan untuk *file* dokumen dan upaya untuk mengatasi permasalahan yang ada yaitu dengan mengimplementasikan mekanisme keamanan seperti algoritma kriptografi yang bisa menyandikan isi *file* dokumen sehingga membuat *file* dokumen menjadi tidak bisa dipahami dan tidak bisa diketahui oleh pihak yang tidak berkepentingan. Beberapa penelitian sebelumnya, menggunakan algoritma kriptografi dengan tujuan untuk mengamankan pesan teks [1], [2], [3], mengamankan citra digital [4], mengamankan *file* dokumen [5], [6], [7], [8] dan juga mengamankan basis data [9], [10], [11]. Beberapa penelitian sebelumnya, yang menggunakan kombinasi lebih dari satu metode kriptografi yaitu telah dilakukan oleh [9], [11] dengan metode RC4 dan *Base64* untuk mengamankan basis data dan juga telah dilakukan oleh [3] dengan menggunakan metode *Caesar Cipher* dan RC4 untuk mengamankan pesan teks, maka dari itu penelitian ini akan bertujuan membangun aplikasi pengamanan *file* dokumen yang mampu mengimplementasikan algoritma kriptografi dengan menggunakan metode RC4 dan AES-256 untuk mengamankan *file* dokumen agar isi *file* dokumen tidak bisa dipahami atau tidak bisa dibaca oleh pihak yang tidak berkepentingan. Kontribusi penelitian ini adalah mengimplementasikan kriptografi dengan mengkombinasikan dua metode enkripsi kunci simetri yaitu RC4 dan AES-256 untuk mengamankan *file* dokumen.

Kriptografi terdiri dari dua kata yang merupakan bahasa yunani yaitu "*crypto*" dan "*graphia*". "*Crypto*" yang mempunyai arti yaitu "*secret* (rahasia)" dan "*graphia*" yang mempunyai arti yaitu "*writing* (tulisan)". Pengertian kriptografi itu sendiri yaitu "ilmu dan seni yang menjaga keamanan pesan ketika pesan dikirimkan dari suatu tempat ke tempat lain" [10]. "Pesan atau data asli yang belum dienkripsi dikenal dengan sebutan *plaintext*. Pesan yang sudah mengalami perubahan atau penyandian menjadi suatu sandi atau kode dikenal sebagai *ciphertext*. Proses pengubahan *plaintext* menjadi *ciphertext* dikenal dengan sebutan enkripsi. Kebalikan dari proses enkripsi dikenal dengan sebutan dekripsi" [9]. "Tujuan dari kriptografi yang juga merupakan aspek keamanan yang mendasar antara lain kerahasiaan (*confidentiality*), autentikasi (*authentication*), integritas data atau keutuhan data (*data integrity*) dan tak terbantahkan (*non-repudiation*)" [12].

Metode RC4 diciptakan oleh *Ron Rivest* pada tahun 1987. "Kehandalan RC4 yaitu kecepatan dan kesederhanaan algoritmanya dalam menangani banyak aplikasi, membuatnya mudah untuk diimplementasikan dalam perangkat keras dan perangkat lunak" [3]. Algoritma metode RC4 memiliki dua tahapan utama, yaitu: *key setup* (penjadwalan kunci) dan *stream generation* (pembuatan *pseudorandom key*). Tahapan *key setup* memiliki 3 proses yang harus dilalui yaitu inisialisasi *S-Box* RC4, menyalin kunci ke dalam *key byte array* dan terakhir permutasi pada *S-Box* RC4. Nilai *keystream* yang dihasilkan dalam tahapan *stream generation* akan dioperasikan *XOR* dengan *plaintext* ataupun *ciphertext* untuk menghasilkan *ciphertext* ataupun untuk menghasilkan *plaintext* [8].

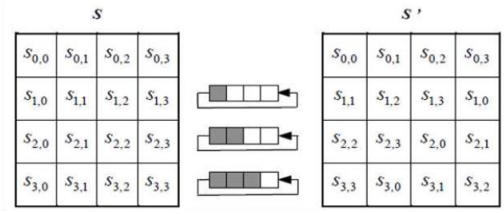
Algoritma *Advanced Encryption Standard* (AES) adalah salah satu algoritma kriptografi jenis *block chiper* dan juga jenis algoritma kunci simetri yang memakai kunci yang sama pada proses enkripsi maupun dekripsi. Penyandian AES menggunakan proses berulang yang disebut ronde. Banyak ronde pada algoritma AES bergantung pada panjang kunci yang digunakan. Setiap ronde membutuhkan kunci ronde dan masukan berupa hasil ronde sebelumnya. Kunci ronde dibangkitkan berdasarkan kunci yang diberikan. Enkripsi dan dekripsi data dalam algoritma AES menggunakan kunci dengan panjang yang bervariasi yaitu 128 *bit*, 192 *bit*, dan juga 256 *bit* [6]. "NIST (*National Institute of Standard and Technology*) mengumumkan standar baru enkripsi yang menggantikan DES (*Data Encryption Standard*) melalui lomba seleksi algoritma baru yang cukup ketat dengan beberapa algoritma lainnya pada November 2001. *Vincent Rijmen* dan *Joan Daemen* telah memenangkan lomba tersebut lalu algoritma buatan mereka dinamakan AES (*Advanced Encryption Standard*)". "Alasan utama dari

terpilihnya AES dikarenakan algoritma AES memenuhi kriteria yaitu memiliki keseimbangan antara aspek keamanan dan fleksibilitas untuk berbagai *platform* perangkat lunak maupun perangkat keras" [7].

