Pengembangan Algoritma Block Cipher pada V Cipher, dan Perbandingannya dengan Algoritma RSA

Putra Hardi Ramadhan and 13516080[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516080@std.stei.itb.ac.id

***Abstract*— Kriptografi adalah salah satu ilmu yang diaplikasikan untuk mengamankan informasi. Block cipher adalah salah satu algoritma Kriptografi Modern yang paling banyak digunakan saat ini. Algoritma ini memiliki banyak variasi, dikarenakan konsepnya yang memungkinkan perancang algoritma untuk merancang algoritma serumit mungkin untuk dipecahkan. V Cipher yang penulis buat merupakan salah satu variasi block cipher. V Cipher memiliki konsep yang sama seperti Algoritma DES, tetapi dilengkapi dengan penggunaan randomizer untuk mengurangi pola yang berulang selama enkripsi, sekaligus menambahkan proses enkripsi baru berupa pengaplikasian masking bits. RSA di bidang kriptografi adalah sebuah algoritma pada enkripsi kunci publik. RSA merupakan algoritma pertama yang cocok untuk digital signature seperti halnya enkripsi, dan salah satu yang paling maju dalam bidang kriptografi public key. RSA masih digunakan secara luas dalam protokol electronic commerce, dan dipercaya dalam mengamankan dengan menggunakan kunci yang cukup panjang.**

***Keywords*— Block Cipher, , RSA, Feistel, Confusion, Diffusion, Randomizer, Masking Bits, Seed.**

I. PENDAHULUAN

Pada era digital informasi telah tersebar secara luas danmudah. Seluruh informasi terbagikan melalui internet dan semua orang dapat mengirim dan menerima informasi tersebut. Informasi tersebut dapat berupa publik maupun rahasia. Informasi yang bersifat publik maupun rahasia perlu dijaga keamanannya dan integritasnya.

Algoritma block cipher adalah salah satu algoritma kriptografi modern. Algoritma kriptografi block cipher bekerja pada suatu data yang berbentuk blok/kelompok data dengan panjang data tertentu (dalam beberapa byte), jadi dalam sekali proses enkripsi atau dekripsi data yang masuk mempunyai ukuran yang sama.

Pada algoritma penyandian blok (block cipher), plainteks yang masuk akan diproses dengan panjang blok yang tetap yaitu n, namun terkadang jika ukuran data ini terlalu panjang maka dilakukan pemecahan dalam bentuk blok yang lebih kecil. Jika dalam pemecahan dihasilkan blok data yang kurang dari jumlah data dalam blok maka akan dilakukan proses padding (penambahan beberapa bit).

Block cipher merupakan algoritma pemetaan blok – blok plaintext ke blok – blok ciphertext. Pada suatu teks sandi sepanjang n-bit, terlebih dahulu kita bagi dalam beberapa blok – blok dengan ukuran panjang yang sama. Dengan kunci yang sama dan dengan algoritma tertentu, blok – blok ini dienkripsi, dan hasil outputnya pun berupa blok – blok sandi yang terenkripsi dan berukuran sama.

Block cipher memiliki beberapa keuntungan, yaitu mudahnya implementasi algoritma block cipher ke dalam software – software. Error Propagation yang terjadi pun tidak merambat ke ciphertext lainnya karena enkripsi masing – masing bloknya independen. Namun, clock cipher sangat mudah dianalisis karena blok – blok yang dienkripsi saling independen dan kuncinya sama, maka hal ini memudahkan kriptanalis untuk mengetahui kunci yang digunakan.

Algortima RSA dijabarkan pada tahun 1977 oleh tiga orang: Ron Rivest, Adi Shamir dan Len Adleman dari Massachusetts Institute of Technology. Singkatan RSA itu sendiri berasal dari inisial nama mereka (Rivest—Shamir—Adleman). Clifford Cocks, seorang matematikawan Inggris yang bekerja untuk GCHQ, menjabarkan tentang sistem ekuivalen pada dokumen internal pada tahun 1973.

Penemuan Clifford Cocks tidak terungkap hingga tahun 1997 karena alasan top-secret classification. Algoritma tersebut dipatenkan oleh Massachusetts Institute of Technology pada tahun 1983 di Amerika Serikat sebagai U.S. Patent 4.405.829. Paten tersebut berlaku hingga 21 September 2000. Semenjak Algoritme RSA dipublikasikan sebagai aplikasi paten, regulasi di sebagian besar negara-negara lain tidak memungkinkan penggunaan paten. Hal ini menyebabkan hasil temuan Clifford Cocks di kenal secara umum, paten di Amerika Serikat tidak dapat mematenkannya.

