

Tugas EI 7010
Keamanan Sistem Informasi

PENYANDIAN DATA DENGAN
ALGORITMA KRIPTOGRAFI NOEKEON

I Made Ari Jaya N

232 02 002



Program Pascasarjana Teknologi Informasi
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Bandung
2003

ABSTRAK

Keamanan data merupakan hal yang sangat penting dalam menjaga kerahasiaan informasi terutama yang berisi informasi sensitif yang hanya boleh diketahui isinya oleh pihak yang berhak saja, apalagi jika pengirimannya dilakukan melalui jaringan publik, apabila data tersebut tidak diamankan terlebih dahulu, akan sangat mudah disadap dan diketahui isi informasinya oleh pihak-pihak yang tidak berhak.

Salah satu cara yang digunakan untuk pengamanan data adalah menggunakan sistem kriptografi yaitu dengan menyandikan isi informasi (*plaintext*) tersebut menjadi isi yang tidak dipahami melalui proses enkripsi (*encipher*), dan untuk memperoleh kembali informasi yang asli, dilakukan proses dekripsi (*decipher*), disertai dengan menggunakan kunci yang benar. Namun sejalan dengan perkembangan ilmu penyandian atau kriptografi, usaha-usaha untuk memperoleh kunci tersebut dapat dilakukan oleh siapa saja, termasuk pihak yang tidak sah untuk memiliki informasi tersebut. Oleh karena itu, penelitian tentang kriptografi akan selalu berkembang untuk memperoleh algoritma kriptografi yang makin kuat, sehingga usaha-usaha untuk memecah kode kriptografi secara tidak sah menjadi lebih sulit.

Melalui tugas ini akan dibahas mengenai algoritma kriptografi NOEKEON, yang diajukan kepada proyek Nessie (*New European Schemes for Signatures, Integrity, and Encryption*) dalam rangka memperkuat posisi industri negara-negara Eropa di bidang kriptografi.

Kata Kunci: Kriptografi, *Block Cipher*, Noekeon.

DAFTAR ISI

LEMBAR JUDUL

ABSTRAK

DAFTAR ISI

i

BAB I	DASAR TEORI	6
I.1	Pengertian Kriptografi	6
I.2	Dasar Matematis	6
I.3	Teknik Kriptografi	7
I.3.1	Symmetric-key	7
I.3.2	Asymmetric-key	7
I.4	Kriptografi Blok Cipher	7
I.4.1	Konsep Dasar	7
I.4.1.1	Cipher Berulang	8
I.4.1.2	Cipher Fiestel	8
I.4.1.3	Avalanche	8
I.4.2	Mode Operasi	8
I.4.3	Kunci Lemah dan Kunci Setengah Lemah	9
I.5	Enkripsi dan Dekripsi	9
I.5.1	Enkripsi	9
I.5.2	Padding	10
I.5.3	Dekripsi	10
I.6	Algoritma Kriptografi Noekeon	10
I.6.1	Penjadwalan Kunci	10
I.6.2	State	11
I.6.3	Theta	11
I.6.4	Shift Offset	11
I.6.5	Gamma	12
I.6.6	Round Constant	12
I.6.7	Enkripsi dan Dekripsi	12
I.6.7.1	Tahap Enkripsi	12
I.6.7.2	Tahap Dekripsi	12

BAB II	PERANCANGAN SIMULASI ALGORITMA KRIPTOGRAFI NOEKEON	13
II.1	Analisa Sistem	13
II.2	Batasan Perancangan Program Simulasi	13
II.3	Perancangan Sistem	13
II.3.1	Model Simulasi	13
II.3.1.1	Diagram Konteks	14
II.3.1.2	Diagram Aliran Data	14
II.3.1.2.1	Proses Enkripsi	14
II.3.1.2.2	Proses Dekripsi	15
II.3.1.3	Kamus Data	16
II.3.1.4	Spesifikasi Proses	16
II.4	Implementasi Sistem	18
BAB III	HASIL PERANCANGAN DAN ANALISA UNJUK KERJA ALGORITMA NOEKEON	19
III.1	Hasil Simulasi	19
III.1.1	Proses Enkripsi	19
III.1.2	Proses Dekripsi	21
III.2	Proses File	22
III.3	Analisa Unjuk Kerja Algoritma Noekeon	23
III.3.1	Struktur Cipher	23
III.3.2	Proses Enkripsi dan Dekripsi	24
III.3.3	Penjadwalan Kunci	24
III.3.4	Sifat Simetris, Kunci Lemah dan Kunci Setengah Lemah	24
III.3.5	Efek Avalanche	26
III.3.6	Kesalahan Propagasi	26
III.3.7	Kekuatan Terhadap Jenis Serangan Brute-Force	27

DAFTAR PUSTAKA

LAMPIRAN A : Hasil Proses Enkripsi Dan Dekripsi

LAMPIRAN B : Hasil Pengamatan Efek Avalanche

LAMPIRAN C : Hasil percobaan kesalahan Propagasi

BAB I

DASAR TEORI

I.1 Pengertian Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti keabsahan, integritas data, serta autentikasi data. Kriptografi tidak berarti hanya memberikan keamanan informasi saja, namun lebih ke arah teknik-tekniknya.

Ada empat tujuan mendasar dari ilmu kriptografi ini yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. Autentikasi, adalah berhubungan dengan identifikasi, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. Non-repudiasi, yang berarti begitu pesan terkirim, maka tidak akan dapat dibatalkan.

I.2 Dasar Matematis

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen *plaintext* dan yang berisi elemen *ciphertext*. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. Apabila elemen-elemen *plaintext* dinotasikan dengan P , elemen-elemen *ciphertext* dinotasikan dengan C , sedang untuk proses enkripsi dinotasikan dengan E , dekripsi dengan notasi D , maka secara matematis proses kriptografi dapat dinyatakan sebagai berikut :

$$\text{Enkripsi : } E(P) = C \quad (1.1)$$

$$\text{Dekripsi : } D(C) = P \text{ atau } D(E(P)) = P \quad (1.2)$$

Pada skema enkripsi konvensional atau *symmetric-key*, digunakan sebuah kunci untuk melakukan proses enkripsi dan dekripsinya. Kunci tersebut dinotasikan dengan K . Sehingga proses kriptografinya adalah sebagai berikut:

$$\text{Enkripsi : } E_K(P) = C \quad (1.3)$$

$$\text{Dekripsi : } D_K(C) = P \text{ atau } D_K(E_K(P)) = P \quad (1.4)$$

Sedangkan pada sistem *asymmetric-key* digunakan kunci umum (*public key*) untuk enkripsi dan kunci pribadi (*private key*) untuk proses dekripsinya. Sehingga kedua proses tersebut dinyatakan dengan :

$$\text{Enkripsi : } E_{PK}(P) = C \quad (1.5)$$

$$\text{Dekripsi : } D_{SK}(C) = P \text{ atau } D_{SK}(E_{PK}(P)) = P \quad (1.6)$$

I.3 Teknik Kriptografi

Secara umum dikenal dua teknik dalam kriptografi, yaitu *symmetric-key* dan *asymmetric-key* (*public-key*).