"Empat transformasi *bytes* utama berikut, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey* akan digunakan dalam enkripsi dengan metode AES. Tahap enkripsi dimulai dari menyalin masukan *plaintext* ke dalam *state* lalu menjalankan transformasi *byte AddRoundKey* pertama terhadap *state*. Tahap berikut adalah *state* dari hasil *AddRoundKey* pertama akan ditransformasikan *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara berulang sebanyak *Nr*. Tahapan berulang tersebut dalam metode AES disebut *round function*. *Round* terakhir memiliki perbedaan dengan *round* lainnya yaitu tidak melakukan transformasi *MixColumns* terhadap *state* dalam *round* terakhir" [6].

AddRoundKey merupakan transformasi yang menggabungkan kunci *round* dan *state* dengan operasi *XOR*. Transformasi *AddRoundKey* tetap sama pada proses enkripsi dan dekripsi, hanya saja terdapat perbedaan pada urutan *round key* yang dioperasikan *XOR* dengan *state*. *Round key* pertama dioperasikan *XOR* dengan *state* di tahap *initial round* enkripsi dan *round key* terakhir dioperasikan *XOR* dengan *state* pada tahap *AddRoundKey* terakhir dalam proses enkripsi. Berbeda urutan pada proses dekripsi, dimana *round key* terakhir dioperasikan *XOR* dengan *state* di tahap *initial round* dan *round key* pertama dioperasikan *XOR* dengan *state* pada tahap *AddRoundKey* terakhir.

"Transformasi *SubBytes* merupakan transformasi *byte* yang dilakukan dengan cara mensubstitusikan atau mengganti setiap *byte* dari *state* dengan *byte* yang berada pada tabel *S-Box* AES" [10]. Setiap nilai *byte* dalam *array state*, misalkan *S* [*r*, *c*] = *xy* dan *xy* adalah *digit* heksadesimal dari nilai *S*[*r*, *c*]. Nilai yang menggantikan *S*[*r*, *c*] dinyatakan dengan *S*' [*r*, *c*], yaitu bilangan heksadesimal yang berada pada tabel *S-Box* AES [5]. Transformasi *Shiftrows* adalah transformasi yang melakukan pergeseran nilai *byte* pada tiga baris terakhir dari *array state*. Banyak pergeseran bergantung pada nilai baris r. Ketentuannya yaitu jika baris r sama dengan 1 maka pergeseran dilakukan sebanyak satu *byte*, jika baris r sama dengan 2 maka pergeseran dilakukan sebanyak dua *byte*, dan jika baris r sama dengan 3 maka pergeseran dilakukan sebanyak tiga *byte*. Baris r sama dengan 0 maka tidak dilakukan pergeseran sama sekali [7]. Gambar 1 mendeskripsikan pergeseran yang dilakukan ke arah kiri pada transformasi *ShiftRows*.



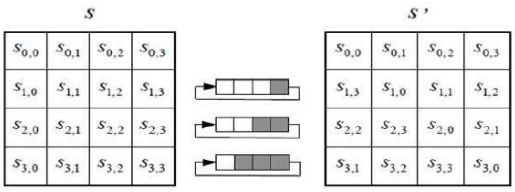
Gambar 1. *ShiftRows* [5]

"Transformasi *MixColumns* merupakan operasi perkalian antara matrik *MixColumns Rijndael* dengan setiap kolom *array state*" [7]. Transformasi *MixColumns* dapat dituliskan dalam perkalian matrik seperti (1). Perkalian matrik *MixColumns* AES akan menghasilkan empat nilai baru untuk setiap kolom *array state*. Persamaan yang menghasilkan empat nilai baru untuk setiap *kolom array state* dirumuskan seperti (2).

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (1)$$

$$\begin{aligned} S'_{0,c} &= (02 \bullet S_{0,c}) \text{ XOR } (03 \bullet S_{1,c}) \text{ XOR } S_{2,c} \text{ XOR } S_{3,c} \\ S'_{1,c} &= S_{0,c} \text{ XOR } (02 \bullet S_{1,c}) \text{ XOR } (03 \bullet S_{2,c}) \text{ XOR } S_{3,c} \\ S'_{2,c} &= S_{0,c} \text{ XOR } S_{1,c} \text{ XOR } (02 \bullet S_{2,c}) \text{ XOR } (03 \bullet S_{3,c}) \\ S'_{3,c} &= (03 \bullet S_{0,c}) \text{ XOR } S_{1,c} \text{ XOR } S_{2,c} \text{ XOR } (02 \bullet S_{3,c}) \end{aligned} \quad (2)$$

Transformasi *invers* yaitu, *InvSubBytes*, *InvShiftRows*, *InvMixColumns* digunakan dalam proses dekripsi AES. Transformasi *AddRoundKey* tidak mempunyai transformasi *invers* dikarenakan transformasi *AddRoundKey* bersifat *self-invers* dengan syarat harus menggunakan kunci yang sama [1]. *InvShiftRows* adalah transformasi *invers* dari *ShiftRows*. Transformasi *InvShiftRows* terhadap sebuah *state* merupakan operasi pergeseran *byte* ke arah kanan pada setiap baris *state*. Banyaknya pergeseran yang dialami setiap baris *state* ditentukan berdasarkan nilai indeks baris *state*. Gambar 2 mengilustrasikan pergeseran yang dimaksudkan pada transformasi *InvShiftRows*.



