Dominos

Ahmad Fajar Prasetiyo (13514053)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514053@std.itb.ac.id

***Abstract*—Dominos merupakan algoritma *block cipher* yang terinspirasi oleh Domino. Dominos menerapkan beberapa prinsip kriptografi klasik yaitu substitusi dan pergeseran. Dominos beroperasi pada ukuran blok yang memiliki kelipatan 32 bytes. Panjang kunci eksternal yang dibutuhkan Dominos bergantung pada panjang blok. Dominos memiliki tiga jenis kunci internal yang dibangkitkan dari kunci eksternal. Ada tiga tahapan dalam Dominos. Dominos dapat beroperasi pada mode ECB, CBC, CFB, OFB dan *Counter Mode*. Dominos memiliki aspek *confusion* dan *diffusion*.**

***Keywords—Kriptografi, Block Cipher, Jaringan Feistel, Domino, Confusion, Diffusion, Pseudorandom*.**

I. Pendahuluan

Kriptografi adalah seni dalam menulis. Kriptografi yang paling pertama yang bisa diketahui muncul pada 4000 tahun yang lalu. Kriptografi paling tua yang sekarang diketahui adalah kriptografi dengan menggunakan *hieroglyph*.

Hieroglyph digunakan untuk berkomunikasi dengan bangsa mesir kuno. *Hieroglyph* ini digunakan untuk kepentingan raja mesir kuno. *Hieroglyph* belum berbentuk tulisan tetapi berbentuk gambar.

Setelah *hieroglyph* muncul kriptografi geser sederhana yaitu *caesar shift cipher*. *Caesar shift cipher* digunakan oleh bangsa Romawi. Kunci dari *caesar shift cipher* adalah jumlah pergeserannya. Jadi banyaknya ruang kunci dari *caesar shift cipher* adalah 26, oleh karena itu *caesar shift cipher* sangat mudah untuk dilakukan *brute force*.

Setelah itu berkembang algoritma kriptografi yang baru yaitu monoalphabetic substitusi. Monoalphabetic substitusi pertama kali muncul sekitar 600-500 tahun sebelum masehi. Cara kerja dari monoalphabetic substitusi mengganti setiap huruf dengan huruf yang lain. Kunci dari monoalphabetic substitusi adalah tabel pergantian huruf. Banyak kemungkinan kunci dari monoalphabetic substitusi adalah 26!. Pada zaman dahulu sulit untuk dilakukan *brute force* karena harus mencoba 26!, tetapi sekarang mudah karena adanya komputer.

Munculnya komputer juga membuat orang harus mengganti kriptografi yang lama karena komputer sangat cepat dalam melakukan *brute force*. Perkembangan komputer menjadi pemicu munculnya kriptografi modern.

Kriptografi modern masih menggunakan prinsip-prinsip dari kriptografi klasik yang digabungkan. Kriptografi modern sering beroperasi dalam bentuk *bit*. Operasi yang paling banyak digunakan dalam kriptografi modern adalah operasi XOR. Kriptografi modern terbagi menjadi dua kelompok yaitu *stream cipher* dan *block cipher*.

Pada makalah ini akan dibahas kriptografi *block cipher* yang baru. Algoritma ini dinamakan Dominos karena terinspirasi dari domino yang setiap hasil dari enkripsi bergantung pada nilai sebelumnya. Dominos juga menerapkan beberapa operasi kriptografi klasik yaitu substitusi dan pergeseran. Selain itu Dominos juga menggunakan operasi XOR dan jaringan feistel.

II. Dasar Teori

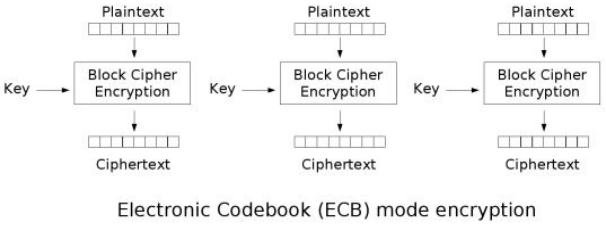
Pada bab ini akan dibahas teori-teori yang mendukung rancangan Dominos.

A. Block Cipher

Sub bab ini akan membahas tentang mode-mode dalam *block cipher*. Dalam *block cipher* terdapat 5 mode, yaitu *Electronic Code Block* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB), dan *Counter Mode*.