II.3.1 *Symmetric-key*

Skema enkripsi akan disebut *symmetric-key* apabila pasangan kunci untuk proses enkripsi dan dekripsinya adalah sama. Pada skema enkripsi *symmetric-key* dibedakan menjadi dua kelas, yaitu *block-cipher* dan *stream-cipher*.

Block-cipher adalah skema enkripsi yang akan membagi-bagi *plaintext* yang akan dikirimkan menjadi string-string (disebut blok) dengan panjang t , dan mengenkripsinya per-blok. Pada umumnya, *block-cipher* memproses *plaintext* dengan blok yang relatif panjang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci. Sedangkan skema *stream-cipher* pada dasarnya juga *block-cipher*, hanya dengan panjang bloknnya adalah satu bit.

I.3.2 *Asymmetric-key*

Skema ini adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Skema ini disebut juga sebagai sistem kriptografi *Public-key* karena kunci untuk enkripsi dibuat secara umum (*public-key*) atau dapat diketahui siapa saja, tapi untuk proses dekripsinya dibuat satu hanya oleh yang berwenang untuk mendekripsinya, disebut *private-key*.

Keuntungan skema model ini, untuk berkorespondensi secara rahasia dengan banyak pihak tidak diperlukan kunci rahasia sebanyak jumlah pihak tersebut, cukup membuat dua buah kunci, yaitu *public-key* bagi para koresponden untuk mengenkripsi pesan, dan *private-key* untuk mendekripsi pesan. Berbeda dengan skema *symmetric-key*, jumlah kunci yang dibuat adalah sebanyak jumlah pihak yang diajak berkorespondensi.

I.4 Kriptografi blok cipher

I.4.1 Konsep Dasar

Blok cipher merupakan sebuah fungsi yang memetakan n -bit blok-blok *plaintext* ke n -bit blok-blok *ciphertext*, dengan n adalah panjang blok. Blok cipher umumnya memproses *plaintext* ke dalam blok-blok yang cukup besar ($n \geq 64$). Beberapa teknik blok cipher modern diantaranya :

1.4.1.1 Cipher Berulang

Pada teknik cipher berulang (*iterated cipher*), blok *plaintext* mengalami pengulangan fungsi transformasi beberapa kali untuk mendapatkan blok *ciphertext*. Fungsi transformasi pada umumnya merupakan gabungan proses substitusi, permutasi, kompresi, atau ekspansi terhadap blok *plaintext*. Sebuah kunci pada setiap putaran (*round key*) akan dikombinasikan dengan *plaintext*. Parameter dalam cipher ini adalah jumlah putaran r , besar blok n , dan besar kunci k . Sub-kunci K_i pada setiap putaran diperoleh dari penurunan kunci input K .

1.4.1.2 Feistel Cipher

Feistel cipher beroperasi terhadap panjang blok data tetap sepanjang n (genap), kemudian membagi 2 blok tersebut dengan panjang masing-masing $n/2$, yang dinotasikan dengan L dan R . Feistel cipher menerapkan metode cipher berulang dengan masukan pada putaran ke- i yang didapat dari keluaran sebelumnya. Secara matematis dapat dinyatakan sebagai berikut:

$$L_i = R_{i-1} \quad (1.7)$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i); \quad i = 1, 2, 3, \dots, r \quad (1.8)$$

K_i adalah kunci untuk putaran ke- i dan f adalah fungsi transformasi.

Blok *plaintext* adalah gabungan L dan R awal atau secara formal *plaintext* dinyatakan dengan (L_0, R_0) . Sedangkan blok *ciphertext* didapatkan dari L dan R hasil putaran terakhir setelah terlebih dahulu dipertukarkan atau secara formal *ciphertext* dinyatakan dengan (R_r, L_r) .

1.4.1.3 Avalanche

Pada blok cipher, perubahan satu buah bit dapat menghasilkan perubahan lebih dari satu bit setelah satu putaran, lebih banyak lagi bit berubah untuk putaran berikutnya. Hasil perubahan tersebut dinamakan sebagai *avalanche effect*. Sebuah algoritma kriptografi memenuhi kriteria *avalanche effect* apabila satu buah bit input mengalami perubahan, maka probabilitas semua bit berubah adalah setengahnya. *Avalanche effect* merupakan salah satu karakteristik yang menjadi acuan untuk menentukan baik atau tidaknya sebuah algoritma kriptografi.

1.4.2 Mode Operasi

Ada beberapa mode operasi yang digunakan dalam kriptografi. Beberapa diantaranya adalah :

1. *Electronic codebook* (ECB)

Pada mode ini, blok-blok *plaintext* (x) yang identik (menggunakan kunci yang sama) akan menghasilkan *ciphertext* (c) yang identik pula. Secara matematis dinyatakan :

$$\text{Enkripsi : } c_j \leftarrow E_K(x_j); \quad 1 \leq j \leq t \quad (1.9)$$

$$\text{Dekripsi : } x_j \leftarrow E_K^{-1}(c_j); \quad 1 \leq j \leq t \quad (1.10)$$

2. Cipher-block chaining (CBC)

Pada prosesnya, mode ini melibatkan penggunaan *initializing vector* (IV), yang menyebabkan blok-blok *ciphertext* yang identik apabila dienkripsi menggunakan kunci dan IV yang sama. Berubahnya IV , kunci, atau blok *plaintext* pertama akan menghasilkan *ciphertext* yang berbeda. Secara matematis dinyatakan :

$$\text{Enkripsi : } c_0 \leftarrow IV, \text{ untuk } 1 \leq j \leq t, c_j \leftarrow E_K(c_{j-1} \oplus x_j). \quad (1.11)$$

$$\text{Dekripsi : } c_0 \leftarrow IV, \text{ untuk } 1 \leq j \leq t, x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j). \quad (1.12)$$

3. Cipher feedback (CFB)

Jika pada mode CBC, *plaintext* sebesar n -bit diproses dalam sekali waktu (menggunakan sebuah n -bit cipher blok). Beberapa aplikasi mengharuskan r -bit *plaintext* untuk dienkripsi terlebih dahulu dan ditransmisikan bebas delay, untuk $r < n$ (biasanya $r = 1$ atau $r = 8$). Dalam kasus ini CFB digunakan. Dalam mode ini juga melibatkan penggunaan *initializing vector*.

4. Output feedback (OFB)

Mode operasi ini digunakan apabila kesalahan propagasi sama sekali harus dihindari. Hampir mirip dengan CFB, dan juga memungkinkan enkripsi menggunakan besar blok yang bervariasi.