II. LANDASAN TEORI

2.1 Metode Block Cipher

Ada 5 method dalam pengenkripsian dalam block cipher, yaitu metode Electronic Code Book, Cipher Block Chaining, Cipher Feedback, Output Feedback, dan Counter.

1. Electronic Code Book (ECB)

ECB merupakan metode standar dari Block Cipher, yaitu setiap blok plainteks dienkripsi dengan kunci yang sama secara satu persatu. Kelemahan utama dari metode ini adalah mudahnya pendeteksian, terutama jika ada blok – blok data yang sama dan dienkripsi dengan kunci yang sama maka akan menghasilkan cipherteks yang sama pula. Hal inilah yang menyebabkan mengapa disebut Electronic Code Book, karena

seolah kita dapat mengetahui dan membuat sebuah kamus atau ensiklopedi dari plainteks dan cipherteks dengan kunci yang sama, dan sangat mudah di kriptanalisis.

2. Cipher Block Chaining (CBC)

Metode CBC didesain untuk mengatasi kelemahan pada metode ECB. Pada dasarnya, enkripsi yang dilakukan sama dengan ECB, namun terdapat perbedaan disini. Plaintext yang akan dienkripsi terlebih dahulu dilakukan XOR dengan ciphertext fase sebelumnya. Untuk fase pertama digunakan Initialization Vector (IV) sebagai nilai awal, yang kemudian di XOR dengan blok plaintext yang selanjutnya dilakukan enkripsi dengan kunci yang telah disepakati. Selanjutnya, blok ciphertext yang dihasilkan, selain dikeluarkan melalui output, blok ciphertext tersebut dilakukan XOR lagi dengan blok plaintext yang selanjutnya.

Pada metode dekripsi, yang dilakukan terlebih dahulu adalah mendekripsi blok ciphertext yang kemudian dilakukan XOR dengan ciphertext sebelumnya, dimana untuk fase awal digunakan Initialization Vector yang sama dengan pada saat enkripsi. Selanjutnya, blok ciphertext ini di-XOR dengan blok ciphertext selanjutnya setelah dilakukan proses dekripsi.

3. Cipher Feedback (CFB)

Metode Cipher Feedback menggunakan sistem Shift Register, dimana yang diproses terlebih dahulu adalah Initialization Vector dalam algoritma enkripsi dengan kunci. Setelah diproses, bit yang dihasilkan akan melalui proses seleksi bit, biasanya bit – bit yang paling kiri, untuk selanjutnya dienkripsi dengan plaintext untuk menghasilkan ciphertext. Bit hasil seleksi yang digunakan tergantung besarnya bit blok plaintext yang diinput. Selanjutnya, setelah mendapatkan blok ciphertext, selain di output, blok ciphertext tersebut dimasukkan ke IV yang sebelumnya, dan IV digeser sebanyak bit blok ciphertext sebelumnya, yang selanjutnya IV yang telah digeser bersama blok ciphertext yang digabung bersama IV tersebut diproses kembali oleh algoritma enkripsi tersebut.

4. Output Feedback (OFB)

Metode OFB memiliki perbedaan dengan CFB, dimana input yang digunakan dalam proses enkripsi. Kalau dalam CFB, input yang digunakan adalah ciphertext yang selanjutnya di-shift bersama IV, dalam OFB yang digunakan adalah output bit hasil dari proses seleksi yang kemudian di shift bersama IV yang sebelumnya. Hasil seleksi tetap digunakan dalam proses enkripsi plaintext untuk mendapatkan ciphertext.

5. Counter Mode

Mode Counter memanfaatkan sebuah nilai counter yang dienkripsi menggantikan blok-blok plaintext. Nilai counter harus berbeda dari setiap blok yang dienkripsi. Pada mulanya, untuk enkripsi blok pertama, counter diinisialisasi dengan sebuah nilai. Selanjutnya, untuk enkripsi blok-blok berikutnya counter dinaikkan nilainya satu. Blok plaintext akan kemudian di-XORkan dengan hasil enkripsi untuk menghasilkan blok-blok ciphertext.