1. *Electronic Code Block* (ECB)

Pada mode ECB setiap block dienkripsi secara independen dari block lainnya. Sehingga nilai dari block sebelumnya tidak mempengaruhi nilai dari block yang akan dienkripsi.

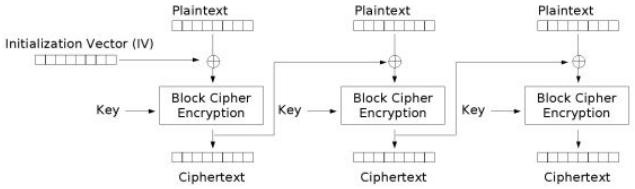


Sumber:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Gambar 1: Proses Enkripsi dalam ECB Mode

Dalam mode ECB memiliki beberapa kelemahan. Kelemahan yang paling fatal adalah nilai dari block *cipher* akan sama jika nilai dari block *plaintext* sama.

2. *Cipher Block Chaining* (CBC)

Pada CBC mode blok akan dienkripsi berdasarkan hasil dari blok sebelumnya. Blok tidak dienkripsi secara independen dengan blok lain. Oleh karena itu jika nilai *block plaintext* dalam blok memiliki nilai yang sama belum tentu memiliki nilai *block cipher* yang sama.



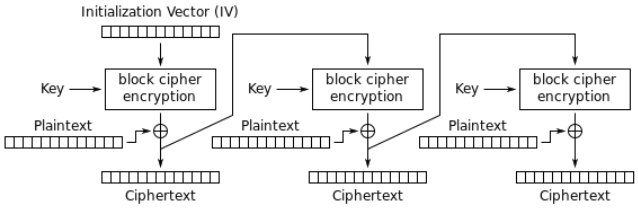
Cipher Block Chaining (CBC) mode encryption

Sumber:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Gambar 2: Proses Enkripsi dalam CBC Mode

Pada mode CBC nilai *cipher* akan rusak jika ada nilai dari *block plaintext* yang rusak. Karakteristik ini merupakan kelemahan sekaligus kelebihan. Karakteristik ini disebut sebagai kelebihan karena dapat mendeteksi bahwa *plaintext* rusak. Karakteristik ini disebut kelemahan karena akan rusak semua jika *plaintext* rusak.

3. *Cipher Feedback* (CFB)

Pada mode CFB data akan dienkripsi lebih kecil dari pada ukuran blok. Dalam mode ini bekerja seperti *stream cipher* yang beroperasi pada level *bit*. Pada mode ini membutuhkan sebuah antrian yang seukuran dengan panjang blok.

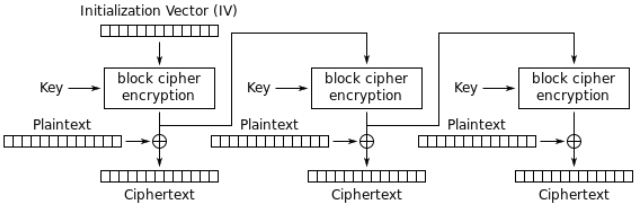


Cipher Feedback (CFB) mode encryption

Sumber:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Gambar 3: Proses Enkripsi dalam CFB Mode

4. *Output Feedback* (OFB)

Pada mode ini memiliki karakteristik dengan mode CFB. Perbedaan dari mode ini dan CFB adalah terletak pada dilakukan XOR pada *block cipher encryption*.

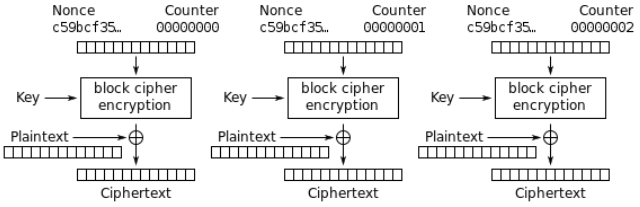


Output Feedback (OFB) mode encryption

Sumber:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Gambar 4: Proses Enkripsi dalam OFB Mode

5. *Counter Mode*

Pada counter mode tidak melakukan *chaining* seperti CBC. Setiap blok dilakukan enkripsi secara independen dengan blok lain. Setiap blok memiliki nilai *counter* yang berbeda. Nilai awal dari counter diinisialisasi dengan nilai sembarang. Panjang *bit* nilai *counteri* harus sama dengan panjang *bit* dari blok.