I.4.3 Kunci Lemah dan Kunci Setengah Lemah

Dalam kriptografi dikenal dengan istilah kunci lemah (*weak-key*) dan kunci setengah lemah (*semi weak-key*). Kunci Lemah adalah kunci yang apabila mengenkripsi suatu *plaintext* kemudian dienkripsi lagi menggunakan kunci yang sama maka *ciphertext*-nya adalah *plaintext* itu sendiri. Sedangkan yang disebut kunci setengah lemah adalah sepasang kunci yang mempunyai sifat jika sebuah *plaintext* dienkripsi dengan suatu kunci, akan dapat didekripsi dengan kunci yang lain.

I.5 Enkripsi dan Dekripsi

I.5.1 Enkripsi

Proses utama dalam suatu algoritma kriptografi adalah enkripsi dan dekripsi. Enkripsi merubah sebuah *plaintext* ke dalam bentuk *ciphertext*. Pada mode ECB (*Electronic Codebook*), sebuah blok pada *plaintext* dienkripsi kedalam sebuah blok *ciphertext* dengan panjang blok yang sama. Secara matematis, proses enkripsi telah diberikan pada persamaan (1.3).

Blok cipher memiliki sifat bahwa setiap blok harus memiliki panjang yang sama (misal 128 bit). Namun apabila pesan yang dienkripsi memiliki panjang blok terakhir yang tidak tepat 128 bit, maka diperlukan mekanisme *padding* yaitu penambahan bit-bit *dummies* untuk menggenapi menjadi panjang blok yang sesuai, biasanya *padding* dilakukan pada blok terakhir *plaintext*.

I.5.2 Padding

Padding pada blok terakhir bisa dilakukan dengan berbagai macam cara, misal penambahan bit-bit tertentu. Salah satu contoh penerapan *padding* dengan cara menambahkan jumlah total *padding* sebagai byte terakhir pada blok terakhir *plaintext*. Misal panjang blok adalah 128 bit (16 byte), dan pada blok terakhir terdiri dari 88 bit (11 byte). Sehingga jumlah *padding* yang diperlukan adalah 5 byte, yaitu dengan menambahkan angka nol sebanyak 4 byte, kemudian menambahkan angka 5 sebanyak satu byte. Cara lain dapat menggunakan penambahan karakter *end-of-file* pada byte terakhir lalu diberi *padding* setelahnya.

I.5.3 Dekripsi

Dekripsi merupakan proses kebalikan dari proses enkripsi, merubah *ciphertext* kembali kedalam bentuk *plaintext*. Proses dekripsi telah diberikan pada persamaan (1.4). Untuk menghilangkan *padding* yang diberikan saat proses enkripsi, dilakukan berdasarkan informasi jumlah *padding* yaitu angka pada byte terakhir.

I.6 Algoritma Kriptografi Noekeon

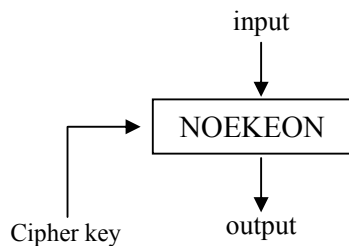
NOEKEON merupakan cipher blok berulang dengan panjang blok dan panjang kuncinya masing-masing 128 bit, yang terdiri dari aplikasi transformasi *round* sederhana yang berulang, diikuti dengan sebuah transformasi output.

NOEKEON memiliki 16 putaran (N_r) iterasi, dalam setiap putarannya dilakukan empat buah transformasi yaitu *theta*, *Shift offset* yang terdiri dari dua buah transformasi P_{i1} dan P_{i2} , dan *gamma*.

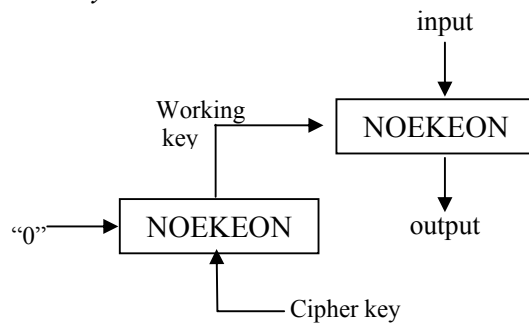
I.6.1 Penjadwalan Kunci

Penjadwalan kunci dilakukan dengan mengkonversi kunci utama (*cipher key*) 128-bit menjadi sebuah *working-key* 128-bit. Karena sifat NOEKEON yang simetri, maka setiap *round*-nya, menggunakan *working-key* yang sama.

Dalam Noekeon, ada mode saat penjadwalan kunci tidak dilakukan, disebut mode *direct-key*, artinya *working-key* adalah *cipher-key* itu sendiri. Mode yang kedua adalah mode *indirect-key* yang melakukan proses penjadwalan kunci untuk mengeliminasi pola serangan *related-key*.



Gambar 2.1 Mode Direct-key [4]



Gambar 2.2 Mode indirect-key [4]

Pada mode *indirect-key*, sebelum kunci diaplikasikan terhadap pesan pada operasi theta, kunci dirubah dahulu menjadi sebuah kunci yang lain dengan tetap menggunakan fungsi yang sama dalam Noekeon. Baru kemudian, kunci tersebut diaplikasikan terhadap pesan pada operasi theta dan seterusnya sebanyak 16 putaran.

I.6.2 State

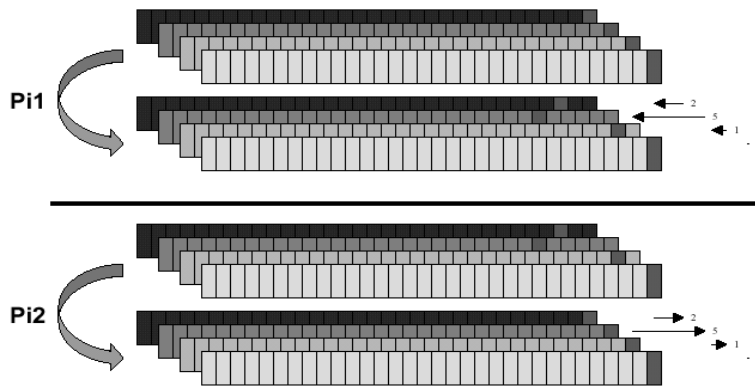
Setiap transformasi *round* dioperasikan pada sebuah state yang terdiri dari empat buah 32-bit word yaitu $a[0]$ sampai $a[3]$.

