IMPLEMENTASI ALGORITMA AUTOKEY CIPHER DAN AES-128 PADA ENKRIPSI FILE

Candra Irawan[1], De Rosal Ignatius Moses Setiadi[2]
[1,2]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
e-mail: [1]candra.irawan@dsn.dinus.ac.id, [2]moses@dsn.dinus.ac.id

ABSTRAK

Kriptografi merupakan suatu bidang studi populer saat ini yang sangat dibutuhkan untuk menjaga keaslian serta keamanan data dari pihak yang tidak berhak untuk mengakses data tersebut. Dalam era digital ini, berbagi informasi seperti file digital meningkat secara signifikan. Untuk melindungi data file dibangunlah aplikasi VertoCrypt berbasis desktop yang dapat mengenkripsi berbagai file teks, gambar, musik, maupun video dengan memanfatkan algoritma Autokey cipher digabung dengan Advanced Encryption Standard (AES-128), kemudian untuk menambah tingkat keamanan, kunci yang digunakan untuk enkripsi dan dekripsi file memanfaatkan fungsi hash SHA-512. Dengan dimanfaatkannya algoritma-algoritma tersebut, algoritma tidak akan mudah dipecahkan oleh kriptanalis. File yang sudah terenkripsi akan membuat pihak ketiga tidak dapat langsung membaca file karena bit-bit file sudah teracak, dengan demikian data yang tersimpan di dunia maya baik itu komputer pribadi, cloud, atau data yang dibagikan melalui suatu aplikasi akan terlindungi. Dengan metode yang diusulkan didapatkan bahwa waktu eksekusi metode yang digabungkan memakan waktu lebih lama dari implementasi salah satu metode saja. Hasilnya akan memperlama kriptanalis dalam meretas file yang telah terenkripsi.