2.2 *Shannon's Confusion dan Diffusion*

Dalam kriptografi, prinsip *diffusion* dan *confusion* adalah 2 sifat dari *secure cipher* yang diperkenalkan oleh Claude Shannon pada tahun 1945. Prinsip yang dikemukakan pada publikasinya—yang berjudul *Communication Theory of Secrecy Systems*—dibuat dengan tujuan untuk mempersulit beberapa metode kriptanalisis seperti metode statistik. Pengertian *confusion* menurut Shannon adalah hubungan antara *ciphertext* dan kunci yang dibuat serumit mungkin sehingga kriptanalis frustasi dalam menemukan hubungan antara keduanya. Sedangkan untuk *diffusion* mengacu pada persebaran pengaruh kunci terhadap bit-bit *plaintext* yang seluas mungkin. Sebagai contoh pada *vigenere cipher*, prinsip *confusion* diterapkan sehingga hubungan antara kunci dan *ciphertext* tidak serta merta terlihat—dengan membuat *ciphertext* yang berasal dari operasi penjumlahan karakter pesan dan karakter kunci.

Diffusion bertujuan agar perubahan bit pada *ciphertext* menampilkan hasil pesan di luar prediksi kriptanalis. Untuk mendapatkan *cipher* dengan keamanan yang tinggi, prinsip *diffusion* dan *confusion* diterapkan secara berulang dalam sebuah blok tunggal dengan kombinasi yang berbeda-beda. Metode paling sederhana untuk memenuhi prinsip *confusion* dan *diffusion* adalah menerapkan jaringan substitusi dan permutasi.

2.3. *Feistel Network*

Feistel network atau jaringan Feistel adalah struktur simetris yang digunakan dalam mengkonstruksi *block cipher*. Model dasar jaringan Feistel membagi blok masukan menjadi beberapa bagian, menggunakan fungsi Feistel kepada upablok-upablok tersebut, kemudian melakukan operasi XOR antara upablok-upablok tersebut. Kunci enkripsi (*round key*) digunakan pada tiap fungsi Feistel yang dikenakan pada upablok. Rangkaian operasi ini dapat dilakukan berulang kali selama beberapa putaran atau *round*. *Round key* pada setiap putaran dibangkitkan secara pseudo-random. Karena sifatnya yang simetris, jaringan Feistel memiliki keunggulan yaitu algoritma yang digunakan untuk proses enkripsi dapat digunakan lagi untuk proses dekripsi. Kelebihan lain dari jaringan Feistel yang membuatnya menjadi menarik untuk dibahas adalah karena fungsi Feistel yang dikenakan pada upablok dapat dirancang sesulit apapun. Perancang fungsi tidak perlu khawatir akan kehilangan properti reversible dari jaringan Feistel. Selain itu perancang fungsi juga tidak perlu menentukan fungsi balikan (f-invers) dari fungsi Feistel tersebut.

2.4 *Algoritma RSA*

RSA merupakan algoritma kriptografi asimetri, dimana kunci yang digunakan untuk mengenkripsi berbeda dengan yang digunakan untuk mendekripsi. Kunci yang digunakan untuk mengenkripsi disebut dengan kunci public atau Public Key, dan yang digunakan untuk mendekripsi disebut dengan kunci privat atau Private Key

RSA dikenal juga sebagai salah satu algoritma kriptografi yang menggunakan konsep kriptografi kunci publik. RSA membutuhkan tiga langkah dalam prosesnya, yaitu : pembangkitan kunci, enkripsi, dan dekripsi. Dimana proses enkripsi dan dekripsi merupakan proses yang hampir sama, maksudnya jika bilangan acak yang dibangkitkan kuat, maka akan lebih sulit untuk melakukan cracking terhadap pesan. Parameter yang dijadikan kuat tidaknya suatu kunci yaitu terdapat pada besarnya bilangan acak yang digunakan.