I.6.3 Theta

Theta adalah pemetaan linear yang menggunakan *working-key* k dan dilakukan operasi pada *state* a . Pada tahap ini, terdiri dari 12 langkah operasi. Langkah yang pertama adalah operasi *xor* antara word a_0 dan a_2 . Selanjutnya, hasil operasi itu dilakukan dua buah pergeseran bit, yaitu kekanan sebanyak 8 bit, dan kekiri sebanyak 8 bit, lalu hasil pergeseran tersebut di-*xor* dengan hasil langkah pertama. Langkah berikutnya, yaitu proses perubahan word a_1 dan a_3 , dengan meng-*xor*-kan a_1 dengan langkah kedua. Setelah itu, keempat word dari *plaintext* masing-masing di-*xor*-kan dengan keempat buah word kunci, dan akan menghasilkan word $[a_0, a_1, a_2, a_3]$ baru. Dari word baru tersebut, a_1 dan a_3 , di-*xor*, dan hasilnya dilakukan dua buah pergeseran 8 bit masing-masing kekanan dan kekiri. Terakhir, a_0 dan a_2 masing-masing akan di-*xor* dengan hasil pergeseran tersebut. Dari proses *Theta* ini akan dihasilkan word $[a_0, a_1, a_2, a_3]$ yang baru, untuk dilakukan proses selanjutnya.

I.6.4 Shift Offset

Pergeseran ini terdiri dari dua yaitu Pi_1 dan Pi_2 yang keduanya saling berkebalikan. Masing-masing terdiri dari empat offset dengan modulo 8 yang berbeda (a_0, a_1, a_2, a_3), untuk a_0 tidak dilakukan pergeseran. Secara jelas digambarkan sebagai berikut:



Gambar 2.3 Visualisasi Shift Offset [3]

Pada gambar diatas, susunan empat buah word mulai teratas adalah a_3 , kemudian a_2, a_1 , dan a_0 . Pada proses Pi_1 , a_1 digeser 1 bit ke kiri, a_2 5 bit ke kiri, dan a_3 2

bit ke kiri. Pada proses Pi_2 , merupakan kebalikan Pi_1 , jadi a_1 digeser 1 bit ke kanan, a_2 5 bit ke kanan, dan a_3 2 bit ke kanan. Sedang a_0 tidak terjadi pergeseran.

I.6.5 *Gamma*

Gamma merupakan pemetaan involusi non-linear. Transformasi non-linear dilakukan dengan tiga langkah yaitu:

- Transformasi non-linear sederhana
- Transformasi linear sederhana
- Transformasi non-linear sederhana

Pada tahap ini, Noekeon akan menghasilkan word-word a_0 , a_1 , dan a_2 yang baru. *Gamma* akan menghasilkan *substitution-box (S-box)* bagi Noekeon. S-box ini berupa word terdiri dari 4 buah word 32 bit yang masing-masing box-nya adalah 4 bit setiap word $[a_0, a_1, a_2, a_3]$ secara berurut.

I.6.6 *Round Constant*

Untuk menghilangkan sifat kelinearan pada setiap putaran Noekeon, dilakukan operasi *round constants*. Pada dasarnya operasi ini adalah sebuah shift register (mod $0x80$, untuk $state[0]$) yang dilakukan terhadap 8 bit terbawah dalam 32-bit word *state* awal.

I.6.7 Enkripsi dan Dekripsi

I.6.7.1 Tahap Enkripsi

Tahap ini diawali dengan adanya masukan dari pemakai berupa teks yaitu untuk *plaintext* dan untuk kuncinya. Teks yang masih berbentuk karakter tersebut selanjutnya akan direpresentasikan kedalam bentuk deretan bit, kemudian akan dibentuk blok dengan panjang 128 bit, yang masing-masing blok untuk *plaintext* dan kuncinya akan dibagi menjadi 4 buah word masing-masing 32 bit, yaitu $[a_0, a_1, a_2, a_3]$ untuk *plaintext* dan $[k_0, k_1, k_2, k_3]$ untuk kuncinya. Bila jumlah bit dalam satu blok tersebut kurang dari 128 bit, maka akan digenapkan dengan menambahkan bit "0". Kedelapan word tersebut yang nantinya akan diproses dalam prosedur algoritma Noekeon.

I.6.7.2 Tahap Dekripsi

Keunggulan algoritma Noekeon terletak pada kesederhanaan kode program atau sirkuit perangkat kerasnya. Kode atau sirkuit yang sama, digunakan baik pada proses enkripsi dan dekripsinya, hanya penerapan pada *theta* yang berbeda. Pada proses enkripsi, *theta* adalah *theta* (k, a), sedangkan pada proses dekripsinya, menjadi *theta* (NullVektor, a). Dengan kata lain, kebalikan dari *theta* adalah *theta* itu sendiri, namun dengan pengaplikasian *null vektor* sebagai *working-key*.

BAB II

PERANCANGAN SIMULASI

ALGORITMA KRIPTOGRAFI NOEKEON

II.1 Analisa Sistem

Pada simulasi algoritma kriptografi Noekeon terdiri dari dua tahapan besar, yaitu tahap enkripsi dan tahap dekripsi, yang masing-masing, satu putarannya (*round*) terdiri dari empat prosedur, yaitu prosedur *theta*, prosedur *Pi1*, prosedur *gamma*, dan prosedur *Pi2*. Pada implementasinya, data sebagai masukan, akan dibagi-bagi menjadi beberapa blok yang masing-masing blok sepanjang 128 bit yang disebut *state*, baik sebagai *plaintext* maupun sebagai kunci, kemudian blok-blok tersebut akan dilakukan operasi dengan empat prosedur diatas tadi sehingga akan mneghasilkan *ciphertext* yang diharapkan.

Tujuan utama pada simulasi ini adalah memberikan penjelasan yang cukup baik dan mudah dipahami bagaimana proses atau langkah kriptografi algoritma Noekeon, mulai dari masukan data yang diberikan oleh pengguna, sampai terjadinya perubahan data tersebut menjadi bentuk yang telah terenkripsi, kemudian kembali lagi menjadi data semula.

II.2 Batasan Perancangan Program Simulasi

Dalam proses perancangan program simulasi, akan ada pembatasan-pembatasan masalah yang diutamakan untuk memudahkan pengamatan langkah-langkah kriptografi pada algoritma ini, yaitu :

1. Simulasi dibuat dengan menggunakan bahasa pemrograman Borland C++ Builder 6.
2. Masukan dari pengguna berupa string, baik untuk kunci kriptografi maupun teks yang aka dienkripsi.
3. Jumlah blok yang akan diamati yaitu satu buah blok, mengingat masukan berupa string juga memiliki panjang yang terbatas.