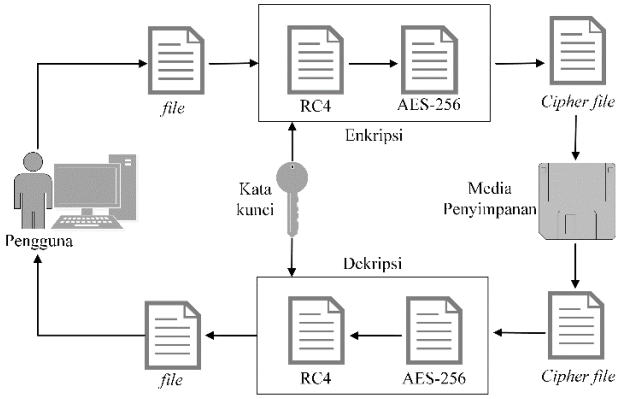
Gambar 2. *InvShiftRows* [5]

Kebalikan dari transformasi *SubBytes* adalah transformasi *InvSubBytes*. Transformasi *InvSubBytes* melakukan pergantian nilai pada *state* dengan nilai baru yang dipetakan dengan menggunakan tabel *Inverse S-Box* AES [5]. "Transformasi *InvMixColumns* yaitu transformasi yang mengalikan setiap nilai dalam kolom *state* dengan matrik perkalian AES" [5]. Perkalian matrik yang terjadi pada transformasi *InvMixColumns* dapat dituliskan seperti (3).

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (3)$$

2. METODE PENELITIAN

2.1 Skema Sistem



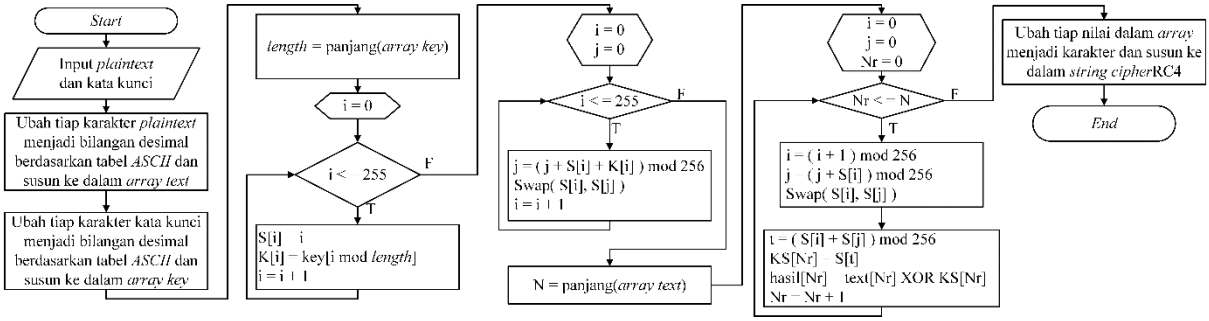
Gambar 3. Skema Sistem

Gambar 3 menerangkan secara ringkas mengenai skema sistem. Persyaratan utama aplikasi pengamanan *file* dokumen adalah harus mampu melakukan dua kemampuan proses atau fungsi utama yaitu proses enkripsi yang bisa menyandikan isi *file* dokumen dengan maksud untuk mengamankan *file* dokumen dan proses dekripsi untuk mengubah kembali isi *file* dokumen agar bisa dimengerti oleh pengguna. Sistem memerlukan dua masukan dari pengguna untuk menjalankan enkripsi yaitu *file* dokumen yang ingin dienkripsi dan kata kunci yang ditentukan pengguna. *File* dokumen melewati dua tahap enkripsi dalam sistem yaitu enkripsi RC4 terlebih dahulu lalu kemudian enkripsi AES-256. Sistem juga memerlukan dua masukan dari pengguna untuk menjalankan dekripsi yaitu *file* dokumen yang ingin didekripsi dan kata kunci yang ditentukan pengguna. *File* dokumen melewati dua tahap dekripsi dalam sistem yaitu dekripsi AES-256 terlebih dahulu lalu kemudian dekripsi RC4. Syarat *file* dokumen bisa terdekripsi menjadi *file* dokumen yang bisa dibaca/dilihat yaitu masukan kunci untuk dekripsi harus identik (sama) dengan kunci pada proses enkripsi sebelumnya dikarenakan kedua metode ini termasuk dalam jenis kunci simetri.