Counter (CTR) mode encryption

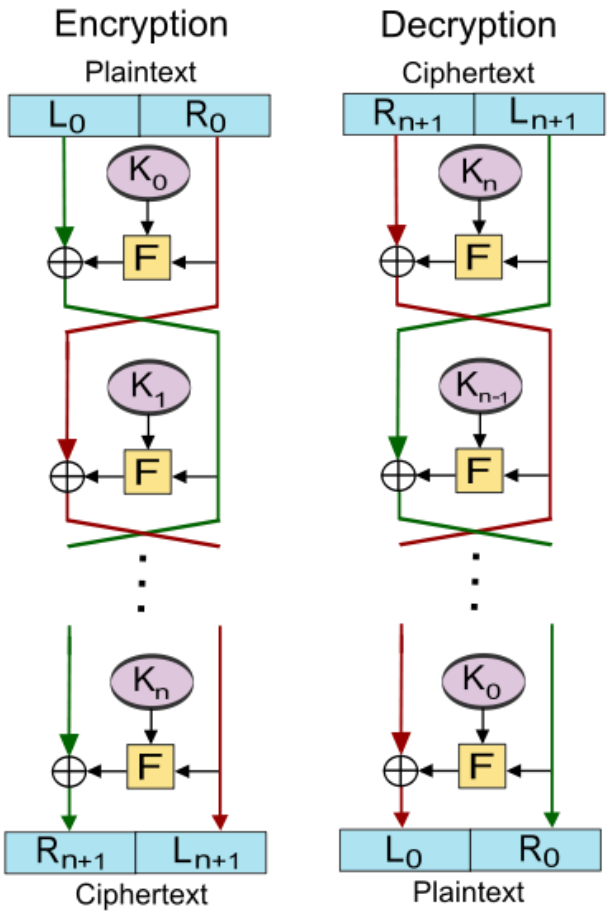
Sumber:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Gambar 5: Proses Enkripsi dalam *Counter Mode*

B. Jaringan Feistel

Jaringan Feistel banyak digunakan oleh algoritma kriptografi modern. Jaringan feistel pertama kali digunakan dalam algoritma yang bernama Lucifer. Selain Lucifer jaringan Feistel juga digunakan dalam *Data Encryption Standard* (DES).

Jaringan Feistel digunakan untuk proses enkripsi yang berulang-ulang. Proses enkripsi dan dekripsi dari jaringan Feistel sama saja dilakukan *inverse*. Sehingga tidak perlu untuk merancang algoritma khusus yang digunakan untuk dekripsi.

Jaringan Feistel terdiri dari beberapa *round*. Banyaknya *round* pada jaringan Feistel tergantung pada desain dari Algoritma. Ketika memasuki jaringan Feistel blok akan dibagi menjadi dua yaitu kanan dan kiri. Setiap *round* pada jaringan Feistel blok kanan dan blok kiri akan ditukar. Blok bagian kanan hanya akan diganti posisinya ke blok bagian kiri, sementara blok bagian kiri akan mengalami XOR dengan hasil *round function* yang mendapatkan input dari *round key* dan blok bagian kanan.



Sumber: https://en.wikipedia.org/wiki/Feistel_cipher

Gambar 6: Jaringan Feistel

C. Confusion dan Diffusion

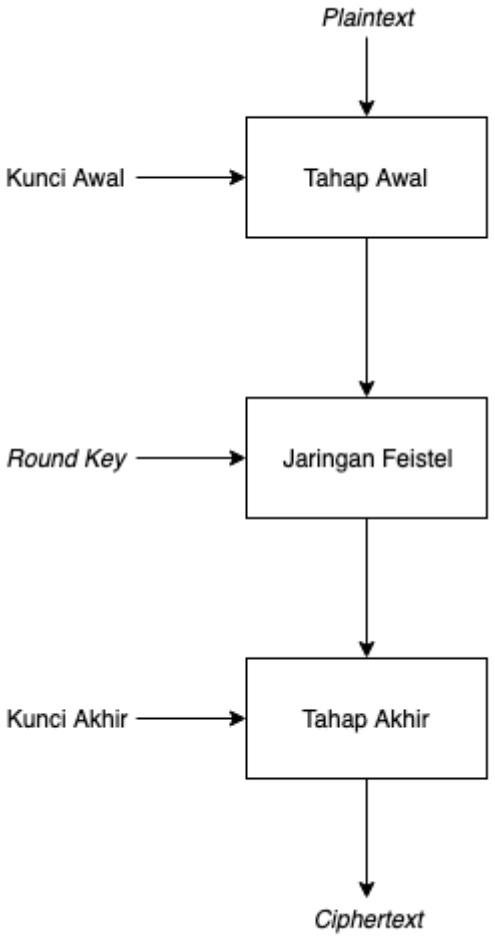
Confusion dan *diffusion* merupakan sifat yang harus ada dalam algoritma Kriptografi. *Confusion* dan *diffusion* pertama kali dikenalkan oleh seorang matematikawan yang bernama Shannon dalam journal yang berjudul *A Mathematical Theory of Cryptography* pada tahun 1945. Sifat ini mendasari seluruh algoritma kriptografi modern. Agar suatu algoritma kriptografi memiliki sifat ini harus menggunakan permutasi, substitusi dan jaringan Feistel.