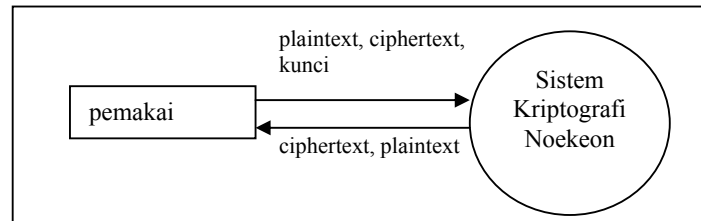
II.3 Perancangan Sistem

II.3.1 Model Simulasi

Model simulasi yang akan dirancang adalah dengan adanya masukan dari pengguna, kemudian visualisasi proses sistem, dan keluaran yang ditampilkan bagi pengguna. Oleh karena itu, akan dibuat dalam bentuk perangkat lunak yang sederhana, namun meliputi seluruh proses dalam enkripsi dan dekripsinya. Dalam menjelaskan tentang sistem yang dirancang, diberikan alur data sistem dalam bentuk diagram konteks dan Diagram Aliran Data, serta akan lebih dirinci ke dalam level diagram berikutnya.

II.3.1.1 Diagram Konteks

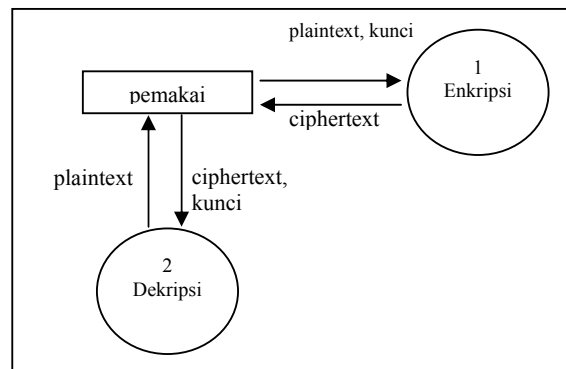
Perancangan dimulai dengan pembuatan diagram konteks, berupa penggambaran sistem penerapan algoritma Noekeon secara garis besar. Berikut ini diagram konteks sistem simulasi yang akan dirancang :



Gambar 2.1 Diagram Konteks

Dari diagram konteks tersebut, diturunkan Diagram Aliran Data (DAD) level 0 untuk penjabaran sistem yang terdiri dari :

- Proses dekripsi
- Proses enkripsi



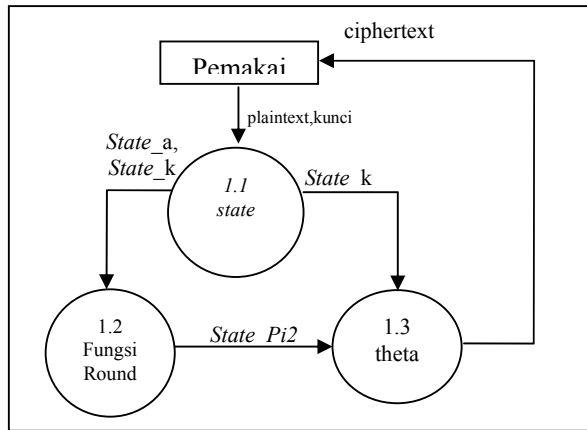
Gambar 2.2 Diagram Aliran Data

Untuk kedua proses enkripsi dan dekripsi yang masing-masing putarannya terdiri dari Θ , Π_1 , Γ , dan Π_2 , dapat lebih dirinci nantinya dalam Diagram Aliran Data yang diturunkan dari DAD level 0.

II.3.1.2 Diagram Aliran Data

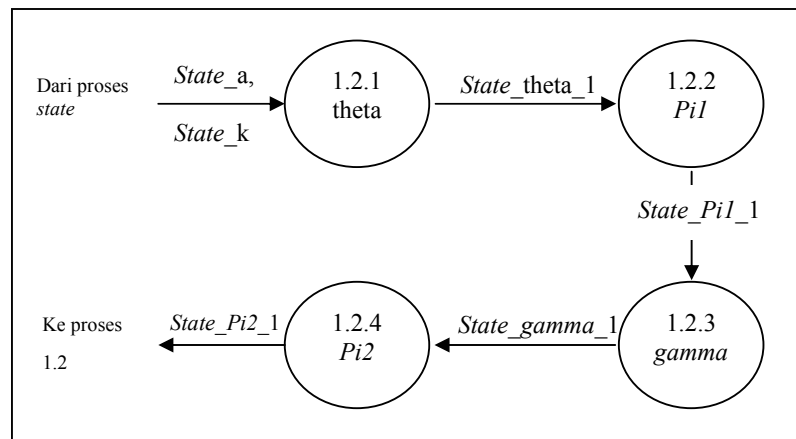
II.3.1.2.1 Proses Enkripsi

Dari DAD level 0 seperti gambar 2.2 sebelumnya, dapat diturunkan menjadi DAD level 1 proses 1 yang merincikan proses dekripsi pada algoritma Noekeon .



Gambar 2.3 DAD level 1 proses 1

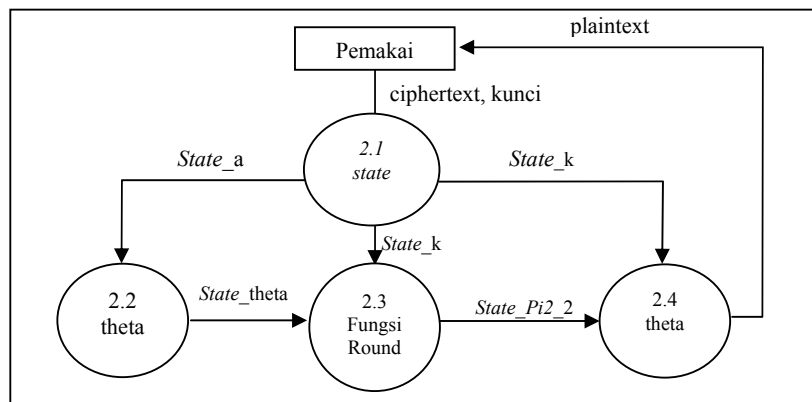
Untuk menjelaskan fungsi *Round*, diturunkan dari DAD level 1 ke DAD level 2 sebagai berikut :



Gambar 2.4 DAD level 2 proses 1.2

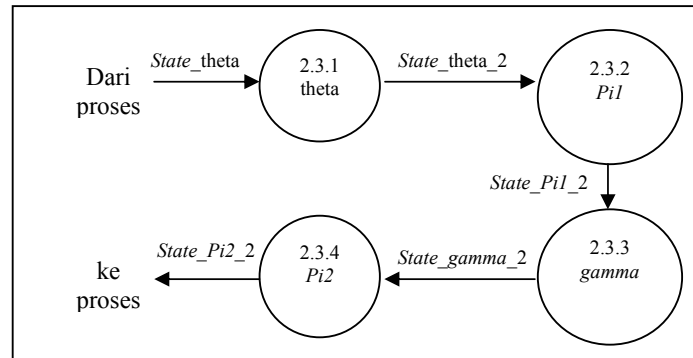
II.3.1.2.2 Proses Dekripsi

Dari DAD level 0 diatas dapat diturunkan kembali untuk menjelaskan proses dekripsi menjadi DAD level 1 proses 2 berikut ini :



Gambar 2.5 DAD level 1 proses 2

Fungsi *Round* yang terjadi pada proses dekripsi sama dengan fungsi *Round* yang terjadi pada proses enkripsi.