2.2 Enkripsi

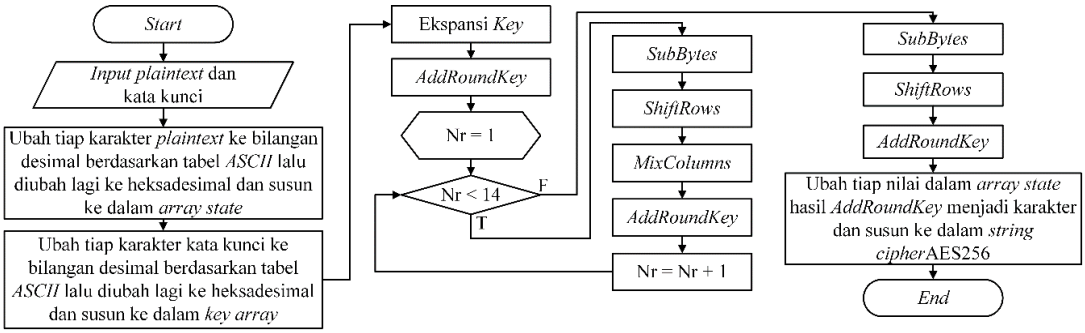
Proses enkripsi digambarkan melalui *flowchart* enkripsi RC4 terlebih dahulu kemudian dilanjutkan dengan *flowchart* enkripsi AES dengan panjang kunci 256 *bit*. Gambar 4 memperlihatkan *flowchart* proses enkripsi RC4 yang dimulai dengan memasukkan *plaintext* dan kata kunci lalu masuk dalam tahapan *key setup* (*Key Scheduling Algorithm*) mengubah tiap karakter *plaintext* menjadi bilangan desimal berdasarkan tabel *ASCII* dan susun ke dalam *array text* kemudian juga mengubah tiap karakter kata kunci menjadi bilangan desimal berdasarkan tabel *ASCII* dan susun ke dalam *key array*. Tahap berikut yaitu melakukan perulangan sebanyak 256 kali untuk

inisialisasi *array* K dari *key array* dan juga inisialisasi *S-Box* lalu kemudian dilakukan perulangan sebanyak 256 kali dalam tahapan permutasi *S-Box* RC4. Tahapan *Pseudo Random Generation Algorithm* melakukan perulangan sebanyak panjang *array text* untuk menghasilkan nilai *pseudorandom key atau keystream* yaitu KS[] dan kemudian nilai tersebut dioperasikan XOR dengan nilai dari *array text* untuk menghasilkan nilai *array* hasil sebagai *ciphertext* dari enkripsi RC4. Tahap terakhir yaitu mengubah setiap nilai dari *array* hasil menjadi karakter dan lalu susun menjadi *string cipher*RC4.



Gambar 4. *Flowchart* Enkripsi RC4

Plaintext pada *flowchart* proses enkripsi AES dengan panjang kunci 256 *bit* adalah hasil dari enkripsi RC4. Gambar 5 mendeskripsikan secara ringkas mengenai *flowchart* proses enkripsi AES dengan panjang kunci 256 *bit*. *Flowchart* dimulai dengan mengubah tiap karakter *plaintext* ke bilangan desimal lalu diubah lagi ke dalam heksadesimal, dilanjutkan dengan menyusunnya ke dalam *array state* dan langkah selanjutnya yaitu mengubah tiap karakter kata kunci ke bilangan desimal lalu diubah lagi ke dalam heksadesimal, dilanjutkan dengan menyusunnya ke dalam *key array*. Tahap Ekspansi *Key* yaitu meliputi inisialisasi kunci dari *key array* dan melakukan ekspansi yang menghasilkan sebanyak 15 *RoundKey* (dari *RoundKey*-0 hingga *RoundKey*-14) yang akan digunakan secara berurutan pada masing-masing transformasi *AddRoundKey* dalam proses enkripsi AES. Transformasi *AddRoundKey* pertama adalah melakukan operasi XOR antara *RoundKey*-0 dengan *array state*. Tahap berikut adalah pengoperasian *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* terhadap *array state* secara berulang hingga 13 kali. Tahap *round* terakhir (*Round* ke 14) melaksanakan operasi transformasi terhadap *array state* yang dimulai dari *SubBytes* hingga *AddRoundKey* tanpa diikuti transformasi *MixColumns*. Proses terakhir yaitu mengubah setiap nilai dari *array state* hasil *AddRoundKey* terakhir menjadi karakter lalu susun menjadi *string cipher*AES256.