Confusion adalah sifat yang menyembunyikan hubungan antara hubungan *plaintext* dan *ciphertext*. *Confusion* membuat *cryptanalyst* sulit untuk melakukan serangan dengan analisis statistika dari *ciphertext*. Sifat ini didapatkan dengan menggunakan substitusi yang komplek.

Diffusion adalah sifat yang menyebarkan pengaruh *plaintext* ke seluruh *chipertext*. Jika suatu algoritma memiliki sifat *diffusion* maka pergantian satu bit pada *plaintext* dapat mengubah beberapa *chipertext*. *Diffusion* didapat dengan menggunakan teknik permutasi.

III. Rancangan Algoritma

Pada bab ini akan dibahas secara detail rancangan dari Dominos. Terdapat tiga tahapan utama dalam Dominos yaitu, tahap awal, tahap *iterasi* dan tahap akhir.



Gambar 7: *Flowchart* dari Dominos

A. Perhitungan Kunci

Dominos beroperasi pada ukuran *byte*. Dominos adalah algoritma yang bertipe *Block Cipher*. Ukuran setiap block dalam Dominos adalah kelipatan dari 32 *byte*.

Panjang kunci *external* yang dibutuhkan dalam Dominos bergantung pada ukuran Block. Kunci *external* digunakan untuk membangkitkan 3 buah kunci.

Kunci yang pertama yang dibangkitkan adalah kunci awal. Kunci awal digunakan sebelum Block memasuki jaringan feistel. Ukuran dari kunci awal ini bergantung pada ukuran dari setiap Block. Panjang kunci awal adalah panjang Block dibagi dengan delapan.

Kunci yang dibangkitkan kedua adalah kunci *round*. Kunci *round* adalah kunci yang digunakan untuk jaringan feistel. Panjang dari kunci *round* bergantung pada banyaknya *iterasi*

pada jaringan feistel dan panjang dari setiap Block. Panjang dari kunci *round* adalah panjang dari kunci awal dikali dengan banyaknya *iterasi*.

Kunci terakhir yang dibangkitkan adalah kunci akhir. Kunci akhir digunakan ketika Block selesai memasuki jaringan feistel. Panjang dari kunci ini sama dengan panjang dari kunci awal.

B. Pembangkitan Kunci

Kunci yang akan dibangkitkan ada tiga macam kunci, yaitu kunci awal, kunci akhir dan kunci *round*. Kunci ini membutuhkan masukan dari kunci eksternal.

Panjang dari kunci eksternal yang dibutuhkan telah dibahas pada sub bab Panjang Kunci. Ada tiga buah kondisi saat memasukkan kunci yaitu, kunci memiliki panjang yang sama dengan yang dibutuhkan, kunci memiliki panjang yang lebih dari pada kunci yang dibutuhkan dan kunci memiliki panjang yang kurang dari panjang kunci yang dibutuhkan.

Jika kunci eksternal memiliki panjang yang lebih dari pada kunci yang dibutuhkan maka akan dilakukan XOR. Kunci akan dibagi menjadi *byte*-*byte*. Setiap *byte* akan dikelompokkan berdasarkan hasil modulo dari posisinya. Setelah semua telah selesai dikumpulkan menjadi kelompok, setiap kelompok akan dilakukan XOR sehingga setiap kelompok menghasilkan satu *byte*.