Gambar 2.6 DAD level 2 proses 2.3

II.3.1.3 Kamus Data

- Plaintext merupakan masukan dan keluaran dari proses sistem.
Plaintext = 0 {karakter ASCII} 16
- Ciphertext merupakan masukan dan keluaran dari proses sistem.
Ciphertext = 0 {karakter ASCII} 16
- State_a merupakan perubahan bentuk masukan (plaintext atau ciphertext) ke dalam bentuk state yaitu 4 buah 32 bit word.
State_a = 0 {karakter ASCII} 16
- State_k merupakan perubahan bentuk kunci ke dalam bentuk state yaitu 4 buah 32 bit word.
State_k = 0 {karakter ASCII} 16
- State_theta_1 merupakan state setelah hasil theta pada proses enkripsi.
State_theta_1 = 0 {karakter ASCII} 16
- State_theta_2 merupakan state setelah hasil theta pada proses dekripsi.
State_theta_2 = 0 {karakter ASCII} 16
- State_Pi1_1 merupakan state setelah hasil Pi1 pada proses enkripsi.
State_Pi1_1 = 0 {karakter ASCII} 16
- State_Pi1_2 merupakan state setelah hasil Pi2 pada proses dekripsi.
State_Pi1_2 = 0 {karakter ASCII} 16
- State_gamma_1 merupakan state setelah hasil Pi1 pada proses enkripsi.
State_gamma_1 = 0 {karakter ASCII} 16
- State_gamma_2 merupakan state setelah hasil Pi1 pada proses dekripsi.
State_gamma_2 = 0 {karakter ASCII} 16
- State_Pi2_1 merupakan state setelah hasil Pi2 pada proses enkripsi.
State_Pi2_1 = 0 {karakter ASCII} 16
- State_Pi2_2 merupakan state setelah hasil Pi2 pada proses dekripsi.
State_Pi2_2 = 0 {karakter ASCII} 16

II.3.1.4 Spesifikasi Proses

Penggunaan algoritma Noekeon pada simulasi sistem kriptografi seperti telah disebutkan, menerapkan beberapa proses yang saling berhubungan sehingga akan membentuk suatu sistem utuh yang diharapkan.

Nomor Proses	1.1, 2.1
Masukan	Teks dari pengguna. Berupa plaintext, ciphertext, atau kunci
Keluaran	State_a, state_k
Logika	128 bit masukan data, dibagi kedalam empat buah word, masing-masing 32 bit.

Nomor Proses	1.2, 2.3
Masukan	State_a, state_k, state_theta
Keluaran	State_Pi2_1, State_Pi2_2
Logika	Theta (k, a) Pi1 (a) Gamma (a) Pi2 (a)

Nomor Proses	1.3, 1.2.1, 2.2, 2.4
Masukan	State_Pi2_1, State_a, State_k, State_a, State_Pi2_2
Keluaran	ciphertext, State_theta_1, State_theta, plaintext
Logika	<pre> temp = a0 ⊕ a2 temp = temp ⊕ (temp <<<8) ⊕ (temp >>>8) a1 = a1 ⊕ temp a3 = a3 ⊕ temp a0 = a0 ⊕ k0 a1 = a1 ⊕ k1 a2 = a2 ⊕ k2 a3 = a3 ⊕ k3 temp = a1 ⊕ a3 temp = temp ⊕ (temp <<<8) ⊕ (temp >>>8) a0 = a0 ⊕ temp a2 = a2 ⊕ temp </pre>

Nomor Proses	1.2.2, 2.3.2
Masukan	State_theta_1, State_theta_2
Keluaran	State_Pi1_1, State_Pi1_2
Logika	$a1 = a1 \lll 1$ $a2 = a2 \lll 5$ $a3 = a3 \lll 2$

Nomor Proses	1.2.3, 2.3.3
Masukan	State_Pi1_1, State_Pi1_2
Keluaran	State_gamma_1, State_gamma_2
Logika	$a1 = a1 \oplus \neg (a3 \vee a2)$ $a0 = a0 \oplus (a2 \wedge a1)$ temp = a3 a3 = a0 a0 = temp $a2 = a2 \oplus a0 \oplus a1 \oplus a3$ $a1 = a1 \oplus \neg (a3 \vee a2)$ $a0 = a0 \oplus (a2 \wedge a1)$

Nomor Proses	1.2.4, 2.3.4
Masukan	State_gamma_1, State_gamma_2
Keluaran	State_Pi2_1, State_Pi2_2
Logika	$a1 = a1 \ggg 1$ $a2 = a2 \ggg 5$ $a3 = a3 \ggg 2$

II.4 Implementasi Sistem

Implementasi perangkat lunak simulasi ini dilakukan dengan membuat antarmuka yang sederhana dalam mempermudah pemberian masukan serta memberikan visualisasi yang baik terhadap proses sistem yang terjadi. Antarmuka ini terdiri atas kontrol masukan dan keluaran, menu pilihan, serta kontrol visualisasi.

BAB III

HASIL PERANCANGAN SIMULASI DAN ANALISA KINERJA ALGORITMA NOEKEON

Dalam bab ini akan ditampilkan urutan kerja pada proses enkripsi dan dekripsi yang dilakukan oleh algoritma Noekeon. Analisa yang dilakukan adalah seberapa jauh perbedaan hasil proses enkripsi dari masukkan *plaintext* yang diberikan. Kemudian, apakah dari hasil teks yang terenkripsi, apabila dilakukan proses pembalikan akan diperoleh *plaintext* sesuai aslinya.

Tampilan muka untuk menu hasil perancangan dapat dilihat pada pada “*pogram simulasi*” yang disertakan pada laporan ini (*MADE_ARI.exe*), dimana pada tampilan terdapat tiga menu utama, yaitu :

1. Menu Simulasi String
Menu simulasi string adalah untuk melakukan proses enkripsi dan dekripsi dengan memasukkan berupa string untuk menjadi *ciphertext* atau sebaliknya. Pada proses tersebut diperlihatkan perubahan untuk setiap langkah Noekeon, yaitu *theta*, *Pi1*, *gamma*, dan *Pi2*, untuk masing-masing putarannya, hingga diperoleh keluaran yang diharapkan.
2. Menu Simulasi File
Menu simulasi file untuk melakukan proses enkripsi dan dekripsi pada file.
3. Menu Efek Avallanche
Sedangkan menu efek avalanche digunakan untuk mengamati efek avalanche pada algoritma Noekeon.