***Kata Kunci**: Kriptografi, Autokey Cipher, AES, SHA, Enkripsi File, Dekripsi File*

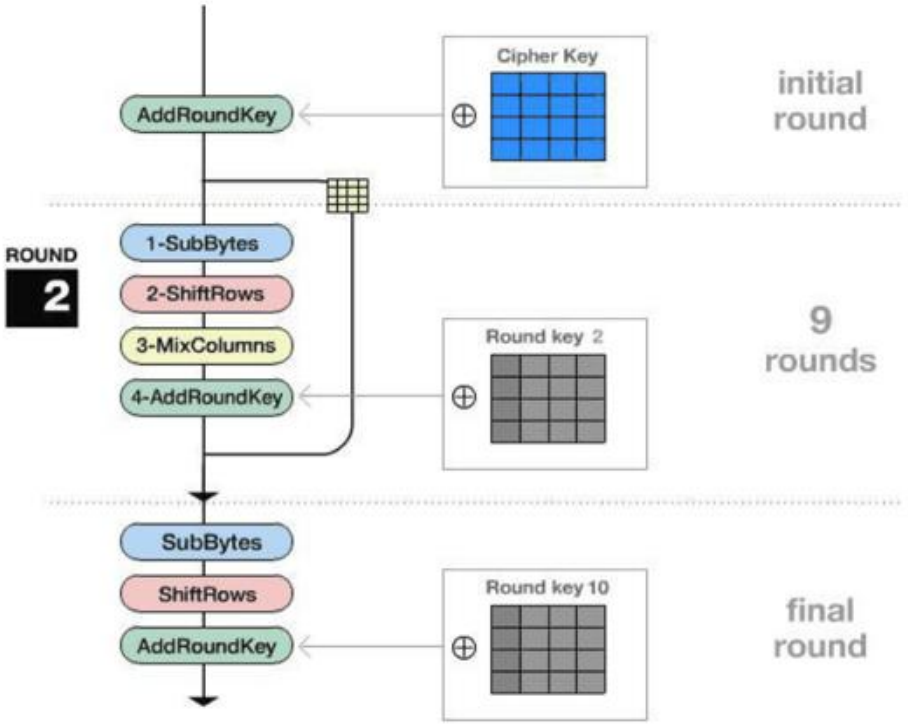
1. PENDAHULUAN

Dari sekian banyak manfaat yang didapat dari perkembangan teknologi saat ini, tentu saja diikuti pula oleh dampak negatif yang tidak kalah besar [1]. Era digital sekarang merupakan era dimana berbagi informasi digital meningkat secara signifikan [2]. Lalu lintas data sangat marak saat ini. Setiap individu pasti memiliki berbagai macam data dari file dokumen, gambar, musik, video, dll yang tidak hanya bersifat publik namun juga privasi. Keamanan menjadi hal yang sangat penting bagi semua pengguna namun tidak dapat dipungkiri bahwa jaminan keamanan masih lemah hingga saat ini. Pada 6 bulan awal tahun 2018 saja sudah ada 4,5 miliar data dicuri berdasarkan perusahaan Gemalto, jumlah ini meningkat sebanyak 113% dari tahun lalu, akan tetapi kasus pencurian data mengalami penurunan dari 1.162 kasus pada tahun lalu menjadi 945 kasus di tahun ini [3]. Media sosial menjadi target terbesar para pihak ketiga sebesar 56,11%, diikuti dengan data-data instansi pemerintah sebesar 26,62%. Meskipun demikian hanya sebagian kecil dari data-data tersebut terlindungi dengan enkripsi yaitu hanya sekitar 4% saja. Cara untuk melindungi data yang paling populer saat ini salah satunya adalah menggunakan teknik kriptografi. Menggunakan satu algoritma dinilai tidak efektif untuk perlindungan dengan tingkat yang tinggi [4]. Dengan demikian peneliti mengusulkan algoritma lebih dari satu yaitu kriptografi dengan algoritma *Autokey cipher* digabung dengan AES-128 sedangkan untuk menyembunyikan dan verifikasi kata kunci, penulis memanfaatkan fungsi hash yaitu algoritma SHA-512. *Autokey cipher* merupakan salah satu algoritma klasik yang mudah dan termasuk kategori *polyalphabetic substitution cipher*. Algoritma klasik *polyalphabetic substitusi* memang mudah diretas tetapi tidak semudah *monoalphabetic substitution* yang tiap karakter pasti disubstitusi oleh karakter yang sama, berbeda dengan *polyalphabetic substitution* yang tiap karakter disubstitusi oleh karakter yang berbeda-beda ketika dilakukan enkripsi [5]. Algoritma AES *(Advanced Encryption Standard)* merupakan salah satu algoritma modern yang menggunakan kunci simetri dan termasuk dalam block cipher. AES adalah algoritma enkripsi yang paling sering digunakan [6]. Dengan menggabungkan *Autokey cipher* dan AES-128, peneliti berasumsi bahwa algoritma yang diusulkan akan menghasilkan algoritma dengan tingkat keamanan yang lebih baik dari AES asli seperti penelitian yang dilakukan oleh Suchita Tayde pada tahun 2015. Suchita Tayde mengimplementasikan algoritma AES pada Android untuk mengenkripsi dan mendekripsi file [7]. Terakhir untuk verifikasi kunci, penulis menggunakan *Secure Hash Algorithm* (SHA-512) yang akan menambah perlindungan karena kunci akan dituliskan pada file dan akan berubah menjadi fungsi hash. Fungsi hash merupakan enkripsi yang irreversible[6]. SHA-512 berasal dari keluarga SHA-2 yang lebih aman dari SHA-1 [8]. Dengan demikian kriptanalis akan kesulitan untuk mengetahui kunci sebenarnya yang digunakan untuk mengenkripsi sebuah file.

2. TINJAUAN PUSTAKA

2.1 Advanced Encryption Standard (AES)

AES adalah salah satu algoritma kriptografi denga model operasi blok cipher. Dengan penggunaan kunci simteris, maka proses enkripsi dan dekripsinya menggunakan kunci yang sama seperti implementasi AES pada aplikasi 7-Zip. Algoritma Rijndael dikenal dengan AES dan popular pada tahun 2006. Operasi AES menggunakan sistem permutasi dan substitusi (P-Box dan S-Box) bukan dengan jaringan *Fiestel* sebagaimana blok cipher pada umunya.