Jika kunci eksternal memiliki panjang yang kurang dari maka akan dilakukan ekspansi. Cara melakukan ekspansi adalah mengganti kunci menjadi *byte*. Dari semuanya *byte* dilakukan perkalian. Kita akan melakukan *pseudorandom* seperti standar bahasa C yang *seed*-nya merupakan hasil perkalian sebelumnya. Setelah itu kita akan menghasilkan bilangan random. Bilangan tersebut akan dilakukan modulo 255 agar menghasilkan bilangan satu *byte*. Proses ini akan diulang sampai panjang kunci memiliki panjang yang sama dengan kunci yang dibutuhkan.

Jika panjang kunci sudah sesuai dengan kunci yang dibutuhkan akan dilakukan pengelompokan kunci. Cara mengelompokkan kunci dengan menggunakan modulo. Langkah pertama yang dilakukan adalah mengelompokkan menjadi *byte*. Setelah itu dilakukan modulo dengan jumlah iterasi ditambah dengan 2. Jika hasil modulo 0 maka akan menjadi kunci awal. Jika hasil modulo 1-jumlah iterasi akan menjadi kunci *round* yang berkoresponden dengan hasil modulo. Jika hasil modulo melebihi jumlah iterasi akan menjadi kunci akhir.

C. Tahap Awal

Pada tahap awal akan terdapat 3 buah aksi. Tahap ini merupakan tahap sebelum memasuki jaringan Feistel. Pada tahap ini kunci yang digunakan adalah kunci awal.

Aksi yang pertama yang dilakukan adalah substitusi. Substitusi ini akan menggunakan s-box yang telah dilampirkan pada makalah ini.

S-box yang digunakan adalah s-box yang berukuran 16x16. S-box ini dibangkitkan secara *random*, belum ada penelitian khusus tentang s-box ini. S-box ini bertujuan untuk mendapatkan sifat *confusion*.

Setelah blok selesai dengan substitusi blok akan memasuki permutasi. Permutasi dengan menggunakan p-box yang telah dilampirkan dalam makalah ini. P-box berukuran 4x8 yang dibangkitkan secara random. Belum ada penelitian khusus tentang p-box yang dibangkitkan.

Hal yang akan dilakukan pada akhir tahap ini adalah melakukan XOR. Block akan dibentuk menjadi 4x8. Baris pertama akan dilakukan XOR dengan kunci. Setelah itu akan menghasilkan sebuah baris, baris ini akan dilakukan XOR dengan baris kedua. Hasil dari baris kedua akan dilakukan XOR dengan baris ketiga. Hal ini dilakukan sampai akhir baris. Melakukan XOR secara beruntun digunakan untuk mendapatkan sifat *diffusion*.

C. Tahap Iterasi (Jaringan Feistel)

Pada tahap ini akan ada beberapa *iterasi*. Pada tahap ini menggunakan jaringan Feistel. Banyaknya iterasi pada tahap ini bergantung pada panjang nya kunci yang ditentukan.

Pada tahap ini kunci yang digunakan adalah *round key*. Pembangkitan dari *round key* telah dibahas pada sub bab pembangkitan kunci.

Fungsi yang digunakan pada jaringan Feistel ini hanyalah XOR dengan kunci. Cara melakukan XOR seperti pada tahap awal untuk mendapatkan sifat *diffusion*. Cara melakukan XOR terinspirasi dari domino, yang bergantung pada nilai sebelumnya.

C. Tahap Akhir

Pada tahap ini terdapat tiga aksi yang sama dengan tahap pertama. Aksi yang ada pada tahap ini adalah substitusi, permutasi dan XOR.

Substitusi yang dilakukan pada tahap ini menggunakan s-box akhir yang telah dilampirkan pada makalah ini. S-box ini dibangkitkan secara random. Belum ada penelitian khusus tentang s-box.

Permutasi dilakukan dengan menggunakan p-box akhir. P-box yang digunakan untuk permutasi ada pada lampiran di makalah ini. Belum ada penelitian khusus tentang p-box ini.

Untuk operasi XOR menggunakan kunci akhir yang telah dibangkit dari kunci eksternal. Pembangkitan kunci akhir telah dibahas pada sub bab pembangkitan kunci.

IV. PENGUJIAN

Pada tahap ini akan dilakukan tiga kali percobaan. Percobaan yang pertama dengan menggunakan panjang kunci 40 *byte* dan panjang *plaintext* 32 *byte*. Pada percobaan kali ini dilakukan 8 kali iterasi jaringan Feistel.