III.1 Hasil Simulasi

Untuk memulai simulasi, terdapat pilihan proses yaitu enkripsi atau dekripsi, serta masukan berupa string dan kunci yang dipakai.

III.1.1 Proses Enkripsi

Pada proses ini dibatasi masukan berupa 16 buah karakter bertipe string, karena akan lebih mudah mengamati perubahannya saat dienkripsi. Kunci yang dipakai dibatasi 16 karakter. Sebagai contoh yang akan diamati adalah string “PESAN YG RAHASIA” sebagai *plaintext*, dan kunci yang akan kita pakai adalah “KUNCI YG RAHASIA”. Untuk memulai proses enkripsi kita pastikan *radio button* kita tandai pada mode enkripsi.

Sebelum proses enkripsi, string masukan dirubah ke bilangan hexadesimal dengan menekan tombol ‘Konversi’. Setelah itu baru dilakukan proses simulasi enkripsi Noekoen. Proses untuk 16 putaran algoritma Noekeon untuk menghasilkan pesan yang terenkripsi atau *ciphertext* yang sedang berjalan dapat diamati di jendela keluaran enkripsi.

Berikut ini pesan dan kunci dalam bentuk hexadesimal-nya :

String	Hexadesimal
PESAN YG RAHASIA	504553414e2059472052414841534941
KUNCI YG RAHASIA	4b554e43492059472052414841534941

Seperti telah dibahas dalam teori algoritma ini, untuk setiap putarannya yang dioperasikan adalah sebuah *state*. Pada putaran pertama *state* adalah hexadesimal dari pesan, yang dibagi menjadi empat buah word.

<i>Plaintext</i>	<i>State(a0,a1,a2,a3)</i>
4b554e43492059472052414841534941	50455341 e205947 20524148 41534941

Setelah itu, *state* bersama *working-key* akan dilakukan operasi pertama pada algoritma Noekeon, yaitu *Theta*. Setelah proses *theta*, *state* akan berubah dan menghasilkan nilai *theta* sebagai berikut :

<i>State</i>	<i>Theta</i>
50455341 e205947 20524148 41534941	1c171d85 e9758ceb 07070007 ee758ceb

Setelah itu, keluaran dari *theta* dilakukan proses berikutnya, yaitu *Pi1*. Pada tahap ini diperoleh nilai *Pi1* sebagai berikut :

<i>Theta</i>	<i>Pi1</i>
1c171d85 e9758ceb 07070007 ee758ceb	1c171d85 d2eb19d7 e0e000e0 b9d633af

Proses selanjutnya yaitu proses *gamma*. Keluaran dari *gamma* diperoleh nilai sebagai berikut :

<i>Pi1</i>	<i>gamma</i>
1c171d85 d2eb19d7 e0e000e0 b9d633af	e9f4e26a f6ead5f5 5123fbcd dcf71d45

Kemudian, keluaran tersebut akan dilakukan operasi yang terakhir yaitu operasi *Pi2*. Keluaran *Pi2* adalah sebagai berikut:

<i>Gamma</i>	<i>Pi2</i>
e9f4e26a f6ead5f5 5123fbcd dcf71d45	e9f4e26a fb756afa 6a891fde 773dc751

Untuk memperoleh *state* dari putaran pertama, dilakukan operasi *round constant*. Nilai inilah yang nantinya akan menjadi *state* untuk putaran kedua dari algoritma ini.

<i>Pi2</i>	<i>Round constant</i>
e9f4e26a fb756afa 6a891fde 773dc751	b0a387a6 e3561c6c 58d97502 676da1c1

Secara lengkap, untuk memperoleh pesan yang telah dienkripsi, Noekeon melakukan proses seperti diatas, berulang sebanyak 16 putaran. Hasil lengkap 16 putaran terdapat pada lampiran.

Dari hasil 16 putaran tersebut akan diperoleh pesan yang telah terenkripsi yaitu :

<i>State terakhir</i>	<i>Ciphertext</i>
60fa82ac 13ba18c4 b593d74d 4f4e2fda	`ú,-□°□Äμ“×MON/Ú

Tampak bahwa pesan hasil enkripsi menjadi tidak terbaca lagi dan sangat jauh berbeda dari pesan asli yang kita berikan.

III.1.2 Proses Dekripsi

Proses dekripsi dilakukan guna memperoleh pesan yang dienkripsi, sehingga *ciphertext* yang dihasilkan dapat terbaca kembali. Sebagai masukan adalah *ciphertext*, sedangkan kunci yang digunakan tetap sama yaitu “KUNCI YG RAHASIA”. Langkah untuk memulai proses simulasi dekripsi adalah sebagai berikut :

- Atur *radio button* ke mode dekripsi.
- Tekan tombol ‘Konversi’ untuk merubah string ke dalam nilai hexadesimal.
- Tekan tombol ‘Proses’ untuk memulai.

Proses yang berjalan akan ditampilkan di jendela keluaran dekripsi. Pada proses ini, merupakan kebalikan dari proses enkripsi. Pertama, nilai hexadesimal dari masukan yang diberikan, dibentuk menjadi *state* yang akan di proses pada operasi *theta*.

Masukan	Hexadesimal
`ú,-□°□Äμ“×MON/Ú	60fa82ac13ba18c4b593d74d4f4e2fda

<i>State(a0,a1,a2,a3)</i>
60fa82ac 13ba18c4 b593d74d 4f4e2fda

Kemudian, setelah dilakukan operasi *theta*, akan menghasilkan keluaran sebagai berikut:

<i>State</i>	<i>Theta</i>
60fa82ac 13ba18c4 b593d74d 4f4e2fda	9d30119a 60109f8d 235e4b70 3497b895

Setelah itu, keluaran dari *theta* dilakukan proses berikutnya, yaitu *Pil*. Pada tahap ini diperoleh nilai *Pil* sebagai berikut :

<i>Theta</i>	<i>Pil</i>
9d30119a 60109f8d 235e4b70 3497b895	9d30114e c0213f1a 6bc96e04 d25ee254

Proses selanjutnya yaitu proses *gamma*. Keluaran dari *gamma* diperoleh nilai sebagai berikut :

<i>Pi1</i>	<i>gamma</i>
9d30114e c0213f1a 6bc96e04 d25ee254	525feef5 c6496ea1 a0a79daf dd313f4e

Kemudian, keluaran tersebut akan dilakukan operasi yang terakhir yaitu operasi *Pi2*. Keluaran *Pi2* adalah sebagai berikut:

<i>Gamma</i>	<i>Pi2</i>
525feef5 c6496ea1 a0a79daf dd313f4e	525feef5 e324b750 7d053ced b74c4fd3

Untuk memperoleh *state* dari putaran pertama, dilakukan operasi *round constant*. Nilai inilah yang nantinya akan menjadi *state* untuk putaran kedua dari algoritma ini.