Gambar 1. Skema AES

AES merupakan pengembangan dari algoritma *Data Encryption Standard* (DES), dengan jumlah kunci bervariasi yaitu 128, 192 dan 256 bit. AES-128 menggunakan 10 round, AES-192 sebanyak 12 round, dan AES-256 sebanyak 14 *round*. Garis besar Algoritma Rijndael yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*):

1. AddRoundKey: melakukan XOR antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak Nr – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *SubBytes*: substitusi byte dengan menggunakan tabel substitusi (S-box).
   b. *ShiftRows:* pergeseran baris-baris array state secarawrapping.
   c. *MixColumns*: mengacak data di masing-masing kolomarray state.
   d. *AddRoundKey*: melakukan XOR antara state sekaranground key.
3. Final round: proses untuk putaran terakhir:
   a. *SubBytes*
   b. *ShiftRows*
   c. *AddRoundKey*

2.2 Autokey

Cipher *Autokey* adalah salah satu contoh variasi dari vigenere cipher. Secara umum, istilah *autokey* mengacu pada cipher mana pun dimana kuncinya didasarkan pada teks asli. Dalam bentuknya yang paling sederhana, ini pertama kali dijelaskan oleh Girolamo Cardano, dan terdiri dari menggunakan plaintext itu sendiri sebagai *keystream*. Namun, karena tidak ada kunci yang terlibat dalam sistem ini, itu menderita cacat besar yang sama dengan Atbash dan Trithemius Ciphers: jika Anda tahu itu telah digunakan, itu mudah untuk memecahkan kode. Namun, versi yang paling terkenal dari Autokey Cipher, dijelaskan oleh Blaise de Vigenère pada tahun 1586 (versi yang kemudian salah mengartikan Vigenère Cipher). Sandi ini memasukkan kata kunci dalam pembuatan *keystream*, serta teks teks asli.

Enkripsi pada *Autokey Cipher* sangat mirip dengan Vigenère Cipher, kecuali dalam penciptaan *keystream*. *Keystream* dibuat dengan memulai dengan kata kunci atau frase kunci, dan kemudian menambahkan ke akhir ini teks teks itu sendiri. *Tabula Recta* digunakan untuk menemukan huruf keystream di bagian atas, dan huruf *plaintext* di sebelah kiri, dan menggunakan huruf *crossover* sebagai huruf ciphertext.

Untuk mendekripsi ciphertext menggunakan *Autokey Cipher*, dimulai dengan menemukan huruf pertama dari kunci di atas, menemukan huruf ciphertext di kolom itu, dan mengambil huruf plaintext di paling kiri dari baris ini.

3. METODE PENELITIAN

Metode yang diusulkan peneliti untuk mengenkripsi file adalah *Autokey cipher* yang merupakan algoritma klasik dalam kriptografi digabung dengan AES-128bit yang merupakan algoritma modern dalam kriptografi, jadi peneliti mengusulkan penggabungan metode klasik dan modern, kemudian untuk enkripsi kunci, digunakanlah fungi *hash* SHA-512.

3.1 Prosedur Enkripsi

Untuk prosedur enkripsi file dijelaskan sebagai berikut:

1. Pengguna memilih file yang akan dienkripsi.
2. Sistem akan mengecek apakah file sudah dalam keadaan terenkripsi dan ekstensi file mendukung, jika belum terenkripsi dan ekstensi file mendukung maka file akan diproses.
3. Pengguna memasukkan kunci minimal 8 karakter.
4. File dan kunci dibaca ke dalam byte.
5. SHA-512 diimplementasikan ke kunci asli untuk menghasilkan kunci hash 512 bit atau 64 *byte* kemudian diubah ke *string hexa* dan dikembalikan ke bentuk *byte* sepanjang 128 *byte*.
6. Untuk mendapatkan cipherfile pertama, plainfile yaitu byte dari file akan dienkripsi menggunakan metode *Autokey cipher* dengan bantuan byte kunci asli bukan kunci hash.
7. Selanjutnya cipherfile pertama dienkripsi lagi dengan menggunakan metode AES-128 dengan bantuan *byte* kunci asli bukan kunci *hash* dan menghasilkan cipherfile kedua.
8. Merekonstruksi file dengan menuliskan kunci hash 128 byte ke dalam file terenkripsi di awal bit untuk verifikasi kunci, kemudian dilanjutkan oleh byte dari cipherfile kedua.
9. File terenkripsi atau encrypted file terbentuk dengan ekstensi.enkrip.