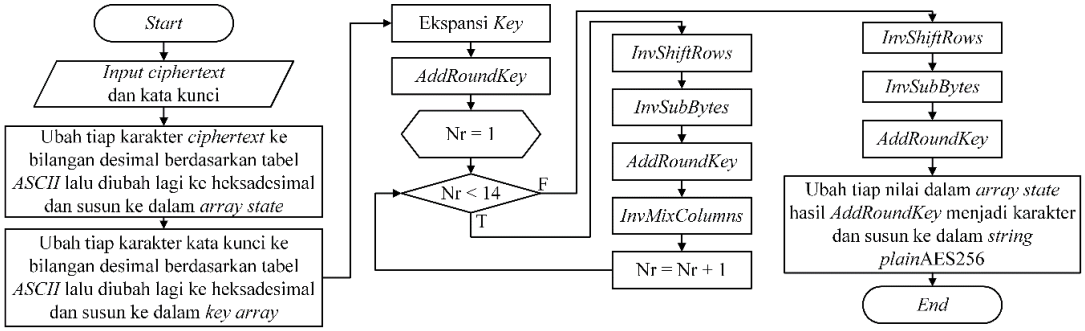


Gambar 5. *Flowchart* Enkripsi AES-256

2.3 Dekripsi

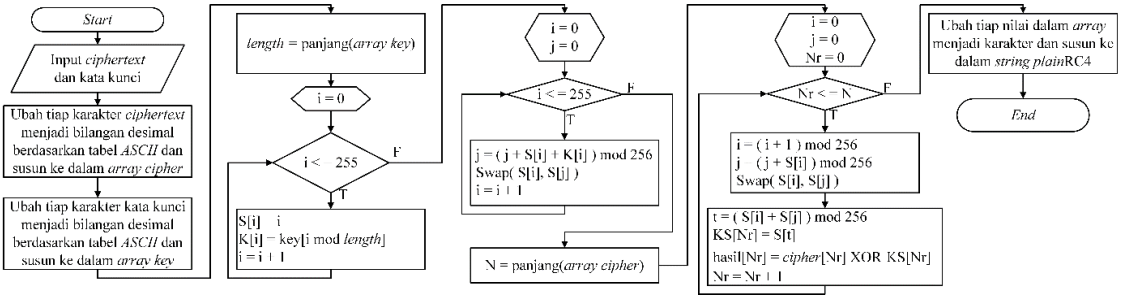
Proses dekripsi digambarkan melalui *flowchart* dekripsi AES-256 terlebih dahulu kemudian dilanjutkan dengan *flowchart* dekripsi RC4. Dekripsi AES dengan panjang kunci 256 *bit* dipaparkan secara sederhana melalui gambar 6. Dekripsi RC4 dimulai dengan mengubah tiap karakter *cipher* ke bilangan desimal lalu diubah lagi ke dalam heksadesimal, dilanjutkan dengan menyusunnya ke dalam *array state* dan langkah selanjutnya yaitu mengubah tiap karakter kata kunci ke bilangan desimal lalu diubah lagi ke dalam heksadesimal, dilanjutkan dengan menyusunnya ke dalam *key array*. Tahap Ekspansi kunci yaitu meliputi inisialisasi kunci dari *key array* dan operasi ekspansi yang menghasilkan 15 kunci ronde (dari kunci ronde-0 hingga kunci ronde-14) yang akan digunakan secara berurutan mulai dari kunci ronde-14 hingga kunci ronde-0 pada masing-masing operasi *AddRoundKey* dalam proses dekripsi AES. Transformasi *AddRoundKey* pertama dalam dekripsi dilakukan dengan

mengoperasikan XOR antara *RoundKey*-14 dengan *array state*. Tahap berikut adalah pengoperasian *InvShiftRows*, *InvSubBytes*, *AddRoundKey* dan *InvMixColumns* terhadap *array state* secara berulang hingga 13 kali. Tahap *round* terakhir (*Round* ke 14) menjalankan transformasi *SubBytes* hingga *AddRoundKey* terhadap *array state* tanpa diikuti pengoperasian transformasi *InvMixColumns*. Proses terakhir yaitu mengubah setiap nilai dari *array state* hasil *AddRoundKey* terakhir menjadi karakter lalu susun menjadi *string plain*AES256.



Gambar 6. *Flowchart* Dekripsi AES-256

Dekripsi RC4 pada sistem dipaparkan secara ringkas pada gambar 7. *Ciphertext* pada *flowchart* proses dekripsi RC4 adalah hasil dari dekripsi AES dengan panjang kunci 256 *bit*. *Flowchart* proses dekripsi RC4 hampir sama dengan *flowchart* enkripsi RC4, yang membedakan adalah nilai *keystream* (hasil dari tahap *Pseudo Random Generation Algorithm*) akan dioperasikan XOR dengan nilai *array cipher* untuk menghasilkan nilai *array* hasil sebagai *plaintext* dari dekripsi RC4. Tahap terakhir yaitu mengubah setiap nilai dari *array* hasil menjadi karakter dan lalu susun menjadi *string plain*RC4.



Gambar 7. *Flowchart* Dekripsi RC4

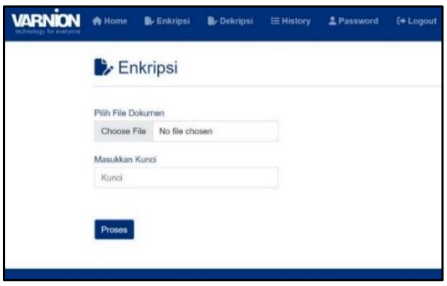
3. HASIL DAN PEMBAHASAN

3.1 Tampilan Layar

Pengguna harus melalui proses autentikasi terlebih dahulu pada halaman *form login* supaya bisa mengakses aplikasi. Proses autentikasi yang dimaksudkan yaitu pengguna harus memasukkan nip dan *password* pada *textfield* yang tersedia lalu kemudian menekan tombol masuk dan jika berhasil maka dapat langsung menggunakan aplikasi. Proses autentikasi dikatakan berhasil jika nip dan *password* yang dimasukkan cocok dengan yang terdaftar dalam basis data. Gambar 8 merupakan tangkapan layar dari *form login.*

Gambar 8. Tangkapan Layar *Form Login*

Tangkapan layar halaman enkripsi ditunjukkan gambar 9. Halaman enkripsi memiliki tombol pilih *file* yang berfungsi mengarahkan pengguna untuk mencari dan memilih *file* dokumen yang ingin dienkripsikan ketika ditekan. Tersedia juga *textfield* kunci yang berfungsi sebagai kotak isian untuk memasukkan kata kunci yang diperlukan untuk enkripsi. Tombol proses pada halaman enkripsi berfungsi untuk memulai enkripsi terhadap *file* dokumen yang sudah dipilih ketika pengguna mengklik / menekannya dengan ketentuan pengguna sudah memilih *file* dokumen yang ingin dienkripsi dan juga sudah memasukkan kata kunci.