Pada percobaan kedua akan memiliki setting yang sama dengan percobaan pertama. Tetapi nilai kunci dari percobaan kedua diganti satu *bit* untuk melihat apakah algoritma ini

memiliki sifat *diffusion*.

Pada percobaan ketiga akan memiliki setting yang sama juga dengan percobaan pertama dan kedua. Tetapi nilai *plaintext* akan diganti satu *bit*. Hal ini digunakan untuk melihat algoritma ini memiliki sifat *diffusion*.

A. Percobaan Pertama

Pada percobaan ini akan digunakan kunci: ini adalah kunci percobaan untuk dominos. Sedangkan *plaintext* yang digunakan adalah dominos memiliki sifat diffusion.

Kunci	ini adalah kunci percobaan untuk dominos
Kunci(Hex)	69 6e 69 20 61 64 61 6c 61 68 20 6b 75 6e 63 69 20 70 65 72 63 6f 62 61 61 6e 20 75 6e 74 75 6b 20 64 6f 6d 69 6e 6f 73
Plaintext	dominos memiliki sifat diffusion
Plaintext(Hex)	64 6f 6d 69 6e 6f 73 20 6d 65 6d 69 6c 69 6b 69 20 73 69 66 61 74 20 64 69 66 66 75 73 69 6f 6e
Cipher(Hex)	ee 13 07 b9 64 64 bb 39 d2 82 49 1d f8 f3 77 51 55 f8 0f 31 79 2b a3 f1 10 55 af a3 8a e4 ad 8c

B. Percobaan Kedua

Pada percobaan kali ini akan diganti satu *bit* pada kunci.

Kunci Lama	ini adalah kunci percobaan untuk dominos
Kunci Lama(Hex)	69 6e 69 20 61 64 61 6c 61 68 20 6b 75 6e 63 69 20 70 65 72 63 6f 62 61 61 6e 20 75 6e 74 75 6b 20 64 6f 6d 69 6e 6f 73
Kunci Baru	jni adalah kunci percobaan untuk dominos
Kunci Baru(Hex)	6a 6e 69 20 61 64 61 6c 61 68 20 6b 75 6e 63 69 20 70 65 72 63 6f 62 61 61 6e 20 75 6e 74 75 6b 20 64 6f 6d 69 6e 6f 73
Plaintext	dominos memiliki sifat diffusion
Plaintext(Hex)	64 6f 6d 69 6e 6f 73 20 6d 65 6d 69 6c 69 6b 69 20 73 69 66 61 74 20 64 69 66 66 75 73 69 6f 6e
Cipher Lama(Hex)	ee 13 07 b9 64 64 bb 39 d2 82 49 1d f8 f3 77 51 55 f8 0f 31 79 2b a3 f1 10 55 af a3 8a e4 ad 8c
Cipher Baru(Hex)	d5 28 8d ea 37 37 e8 6a d2 82 49 1d f8 f3 77 51 b1 54 a3 9d bb e9 61
	33 10 f4 0e e8 c1 af e6 c7

C. Percobaan Ketiga

Pada percobaan kali ini akan diganti satu bit pada *plaintext*.

Kunci	ini adalah kunci percobaan untuk dominos
Kunci(Hex)	69 6e 69 20 61 64 61 6c 61 68 20 6b 75 6e 63 69 20 70 65 72 63 6f 62 61 61 6e 20 75 6e 74 75 6b 20 64 6f 6d 69 6e 6f 73
Plaintext Lama	dominos memiliki sifat diffusion
Plaintext Lama(Hex)	64 6f 6d 69 6e 6f 73 20 6d 65 6d 69 6c 69 6b 69 20 73 69 66 61 74 20 64 69 66 66 75 73 69 6f 6e
Plaintext Baru	eominos memiliki sifat diffusion
Plaintext Baru(Hex)	65 6f 6d 69 6e 6f 73 20 6d 65 6d 69 6c 69 6b 69 20 73 69 66 61 74 20 64 69 66 66 75 73 69 6f 6e
Cipher Lama(Hex)	ee 13 07 b9 64 64 bb 39 d2 82 49 1d f8 f3 77 51 55 f8 0f 31 79 2b a3 f1 10 55 af a3 8a e4 ad 8c
Cipher Baru(Hex)	ee 13 10 ae 73 73 ac 2e d2 82 49 1d f8 f3 77 51 e9 e5 12 2c ec be 36 64 10 55 af 02 2b 45 0c 2d

V. ANALISIS

Terdapat beberapa serangan yang dapat terjadi dalam *ciphertext*. Contoh-contoh serangan adalah analisis statistika, *brute force*, dan penggantian *ciphertext*.