3.2. Prosedur Enkripsi

Sedangkan untuk prosedur dekripsi file dijelaskan sebagai berikut :

1. Pengguna memilih file yang akan didekripsi.
2. Sistem akan mengecek apakah file sudah dalam keadaan terenkripsi atau belum, jika sudah maka file akan diproses.
3. Pengguna memasukkan kunci.
4. File dan kunci dibaca ke dalam *byte*.
5. SHA-512 diimplementasikan ke kunci asli untuk menghasilkan kunci hash 512 bit atau 64 byte kemudian diubah ke *string hexa.*
6. Kunci *hash* dicek ke dalam file terenkripsi apakah sesuai dengan *hash* kunci yang terletak di awal 128 byte untuk verifikasi kunci.
7. Untuk mendapatkan cipherfile pertama, cipherfile terakhir yaitu *byte* dari file akan didekripsi menggunakan metode AES-128 dengan bantuan byte kunci asli bukan kunci *hash*.
8. Selanjutnya cipherfile pertama didekripsi lagi dengan menggunakan metode *Autokey cipher* dengan bantuan *byte* kunci asli bukan kunci hash dan menghasilkan plainfile yang sesungguhnya.
9. File asli atau plainfile terbentuk dengan ekstensi tanpa.enkrip

4. HASIL DAN PEMBAHASAN

Eksperimen menguji beberapa ekstensi file yang didukung oleh aplikasi VertoCrypt yang dibangun yaitu berekstensi .txt, .doc, .pdf, .jpg, .png, .mp3, .mp4, .mkv. Eksperimen diuji dengan menggunakan password “13anisadila”. Proses enkripsi dan dekripsi menggunakan *Autokey Cipher* yang dilanjutkan AES 128 menghabiskan waktu lebih lama dari pada menggunakan satu metode saja seperti *Autokey Cipher* saja atau AES 128 saja. Untuk lebih jelasnya bisa dilihat pada Tabel 1 berikut.

Tabel 1. Hasil Enkripsi pada beberapa ekstensi file berbeda

Original File			Autokey Cipher		AES 128		Autokey + AES 128	
Nama File	Ukuran Plainfile	Ukuran Cipherffile	ENC	DEC	ENC	DEC	ENC	DEC
CorelDRAW Graphics Suite X7.txt	258 *bytes* (258 *bytes*)	386 *bytes* (386 *bytes*)	0.02289 7012	0.0125 28546	0.2597 52645	0.1776 59793	0.2561 65912	0.15899 4415
PANITIA HUT PROKLAMASI KEMERDEKAAN.docx	17,3 KB (17.771 *bytes*)	17,4 KB (17.899 *bytes*)	0.00372 2727	0.0033 03626	0.1366 41378	0.0895 37449	0.1331 99621	0.09396 1529
JadwalUjian_Proposal_A11.2015.08721.pdf	67,3 KB (68.937 *bytes*)	67,4 KB (69.065 *bytes*)	0.01502 134	0.0057 18159	0.1635 5406	0.1257 7555	0.1632 1322	0.11656 0466
ferrari-488-gtb.jpg	121 KB (124.485 *bytes*)	121 KB (124.613 *bytes*)	0.00494 2395	0.0047 05475	0.1507 28722	0.1075 79736	0.1476 25666	0.10300 8972
key_PNG1180.png	22,3 KB (22.926 *bytes*)	22,5 KB (23.054 *bytes*)	0.00427 6966	0.0054 41467	0.1304 56221	0.1021 18597	0.1309 94209	0.10127 0132
Radiohead Creep.mp3	4,26 MB (4.476.268 *bytes*)	4,26 MB (4.476.396 *bytes*)	0.09620 2434	0.0628 37312	2.3670 12069	1.6319 23074	2.3347 48154	1.67004 841
Jessie J Sweet Talker Acoustic in Camden Transmitter Live – YouTube.MP4	14,1 MB (14.829.128 *bytes*)	14,1 MB (14.829.256 *bytes*)	0.23932 961	0.2694 51834	7.5336 74743	5.1797 56151	7.3985 75057	5.27909 1578
PADI - Semua Tak Sama – YouTube.MKV	13,2 MB (13.848.250 *bytes*)	13,2 MB (13.848.378 *bytes*)	0.26107 4951	0.2008 37774	6.6541 54657	4.9037 14173	6.6544 91648	4.86067 3811