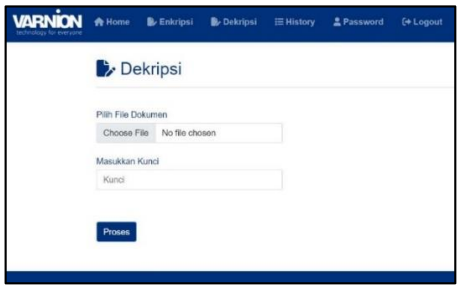


Gambar 9. Tangkapan Layar dari Halaman Enkripsi

Pengguna akan melihat halaman hasil enkripsi sesudah proses enkripsi selesai dilakukan. Menyimpan *file* dokumen hasil enkripsi dilakukan dengan cara menekan / mengklik tombol simpan pada halaman hasil enkripsi. Tombol kembali berfungsi untuk kembali ke halaman enkripsi. Gambar 10 memperlihatkan tangkapan layar dari halaman hasil enkripsi.

Gambar 10. Tangkapan Layar dari Halaman Hasil Enkripsi

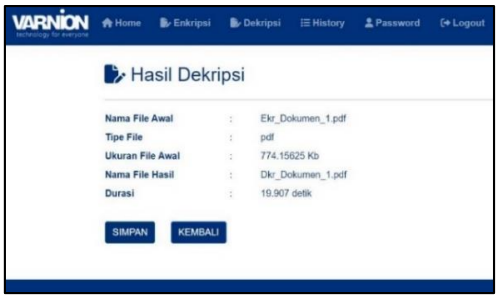
Gambar 11 menunjukkan tampilan layar halaman dekripsi. Tombol pilih *file* yang berada di halaman dekripsi berfungsi mengarahkan pengguna untuk mencari dan memilih *file* dokumen yang ingin didekripsi ketika ditekan. Halaman dekripsi juga menyediakan *textfield* kunci yang berfungsi sebagai kotak isian untuk memasukkan kunci yang dibutuhkan dalam proses dekripsi dan harus menekan atau mengklik tombol proses untuk menjalankan dekripsi. Syarat supaya *file* dokumen bisa terdekripsi sempurna menjadi *file* dokumen yang dapat dibaca/dilihat yaitu kunci pada dekripsi harus identik (sama) dengan kunci pada proses enkripsi sebelumnya.



Gambar 11. Tampilan Halaman Dekripsi

Halaman hasil dekripsi akan ditampilkan setelah proses dekripsi selesai dilakukan. Menyimpan *file* dokumen hasil dekripsi bisa dilakukan dengan cara mengklik / menekan tombol simpan pada halaman hasil dekripsi. Tombol kembali digunakan untuk berpindah ke halaman dekripsi. Gambar 12 memperlihatkan tampilan halaman hasil dekripsi.

e-ISBN 999-999-99999-9-9
Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)
Jakarta – Indonesia, dd mmmm yyyy



Gambar 12. Tampilan Halaman Hasil Dekripsi

3.2 Pengujian

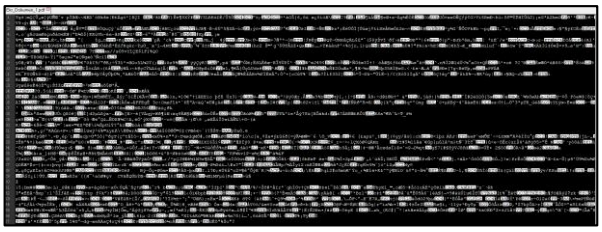
Metode *blackbox* digunakan dalam pengujian sistem untuk mengetahui sistem aplikasi dapat menjalankan kemampuan yang sesuai dengan harapan peneliti. Pengujian yang dimaksudkan terdiri dari beberapa skenario untuk menguji fungsi autentikasi, enkripsi dan dekripsi beserta fungsi pendukung lainnya. Hasil evaluasi *blackbox* terhadap aplikasi bisa dilihat pada tabel 1. Hasil tersebut menyatakan bahwa aplikasi sudah memenuhi harapan peneliti.

Tabel 1. Hasil Pengujian *Blackbox* Aplikasi

Aktivitas Pengujian	Skenario masukan pengujian	Keluaran yang diharapkan	Hasil Pengujian	Akurasi
Login aplikasi	Memasukkan nip dan *password* pada *form login* kemudian tekan tombol masuk	*Login* berhasil akan langsung masuk dalam halaman menu utama. *Login* gagal akan muncul pesan *invalid* nip or *password*.	Diterima	100%
Enkripsi Dokumen	Memilih *file* dokumen dan memasukkan kata kunci lalu menekan tombol proses	Menampilkan halaman hasil enkripsi apabila dokumen sudah dipilih dan kata kunci sudah mencapai 10 karakter Muncul pesan *file* belum dipilih apabila *user* belum memilih dokumen Muncul pesan kunci kurang dari 10 karakter apabila kata kunci yang dimasukkan belum mencapai 10 karakter	Diterima	100%
Dekripsi Dokumen	Memilih *file* dokumen dan memasukkan kata kunci lalu menekan tombol proses	Menampilkan halaman hasil dekripsi apabila dokumen sudah dipilih dan kata kunci sudah mencapai 10 karakter Muncul pesan *file* belum dipilih apabila *user* belum memilih dokumen Muncul pesan kunci kurang dari 10 karakter apabila kata kunci yang dimasukkan belum mencapai 10 karakter	Diterima	100%
Mengubah *password* pengguna	Memasukkan *password* lama, *password* baru dan konfirmasi *password* baru	Menampilkan pesan *password* berhasil diubah jika *password* lama terdaftar dalam basis data serta isian konfirmasi *password* baru dan *password* baru sama Menampilkan pesan *password* lama salah jika *password* lama yang dimasukkan tidak terdaftar dalam basis data	Diterima	100%