A. Analisis Statistika

Pada analisis statistik *cryptanalysis* akan membuat statistika dari *ciphertext*. Setelah itu *cryptanalysis* akan mencari keterhubungan antara *ciphertext* dan *plaintext*. Cara ini sangat efektif untuk algoritma kriptografi yang tidak memiliki sifat *confusion*.

Banyak algoritma kriptografi yang dapat diserang menggunakan cara ini salah satu contohnya adalah Vigenere Cipher. Hal ini disebabkan karena Vigenere Cipher tidak memiliki sifat *confusion*.

Serangan analisis statistik tidak akan bisa bila dilakukan kepada Dominos. Hal ini disebabkan karena Dominos memiliki sifat *confusion*. Sifat *confusion* dari Dominos dapat

dilihat pada percobaan pertama bahwa secara statistik seluruh *plaintext* beberapa memiliki nilai yang sama, tetapi jika kita melihat statistik dari *ciphertext* semuanya hanya memiliki kemunculan satu dan dua saja, atau bisa disebut bahwa statistik dari *ciphertext* yang dihasilkan oleh Dominos bersifat datar. Oleh karena statistik *ciphertext* yang dihasilkan oleh Dominos datar maka tidak bisa dilakukan serangan secara analisis statistik.

B. Brute Force

Serangan *brute force* adalah serangan yang mencoba seluruh kemungkinan kunci. Kunci minimal dari Dominos adalah 40 *byte* atau 320 *bit*.

Dengan ruang kunci sebanyak itu akan sulit untuk melakukan *brute force*. Karena terdapat 2^320 kemungkinan kunci yang perlu dicoba.

Komputer yang paling cepat saat ini adalah yang memiliki kecepatan 10^9 atau setara dengan 2^30. Jika dalam satu detik komputer dapat mencoba 2^30 maka waktu satu komputer untuk mencoba semua kunci adalah 2^290 detik atau setara dengan 6.3038236 × 10^79 years. Jadi mustahil untuk melakukan penyerangan secara *brute force*.

C. Pergantian Ciphertext

Pergantian *ciphertext* tidak membutuhkan kunci. *Cryptanalysis* hanya membutuhkan koresponden *plaintext* dengan *ciphertext*.

Serangan seperti ini dapat berhasil jika algoritma kriptografi tidak memiliki sifat *diffusion*. Jika algoritma kriptografi memiliki sifat *diffusion* maka akan sulit untuk mencari koresponden *plaintext* dan *ciphertext*.

Dominos tidak dapat diserang dengan menggunakan metode ini karena Dominos memiliki sifat *diffusion*. Sifat *diffusion* yang dimiliki oleh Dominos dapat dilihat pada percobaan kedua dan percobaan yang ketiga.

VI. Kesimpulan

Disini disimpulkan bahwa Dominos memiliki sifat *confusion* yang dapat dilihat dari percobaan pertama. Selain sifat *confusion* Dominos juga memiliki sifat *diffusion* yang dapat dilihat dari percobaan kedua dan ketiga.

Dominos cepat bila diimplementasikan pada *hardware* karena operasi yang dilakukan Dominos sebagian besar berupa permutasi, substitusi dan XOR.

References

[1] https://www.tutorialspoint.com/cryptography/modern_cryptography.htm diakses pada tanggal 12 Maret 2019.

[2] https://www.tutorialspoint.com/cryptography/origin_of_cryptography.htm diakses pada tanggal 12 Maret 2019.

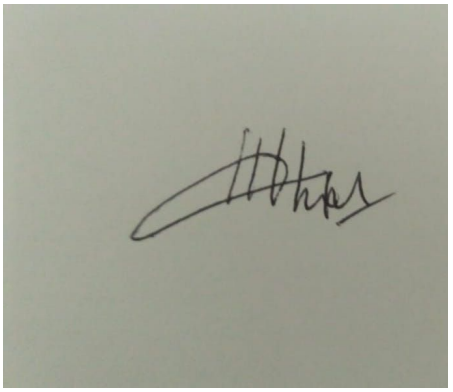
[3] https://www.tutorialspoint.com/cryptography/feistel_block_cipher.htm diakses pada tanggal 12 Maret 2019.