Berikut ini adalah contoh perhitungan manual enkripsi dan dekripsi menggunakkan algoritma RSA. Dimana sebelum-nya kita harus menentukkan dulu Public Key Dan Private Key nya.Dan berikut langkah-langkah algoritma RSA mendapatkan Public Key Dan Private Key :

1. Pertama, menentukan 2 buah bilangan prima untuk p dan q :
   p = 11
   q = 13
2. Selanjutnya mendapatkan nilai n dimana rumus-nya :
   n = p * q, dan akan menjadi seperti ini :
   n = 11 * 13
   n = 143
3. Mendapatkan nilai m dimana rumus-nya :
   m = (p - 1) * (q - 1), dan akan menjadi seperti ini :
   m = (11 - 1) * (13 - 1)
   m = (10) * (12)
   m = 120
4. Menentukan nilai e dengan syarat :
   e = e > 1 and GCD(m,e) = 1
   Dimana "17" adalah nilai yang memenuhi syarat nilai e
   e = GCD(120,17) = 1
5. Menentukan nilai d dengan syarat :
   d = (d * e) mod m = 1
   Dimana "473" adalah nilai yang memenuhi syarat nilai d
   d = (473 * 17) mod 120 = 1
6. Dari proses diatas, maka akan mendapatkan kunci public dan kunci privat dimana :
   public key = (e,n), private key = (d,n)
   Dan kunci akan menjadi seperti ini :
   public key = (17,143), private key = (473,143)
7. Setelah kita mendapatkan public key dan private key, proses selanjutnya melakukan Enkripsi dan Dekripsi, yaitu kata "INDONESIA". Berikut prosesnya :

Teks	ASCII	Enkripsi C=A^e mod n	Dekripsi Y = C^d mod n
I	73	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 9 = 50.9	= (50 ^ 473) mod 143 = 7 = (9 ^ 473) mod 143 = 3 = 73 = I
N	78	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 112 = 50.112	= (50 ^ 473) mod 143 = 7 = (112 ^ 473) mod 143 = 8 = 78 = N
D	68	= (7 ^ 17) mod 143 = 41 = (3 ^ 17) mod 143 = 112 = 41.112	= (41^ 473) mod 143 = 6 = (112 ^ 473) mod 143 = 8 = 68 = I
O	79	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 9 = 50.9	= (50 ^ 473) mod 143 = 7 = (81 ^ 473) mod 143 = 9 = 73 = I
N	78	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 112 = 50.112	= (50 ^ 473) mod 143 = 7 = (112 ^ 473) mod 143 = 8 = 78 = N
E	69	= (7 ^ 17) mod 143 = 41 = (3 ^ 17) mod 143 = 81 = 41.81	= (41 ^ 473) mod 143 = 6 = (81 ^ 473) mod 143 = 9 = 69 = E
S	83	= (7 ^ 17) mod 143 = 112 = (3 ^ 17) mod 143 = 9 = 112.9	= (112^ 473) mod 143 = 8 = (9 ^ 473) mod 143 = 3 = 83 = S
I	73	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 9 = 50.9	= (50 ^ 473) mod 143 = 7 = (9 ^ 473) mod 143 = 3 = 73 = I
A	65	= (7 ^ 17) mod 143 = 41 = (3 ^ 17) mod 143 = 135 = 41.135	= (41 ^ 473) mod 143 = 6 = (135 ^ 473) mod 143 = 5 = 65 = A

III. IMPLEMENTASI

A. Implementasi Algoritma V Cipher

3.1 Struktur Algoritma

Secara struktur, algoritma V *cipher* tidak berbeda jauh dengan algoritma DES, dimana proses enkripsi terdiri dari :

1. *Initial Permutation*
2. *Iterated Cipher* dengan *Feistel Network*
3. *Inverse Permutation*

Tahapan *Initial Permutation* dan *Inverse Permutation* tidak berbeda dari tahapan serupa pada algoritma DES, sementara tahap *Iterated Cipher* mengalami beberapa perubahan yang menggunakan *randomizer*, dan akan dijelaskan pada subbab-subbab selanjutnya.

Pemrosesan *plaintext* dilakukan dalam blok-blok berukuran 64 *bit*. Kunci eksternal yang digunakan berukuran 64 *bit* dengan 8 *bit* paritas layaknya algoritma DES. Akan tetapi, panjang kunci eksternal yang dimasukkan oleh pengguna tidak dibatasi sepanjang 8 karakter saja. Pengguna dapat memasukkan kunci eksternal dengan panjang kurang atau lebih dari 8 karakter, selama tidak 0, layaknya sebuah *password*. Dari kunci eksternal ini, akan dibangkitkan sebuah *seed* yang akan digunakan untuk menginisiasi *randomizer* yang nantinya akan digunakan untuk membangkitkan kunci-kunci internal untuk proses *Iterated Cipher* nantinya. Proses pembangkitan *seed* adalah sebagai berikut:

1. Ambil nilai sebuah *initial seed* berupa *integer value* dari setiap karakter pada kunci eksternal
2. Gunakan *initial seed* tersebut untuk menginisiasi *randomizer*
3. Inisialisasi nilai *seed* dengan *value* 0
4. Gunakan *randomizer* untuk mendapatkan nilai *n* berupa sebuah bilangan antara 0 dan panjang kunci eksternal - 1
5. Tambahkan *integer value* dari karakter pada indeks *n* dari kunci eksternal ke dalam nilai *seed*
6. Ulangi langkah 4 dan 5 hingga (panjang kunci eksternal / 2) kali
7. Simpan nilai *seed*

3.2 Pembangkitan Kunci Internal

Proses pembangkitan kunci internal tidak jauh berbeda dengan proses pembangkitan kunci eksternal pada algoritma DES, namun jumlah kunci yang dibangkitkan hanya sejumlah 8 buah, karena proses *Iterated Cipher* nantinya hanya akan dilakukan sebanyak 8 kali putaran. Selain itu, proses pergeseran saat membangkitkan kunci internal kini ditentukan oleh *seed* yang didapat dari kunci eksternal. Proses pembangkitan kunci internal adalah sebagai berikut:

1. Inisialisasi sebuah list *keys* untuk kunci-kunci internal yang akan dibangkitkan
2. Inisialisasi variabel *key* dengan kunci eksternal. Jika kunci eksternal bukan merupakan kelipatan 8, maka tambahkan *padding* dengan karakter *dummy* hingga *key* kelipatan 8
3. Jika panjang *key* lebih dari 8 karakter, maka bagi menjadi blok-blok 8 karakter, lakukan XOR pada masing-masing *bit* pada blok-blok 8 karakter, dan simpan hasilnya ke dalam *key*

4. Lakukan permutasi pada *key* menggunakan matriks permutasi 1 untuk mendapatkan 56 *bit* yang akan dipakai, lalu bagi menjadi 2 blok 28 *bit*
5. Inisiasi *randomizer* dengan *seed* yang didapat dari kunci eksternal
6. Gunakan *randomizer* untuk mendapatkan nilai *n* berupa sebuah bilangan antara 1 dan 27
7. Geser *bit-bit* pada kedua blok 28 *bit* ke kanan sejauh *n* posisi
8. Lakukan permutasi pada kedua blok 28 *bit* dengan matriks transformasi 2 untuk mendapatkan kunci internal sepanjang 48 *bit* dan simpan di list *keys*
9. Ulangi langkah 6 hingga 8 sampai terbentuk 8 buah kunci internal

3.3 Iterated Cipher

Proses ini juga tidak jauh berbeda dengan algoritma DES, tetapi putaran hanya dilakukan sebanyak 8 kali. Jumlah *S-Box* yang digunakan untuk proses substitusi lebih sedikit dari algoritma DES biasa, yaitu hanya sebanyak 4 buah. Namun, penentuan penggunaan *S-Box* akan kembali diatur oleh *randomizer*, dimana *seed* yang akan digunakan dibangkitkan dari masing-masing kunci internal.

Proses baru yang ditambahkan adalah penggunaan *masking bits* pada saat proses enkripsi berulang, dimana kedua blok kode akan di XOR kan dengan *masking bits* yang dibangkitkan oleh *randomizer* yang juga menggunakan *seed* dari kunci internal. Terdapat 2 kali 32 *masking bits* yang akan dibangkitkan, dan 32 *bit* terakhir merupakan hasil kebalikan dari 32 *bit* pertama. Sebagai contoh, apabila *masking bits* pertama adalah 1010, maka *masking bits* kedua adalah 0101. Proses ini ditambahkan sebagai enkripsi terakhir setelah fungsi enkripsi *f* pada jaringan Feistel selesai dilaksanakan. Proses *Iterated Cipher* adalah sebagai berikut:

1. Lakukan permutasi awal untuk mengacak urutan *plaintext*
2. Bagi blok 64 *bit plaintext* menjadi blok L dan R yang masing-masing memiliki panjang 32 *bit*
3. Gunakan matriks ekspansi untuk meng-*expand* blok R menjadi 48 *bit*
4. Lakukan XOR antara blok R dengan kunci internal untuk putaran saat ini
5. Bagi blok R menjadi 8 blok masing-masing 6 bit
6. Inisialisasi *randomizer* dengan *seed* dari kunci internal. Untuk mendapatkan nilai *seed*, pertama nilai masing-masing *bit* pada kunci internal dengan indeks *bit* tersebut dan dijumlahkan untuk mendapatkan *initial seed*. Lalu setelah *randomizer* diinisiasi dengan *initial seed*, bangkitkan angka *random n* dengan nilai antara 0 hingga 47. *Seed* yang memiliki nilai awal 0 akan dijumlahkan dengan nilai *bit* pada indeks *n* dikalikan dengan *n*. Ulangi proses sebanyak 24 kali
7. Gunakan *randomizer* untuk menghasilkan 8 buah angka antara 1 hingga 4. Angka yang dihasilkan akan menentukan indeks *S-Box* yang akan digunakan untuk masing-masing blok 6 *bit*
8. Gunakan *S-Box* untuk mensubstitusi nilai 6 *bit* menjadi 4 *bit*, dan gabungkan semuanya menjadi blok R yang baru sepanjang 32 *bit*
9. Lakukan permutasi untuk mengacak urutan blok R
10. Lakukan XOR antara blok R baru dengan blok L lama untuk mendapatkan blok L baru
11. Dengan *seed* dari kunci internal, bangkitkan 2 blok *masking bits* yang masing-masing berukuran 32 *bit*. Pertama akan dibangkitkan 32 bilangan *random*. Isi *masking bits* pertama dengan 0 apabila bilangan *random* genap, dan 1 apabila bilangan *random* ganjil. Setelah didapat blok *masking bits* pertama, buat blok kedua dengan membalikkan nilai masing-masing *bit* pada *masking bits* pertama
12. Lakukan XOR antara blok R lama dengan *masking bits* pertama, dan antara blok L baru dengan *masking bits* kedua
13. Blok L baru akan menjadi blok R di putaran berikutnya, dan blok R lama akan menjadi blok L baru di putaran selanjutnya
14. Ulangi langkah 3 hingga 13 sampai telah dilalui 8 putaran
15. Gabungkan blok L dan blok R menjadi blok 64 *bit*
16. Lakukan inversi permutasi untuk mengembalikan susunan *ciphertext*

B. Implementasi Algoritma V Cipher Baru

Pada Algoritma V Cipher diatas, terdapat satu kelemahan, yaitu apabila Plainteks dibah satu karakter, maka perubahan pada cipherteks juga hanya akan beruah padad titik itu. Sehingga, kelemahan tersebut dapat dimanfaatkan oleh Kriptanalis untuk mendekripsi pesan cipherteks yang ada. Hal ini membuat Algoritma V Cipher diatas masih kurang aman.

Oleh karena itu pada penelitian kali ini akan dilkaukan ujicoba lebih lanjut pada V cipher jenis baru. V cipher jenis baru akan mengimplementasikan *randomizer* dan proses enkripsi baru. Dengan menambah pengaruh *randomizer* di bagian-bagian lain pada proses enkripsi. Selain itu, penambahan proses enkripsi baru di antara proses yang sudah ada juga dapat menambahkan kompleksitas dari algoritma saat ini. Penmabhan algoritma tersebut adalah penggunaan beberapa algoritma sederhana seperti *playfair cipher* dan *Caesar cipher* agar cipherteks lebih sulit untuk didekripsi dan di kripatanalis.

IV. EKSPERIMEN DAN HASIL ANALISIS

Berikut adalah hasil EKsperimen Algoritma V Cipher Lama dan Baru, serta perandingannya dengan algoritma RSA. Dalam pengujian, dilakukan dengan membandingkan perubahan cipherteks, dengan perbahan pada plainteks. Plainteks yang digunakan pada setiap algoritma adalah sama. Plainteks kemudian akan diubah satu karakter, apabila perubahan pada cipherteks menjadi sangat jauh berubah maka algoritma itulah yang lebih baik dan dapat menjaga kerahasiaan Plainteks. Percobaan atau eksperimen ini disebut juga ujicoba *confusion dan diffusion*

Kalimat yang digunakan pada eksperimen adalah “Kegiatan belajar kini tidak hanya dilakukan di dalam kelas. Proses pembelajaran kini bisa dilakukan secara online. Sejumlah situs menyediakan program pembelajaran online, mulai dari yang berbayar hingga ada yang gratis. Kebebasan memilih mata

pelajaran, serta kegiatan belajar yang bisa dilakukan kapan pun adalah kelebihan dari sistem pembelajaran online ini. Kini, kegiatan belajar pun bisa dilakukan secara online." Dengan pada setiap percobaan, karakter pada posisi 1, 10, 25, 35, dan 50 akan diganti.