		Menampilkan pesan konfirmasi *password* baru gagal jika konfirmasi *password* baru dan *password* baru diisi berbeda		
Melihat *History* aktivitas sistem	Memilih menu *history*	Sistem menampilkan *history* dari setiap aktivitas pada sistem	Diterima	100%

Isi dari *file* dokumen menjadi tidak bisa dipahami atau tidak bisa dibaca setelah melalui proses enkripsi dikarenakan telah disandikan. Gambar 13 memperlihatkan penampakan isi *file* dokumen yang sudah terenkripsi. Isi dari *file* dokumen dapat dikembalikan semula sama persis dengan *file* dokumen asli setelah melalui proses dekripsi dengan ketentuan yaitu penggunaan kata kunci dalam proses dekripsi harus identik (sama) seperti kata kunci yang dimasukkan pada proses enkripsi. Gambar 14 memperlihatkan penampakan isi *file* dokumen yang sudah melalui proses dekripsi. Gambar 13 dan gambar 14 menunjukkan keberhasilan aplikasi dalam menjalankan enkripsi dan dekripsi tehadap *file* dokumen dengan menggunakan kedua metode yang diusulkan.



Gambar 13. Tampilan Hasil Enkripsi

Gambar 14. Tampilan Hasil Dekripsi

Tabel 2. Hasil Evaluasi Pengujian Proses Enkripsi dan Dekripsi

Dokumen	Ukuran Dokumen (KB)			Selisih Ukuran	Durasi (Detik)	
	Asli	Enkripsi	Dekripsi		Enkripsi	Dekripsi
Dokumen_1.pdf	774,14648	774,15625	774,14648	0,00977	21,522	19,907
Dokumen_2.pdf	742,65332	742,65625	742,65332	0,00293	16,7365	17,557
Dokumen_3.pdf	1.985,24512	1985,25	1985,24512	0,00488	55,2598	54,6649
Dokumen_4.pdf	34,25391	34,26563	34,25391	0,01172	1,66444	1,00017
Dokumen_5.docx	756,29102	756,29688	756,29102	0,00586	23,7339	24,0636
Dokumen_6.docx	683,7373	683,75	683,7373	0,0127	24,945	22,3682
Dokumen_7.xls	35	35,01563	35	0,01563	0,785857	0,869392
Dokumen_8.xlsx	124,39844	124,40625	124,39844	0,00781	4,1886	4,32445
Dokumen_9.txt	4,84863	4,85938	4,84863	0,01075	0,37437	0,20585
Rata-rata	571,17491	571,18403	571,17491	0,00917	16,5789	16,1067

Tabel 2 merupakan hasil evaluasi pengujian proses enkripsi dan dekripsi yang mengimplementasikan metode RC4 (*Rivest Code* 4) dan AES 256 (*Advanced Encryption Standard* 256) terhadap 9 sampel *file* dokumen. Berdasarkan pengujian yang dilakukan, maka didapatkan hasil bahwa rata-rata ukuran *file* dokumen awal yaitu 571,17491 *KiloBytes* kemudian rata-rata ukuran *file* hasil enkripsi yaitu 571,18403 *KiloBytes* tidak berbeda jauh dengan rata-rata ukuran *file* hasil dekripsi yaitu 571,17491 *KiloBytes*. Selisih terbesar ukuran *file* dokumen sesudah enkripsi sebesar 0,01563 KB (dalam persentase 0,00075%) dialami Dokumen_7 dan selisih terkecil ukuran *file*

dokumen sesudah enkripsi sebesar 0,00293 KB (dalam persentase 0,000394531%) dialami Dokumen_2 namun ukuran *file* dokumen yang sudah didekripsi serupa dengan ukuran *file* dokumen asli. Dokumen_3 yang memiliki ukuran dokumen sebesar 1.985,24512 KB, mengalami durasi enkripsi atau dekripsi paling lama yaitu 55,2598 detik untuk enkripsi dan 54,6649 detik untuk durasi dekripsi. Dokumen_3 yang memiliki ukuran dokumen sebesar 1.985,24512 KB, mengalami durasi enkripsi atau dekripsi terlama yaitu 55,2598 detik untuk enkripsi dan 54,6649 detik untuk durasi dekripsi. Dokumen_7 yang memiliki ukuran dokumen sebesar 35 KB, mengalami durasi enkripsi atau dekripsi tercepat yaitu 0,785857 detik untuk enkripsi dan 0,869392 detik untuk durasi dekripsi. Berdasarkan data di atas, bisa disimpulkan bahwa ukuran *file* berbanding lurus dengan durasi atau lama waktu proses enkripsi ataupun proses dekripsi pada *file* dokumen. Semakin besar ukuran *file* dokumen asli, semakin lama juga rentang proses enkripsi maupun dekripsi bisa selesai.