Eksperimen seluruhnya berhasil dilakukan, dapat mengenkripsi file dengan baik sehingga file yang terenkripsi tidak dapat dibuka secara langsung, hanya orang yang memiliki kunci lah yang dapat mengembalikan file tersebut ke bentuk semula.

5. KESIMPULAN

Pada penelitian kali ini peneliti dapat menyimpulkan beberapa hal penting yaitu sebagai berikut:

1. Peneliti berhasil membangun software bernama VertoCrypt berbasis desktop untuk mengenkripsi file sehingga file dapat terlindungi dengan mengimplementasikan metode SHA-512 untuk verifikasi kunci serta *Autokey cipher* dan AES-128 untuk enkripsi file.
2. Software berhasil mengenkripsi dan mendekripsi file dokumen teks, gambar, musik, maupun video dengan ekstensi .txt, .doc, .pdf, .jpg, .png, .mp3, .mp4, dan .mkv.

3. Ukuran file hasil dari proses enkripsi bertambah besar dari file aslinya akan tetapi ukuran kembali seperti semula setelah file didekripsi.
4. Metode akan lebih lama dipecahkan oleh kriptanalis dengan alasan langkah untuk mengenkripsi *file* cukup banyak karena tidak hanya menggunakan satu metode tetapi lebih, selain itu kriptanalis tidak akan menyadari bahwa bit awal dari *file* yang terenkripsi merupakan hash kunci hasil fungsi SHA-512, terlebih lagi hash kunci tidak ditulis sepanjang 512 bit atau 64 *byte* melainkan 128 *byte*.

DAFTAR PUSTAKA

[1] Z. Ardi, K. Viola, and I. Sukmawati, "An Analysis of Internet Abuses Impact on Children ' s Moral Development," *JPPI (Jurnal Penelit. Pendidik. Indones.*, vol. 4, pp. 44–50, 2018.
[2] K. Saranya, R. Mohanapriya, and J. Udhayan, "A Review on Symmetric Key Encryption," *Int. J. Sci. Eng. Technol. Res.*, 2014.
[3] A. S. Wardani, "4,5 Miliar Data Dicuri Selama 6 Bulan Pertama 2018," *www.liputan6.com*, 2018.
[4] P. V Maitri and A. Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm," in *Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2016*, 2016, pp. 1635–1638.
[5] M. Mohan, M. K. K. Devi, and V. J. Prakash, "Security Analysis and Modification of Classical Encryption Scheme," *Indian J. Sci. Technol.*, vol. 8, no. 8, pp. 542–548, 2015.
[6] V. R. Pancholi and B. P. Patel, "Enhancement of Cloud Computing Security with Secure Data Storage using AES," *Int. J. Innov. Res. Sci. Technol.*, vol. 2, no. 09, 2016.
[7] S. T. & A. P. S. Siledar, "File Encryption , Decryption Using AES Algorithm in Android Phone International Journal of Advanced Research in File Encryption , Decryption Using AES Algorithm in Android Phone," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 5, pp. 550–554, 2015.
[8] R. Olufemi Akinyede and O. Aidohelen Esese, "Development of a Secure Mobile E-Banking System," *Int. J. Comput.*, vol. 26, pp. 23–42, 2017.