Posisi penggantian	Jumlah kesamaan Byte	Persentase Kesamaan Cipherteks
Posisi Pertama	2/424	0,47%
Posisi Ke-10	9/424	2,1%
Posisi Ke-25	21/424	5%
Posisi Ke-35	36/424	8,5%
Posisi Ke-50	46/424	10,9%
Rata-rata	22.8/424	5,3%

Pada Tabel diatas dapat dilihat bahwa V cipher memilki kecenderungan untuk tidak mengubah plainteks pada again sebelum ada karakter yang diganti, dari pesan asli. Dengan demikian, semakin jauh posisi karakter yang diganti, maka akan semakin sama perubahan yang terjadi pada teks. Hal ini dapat menyebabkan pesan bisa dideteksi oleh kriptanalis.

Selanjutnya adalah percobaan pada V cipher yang telah coba untuk diperbaiki. Percobaan akan dilakukan sama persis, agar dapat dibandingkan dengan jelas.

Posisi penggantian	Jumlah kesamaan Byte	Persentase Kesamaan Cipherteks
Posisi Pertama	0/424	0%
Posisi Ke-10	8/424	1,8%
Posisi Ke-25	17/424	4,01%
Posisi Ke-35	28/424	6,6%
Posisi Ke-50	35/424	8,25%
Rata-rata	17.6/424	4,15%

Pada Tabel diatas dapat ditunjukkan bahwa dengan penempatan posisi dan perubahan karakter yang sama dengan percobaan sebelumnya, V cipher yang telah coba diperbaiki lebih sedikit memiliki kesamaan dengan pesan asli. Walaupun perubahan masih belum signifikan, namn sudah cukup berhasil karena semakin jauh perubahan karakter, semakin beda selisihnya dengan percobaan sebelumnya.

Sebagai perbandingan, akan dilakkan ekperimen yang sama pada Algoritma RSA, dengan pesan yang sama dan posisi perubahan karakter yang sama.

Posisi penggantian	Jumlah kesamaan Byte	Persentase Kesamaan Cipherteks
Posisi Pertama	0/424	0%
Posisi Ke-10	2/424	0,47%
Posisi Ke-25	1/417	0,24%
Posisi Ke-35	3/424	0,71%
Posisi Ke-50	2/424	0,47%
Rata-rata	1.6/424	0,38%

Pada Tabel diatas dapat dilihat bahwa perubahan pada plainteks tidak terlalu mempengaruhi kesamaan pada hasil cipherteks. Artinya, algoritma sudah dengan berhasil dapat menghasilkan cipherteks yang berbeda, walaupun hanya satu karakter yang diubah. Hasil ini tentu jauh lebih baik jika dibandingkan dengan dua percobaan sebelumnya.

V. KESIMPULAN

Algoritma V *Cipher* yang didesain sudah memberikan hasil yang baik dalam mengaplikasikan konsep keamanan seperti prinsip *confusion* dan *diffusion*, penangan terhadap serangan *brute force*, dan juga sudah relatif aman jika melihat hasil pada analisis pengulangan *hexadecimal*.

Algoritma V *Cipher* sudah berhasil ditingkatkan lebih lanjut dengan menambah pengaruh *randomizer* di bagian-bagian lain pada proses enkripsi. Selain itu, penambahan proses enkripsi baru di antara proses yang sudah ada sudah dapat berhasil menambahkan kompleksitas dari algoritma saat ini.

Walaupun jika dibandingkan dengan algoritma yang sudah paten seperti algoritma RSA, hasil dari algoritma V cipher masih berbeda jauh, namun dengan penelitian ini dapat ditunjkkan bahwa algoritma yang sudah ad adapt dikembangkan dan kedepannya akan coba untuk dikembangkan lebih lanjut.

REFERENCES

[1] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Kriptografi Modern (Bagian 3). Diakses pada tanggal 21 Desember 2020

[2] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Kriptografi Modern (Bagian 4). Diakses pada tanggal 21 Desember 2020

[3] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Data Encryption Standard (DES). Diakses pada tanggal 21 Desember 2020

[4] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Algoritma RSA. Diakses pada tanggal 21 Desember 2020

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21 Desember 2020

Putra Hardi Ramadhan 13516080