[4] Munir, Rinaldi. 2019. Slide Kuliah IF4020 Kriptografi: Kriptografi Modern.

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, 13 Maret 2019



Ahmad Fajar Prasetiyo (13514053)

Lampiran

S-Box Awal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	68	9B	62	5C	01	39	96	09	42	8A	F4	F3	98	C7	DB	B3
1	40	D3	8C	A6	1B	7F	AF	30	6E	05	41	A1	56	24	23	73
2	CE	04	4F	DA	99	B5	88	C2	DC	54	7A	E3	BA	7E	A7	72
3	AA	07	C6	FB	18	EB	4A	BE	61	F0	0A	F8	60	4E	65	FC
4	EE	16	3E	77	3A	34	70	3D	95	FF	BB	FE	B4	13	F6	44
5	FA	F9	AC	4D	97	21	DD	F1	C3	67	12	32	CD	78	20	CB
6	84	1D	FD	71	38	36	45	D6	37	4C	85	AD	E0	86	83	6F
7	9E	D1	1E	A5	43	5B	B8	5E	1F	22	59	94	2C	8B	0B	B0
8	53	4B	6A	2F	92	90	79	26	57	3B	D0	8D	6D	02	E4	46
9	B2	10	A9	64	63	CC	1C	06	66	51	76	49	7C	7D	D8	BF
A	48	7B	75	B9	B7	ED	5D	87	25	91	F7	50	47	AB	27	19
B	F2	0E	93	E8	D9	33	3C	55	BD	C8	11	9C	E9	CF	35	F5
C	8F	CA	2B	0D	D4	52	1A	2A	C1	A3	D2	AE	6B	5A	B1	2D
D	C9	EA	C5	EF	5F	15	3F	17	EC	A2	81	E2	2E	A8	BC	0F
E	B6	31	9F	89	29	08	69	80	58	A4	C4	74	DE	28	E7	E6
F	8E	D5	E5	9A	82	0C	9D	C0	03	A0	E1	D7	6C	14	00	DF

S-Box Akhir

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F1	52	F5	03	82	E3	DE	D3	51	53	87	2D	CD	14	A5	9F
1	66	B1	17	D2	91	95	D4	DD	E2	68	A9	96	2A	43	65	81
2	F7	A2	50	CF	C8	F4	63	07	1C	05	6D	A6	70	75	31	20
3	25	57	4A	92	6F	CB	0E	E8	10	8D	84	AE	1D	F8	EB	09
4	98	F9	7E	B6	B4	BA	76	39	34	6A	BB	F0	BF	7D	3A	B8
5	A0	5D	DC	3F	FD	B0	4C	AB	C4	E9	DA	A4	6E	37	8A	5F
6	E4	79	8C	45	13	1F	D9	DB	32	C1	56	F6	6C	C7	5B	3E
7	A3	94	A7	B7	86	A8	CE	BD	D7	83	02	AC	F3	15	1E	D0

8	46	89	AF	08	FF	C5	21	30	9D	ED	D1	24	E0	22	D5	42
9	78	BC	77	16	E7	7B	D6	EC	4D	FC	3B	0A	5E	06	EE	99
A	5C	B3	18	B5	04	36	2F	73	71	55	8B	74	41	44	72	5A
B	67	C3	2E	BE	3D	DF	58	49	01	E1	1B	27	97	9E	85	7F
C	64	88	3C	29	0C	C2	8F	E6	93	E5	AD	7A	61	8E	4F	0F
D	60	A1	4E	00	AA	62	FA	EF	6B	19	D8	C6	11	40	EA	9B
E	33	C0	0D	59	23	B9	12	69	B2	80	2C	CC	26	4B	90	48
F	7C	F2	54	CA	2B	FE	9C	0B	C9	28	47	9A	38	1A	35	FB

P-Box Awal

	0	1	2	3	4	5	6	7
0	03	32	17	30	05	31	14	13
1	04	25	21	37	27	00	24	22
2	36	15	34	16	01	26	10	33
3	35	11	12	23	20	06	07	02

P-Box Akhir

	0	1	2	3	4	5	6	7
0	31	00	03	24	02	20	21	33
1	13	06	15	10	32	30	22	37
2	05	01	04	11	36	34	26	27
3	12	35	16	23	07	25	14	17