4. KESIMPULAN

Aplikasi pengamanan *file* dokumen yang sudah dibuat, telah berhasil mengimplementasikan algoritma kriptografi dengan mengkombinasikan metode RC4 dan AES 256 untuk mengamankan *file* dokumen yang berekstensi *.doc, .docx, .xls, .xlsx, .txt* dan juga *.pdf* dengan ukuran maksimal 3 MB. Hal ini ditunjukkan dengan *file* dari hasil pengujian enkripsi yang tidak bisa dipahami dan tidak bisa dibaca oleh manusia sehingga bisa membantu dalam mengamankan *file* dokumen dari pihak yang tidak berhak atau tidak berkepentingan. *File* dokumen dari hasil pengujian proses dekripsi yang memperlihatkan hasil yang selalu serupa (identik) dengan *file* dokumen awal yang di-*input*-kan sehingga pihak yang berwenang dapat membaca dan mengerti isi dari *file*. Berdasarkan hasil analisa pengujian, ukuran *file* berbanding lurus dengan lama waktu proses enkripsi atau dekripsi pada *file* dokumen. Semakin besar ukuran *file* dokumen asli, semakin lama juga proses enkripsi maupun dekripsi bisa selesai.

Sistem aplikasi pengamanan *file* dokumen yang sudah dibuat, diharapkan ke depannya ditambahkan fitur untuk mengatur level *user* dan juga manajemen *user*. Aplikasi pengamanan *file* dokumen juga diharapkan mampu mengenkripsi *file* selain *.doc, .docx, .xls, .xlsx, .txt, .pdf* seperti *file* dengan ekstensi *.ppt, .jpg, .png, .wav, .mp3* dan jenis ekstensi lainnya. Penelitian berikutnya diharapkan bisa melakukan modifikasi terhadap algoritma kriptografi yang sudah ada sebelumnya atau mengkombinasikan dengan berbagai metode algoritma kriptografi lainnya dan juga dapat menambahkan algoritma steganografi untuk lebih meningkatkan keamanan pada *file* dokumen.

DAFTAR PUSTAKA

[1] D. Darwis, R. Prabowo, and N. Hotimah, "Kombinasi Gifshuffle, Enkripsi AES dan Kompresi Data Huffman untuk Meningkatkan Keamanan Data," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 5, no. 4, pp. 389-394, 2018.
[2] D. R. Saragi, J. M. Gultom, J. A. Tampubolon, and I. Gunawan, "Pengamanan Data File Teks (Word) Menggunakan Algoritma RC4," *Jurnal Sistem Komputer dan Informatika (JSON)*, vol. 1, no. 2, pp. 114-119, 2020.
[3] A. D. Wiranata and R. T. Aldisa, "Aplikasi Steganografi Menggunakan Least Significant Bit (LSB) dengan Enkripsi Caesar Chipper dan Rivest Code 4 (RC4) Menggunakan Bahasa Pemrograman JAVA," *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 5, no. 3, pp. 277-281, 2021.
[4] T. Zebua and Ndruru Eferoni, "Pengamanan Citra Digital Berdasarkan Modifikasi Algoritma RC4," *Jurnal Teknologi Informasi dan Ilmu Komputer* , vol. 4, pp. 275-282, 2017.
[5] D. Nurnaningsih and A. A. Permana, "Rancangan Aplikasi Pengamanan Data dengan Algoritma Advanced Encyption Standard (AES)," *Jurnal Teknik Informatika*, vol. 11, no. 2, pp. 177-186, 2018.
[6] B. E. Widodo and A. S. Purnomo, "Implementasi Advanced Encryption Standard pada Enkripsi dan Dekripsi Dokumen Rahasia Ditintelkam Polda DIY," *Jurnal Teknik Informatika (Jutif)*, vol. 1, no. 2, pp. 69-77, 2020.
[7] S. Setti, I. Gunawan, B. E. Damanik, S. Sumarno, and I. O. Kirana, "Implementasi Algoritma Advanced Encryption Standard dalam Pengamanan Data Penjualan Ramayana Department Store," *JURIKOM (Jurnal Riset Komputer)*, vol. 7, no. 1, pp. 182-193, 2020.
[8] F. S. Febriyani and A. Arfriandi, "Implementasi Algoritma RC4 pada Sistem Pengamanan Dokumen Digital Soal Ujian," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 6, no. 3, pp. 171-177, 2021.
[9] I. Afrianto and N. Taliasih, "Sistem Keamanan Basis Data Klien P.T. Infokes Menggunakan Kriptografi Kombinasi RC4 Dan Base64," *Jurnal Nasional Teknologi dan Sistem Informasi*, vol. 6, no. 1, pp. 9-18, 2020.
[10] L. Mustika, "Implementasi Algoritma AES Untuk Pengamanan Login Dan Data Customer Pada E-Commerce Berbasis Web," *JURIKOM (Jurnal Riset Komputer)*, vol. 7, no. 1, pp. 148-155, 2020.
[11] M. Rizky Royani and A. Wibowo, "Web Service Implementation in Logistics Company uses JSON Web Token and RC4 Cryptography Algorithm," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 3, pp. 591-600, 2020.
[12] J. Simarmata, Sriadhi, and R. Rohim, *KRIPTOGRAFI Teknik Keamanan Data & Informasi*. Yogyakarta: Andi, 2019.