<i>Pi2</i>	<i>Round constant</i>
525feef5 e324b750 7d053ced b74c4fd3	a6ceb3f3 a0c07d9d e2936e8a fcdb9518

Secara lengkap, untuk mengembalikan pesan yang telah dienkripsi menjadi terbaca kembali, Noekeon melakukan proses seperti diatas, berulang sebanyak 16 putaran. Hasil lengkap 16 putaran terdapat pada lampiran.

Dari hasil 16 putaran tersebut akan diperoleh pesan yang asli yaitu :

<i>State terakhir</i>	<i>Ciphertext</i>
50455341 4e205947 20524148 41534941	PESAN YG RAHASIA

III.2 Proses File

Pada submenu file, berfungsi untuk dekripsi dan enkripsi arsip. Pada sub menu ini tidak diperlihatkan proses kerja algoritma, hanya menunjukkan kemampuan algoritma Noekeon dalam mengenkripsi dan mendekripsi suatu file.

Tabel berikut ini menunjukkan hasil proses enkripsi dan dekripsi untuk masukan berbentuk file.

Tabel 3.1 Hasil Proses Pada File

Nama File	Besar File (byte)	Ukuran File Hasil Proses (Byte)	
		Enkripsi	Dekripsi
OST_FFVIII_MID.zip	422,842	422,852	422,842
Noekeon-spec.pdf	166,187	166,196	166,187
history.swf	1,551,900	1,551,908	1,551,900
Cryptography-Digest Digest #341.htm	35,629	35,636	35,629
noekeon in perl.txt	5,773	5,780	5,773
Made Ari.jpg	425,708	425,716	425,708
Atm_1.doc	1,345,536	1,345,556	1,345,536
Sophie_Ellis_Bextor-MurderOnTheDancefloor.mpeg	40,316,752	40,316,772	40,316,752
Andrea Bocelli & Sarah Brightman - Time to Say Goodbye.MP3	3,408,428	3,408,436	3,408,428

Dari hasil pengujian, tampak bahwa untuk mengenkripsi suatu file tidak dibatasi jenisnya, karena setelah dilakukan proses dekripsi, hasil yang diperolehnya identik dengan file asli sebelum proses enkripsi untuk setiap jenis file.

Sedangkan ukuran file hasil enkripsi lebih besar dari ukuran asli, hal ini disebabkan adalah proses *padding* untuk memperoleh blok terakhir sebesar 128 bit.

III.3 Analisa Unjuk Kerja Algoritma Noekeon

III.3.1 Struktur Cipher

Struktur *cipher* pada algoritma Noekoen ditekankan pada kesederhanaan transformasi, yaitu komposisi desainnya yang hanya terdiri dari tiga buah transformasi linear *theta*, *Pi1*, dan *Pi2*, serta satu buah transformasi non-linear *gamma*. Operasinya diimplementasikan hanya menggunakan operasi-operasi *bit-wise* serta operasi pergeseran bit. Namun struktur yang dirancang, diharapkan memberikan keamanan data yang cukup baik.

Suatu *cipher* dikatakan memiliki peluang keamanan yang sempurna (*perfect secrecy*), jika antara *plaintext* dan *ciphertext* tidak memiliki keterhubungan satu sama lain. Untuk menganalisa ada tidaknya keterhubungan tersebut pada algoritma Noekoen, dilakukan pengujian terhadap sebuah *plaintext*, namun dilakukan enkripsi dengan beberapa kunci yang berbeda.

Yang pertama diberikan nilai kosong pada kunci dan pesan yang akan dienkripsi. Hasil enkripsi sebagai berikut :

```
Key          : 30000000000000000000000000000000000000000000
Plaintext   : ABCDEFGHIJKLMNOP
Ciphertext  : ffa692f617f0eb6850ae9a36c1e1c195
```

Selanjutnya dilakukan lagi pengamatan *plaintext* yang sama, namun pada kunci dilakukan perubahan sebanyak satu bit, hasilnya sebagai berikut:

```
Key       : 31000000000000000000000000000000
Plaintext : ABCDEFGHIJKLMNOP
Ciphertext : 7f2611f312f3dd5fe01c1937c763f7a0
```

Hubungan antara *plaintext* dan *ciphertext* kedua percobaan diatas, jika ditabelkan adalah sebagai berikut:

Tabel 3.2 Hubungan *plaintext-ciphertext* dengan dua kunci berbeda

key	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ff	a6	92	f6	17	f0	eb	68	50	ae	9a	36	c1	e1	c1	95
2	7f	26	11	f3	12	f3	dd	5f	e0	1c	19	37	c7	63	f7	a0

Dari tabel 3.2, dari percobaan diatas, tidak nampak hubungan nilai masing-masing karakter *plaintext* dengan nilai *ciphertext* yang diperolehnya walaupun kunci yang digunakan hanya berbeda satu bit. Hal ini memenuhi persyaratan *perfect secrecy* bagi algoritma Noekeon.

III.3.2 Proses Enkripsi dan Dekripsi

Struktur enkripsi dan dekripsi pada Noekeon dirancang menggunakan kode maupun rangkaian yang sama. Sehingga akan menghasilkan kesederhanaan dalam implementasinya.

III.3.3 Penjadwalan Kunci

Penjadwalan kunci pada Noekeon digunakan pada mode *indirect-key*. Metode ini akan lebih memperkuat algoritma terhadap usaha kriptanalisis, terutama dalam usaha untuk mendapatkan kunci yang dipakai. Penjadwalan kunci dilakukan melalui proses yang sama dengan algoritma Noekeon itu sendiri, yaitu melalui tahap *Theta*, *Pi1*, *gamma*, dan *Pi2*, dan dilakukan putaran sebanyak 16 kali. Namun *working-key* yang digunakan adalah *NullVektor*. Sehingga kunci yang akan menjadi kunci enkripsi mempunyai kekuatan yang sama dengan *cipher* Noekeon itu sendiri.

III.3.4 Sifat Simetris, Kunci lemah, dan Kunci Setengah Lemah

Sifat simetri setiap putaran pada suatu algoritma kriptografi memberikan peluang seorang kriptanalisis untuk membongkar suatu *ciphertext* dengan mengaplikasikan suatu teknik serangan tertentu. Pada Noekeon ini dapat terjadi apabila dalam setiap transformasinya tidak menambahkan *round constant*. Seperti telah dijelaskan, sifat simetris pada Noekeon dihilangkan dengan penggunaan *round constant* yang berbeda pada setiap putarannya. Sehingga secara praktis akan menghilangkan kemungkinan timbulnya kunci lemah dan kunci setengah lemah, seperti yang dimiliki oleh DES.

DES memiliki empat buah kunci lemah dan enam buah pasangan kunci setengah lemah. Dalam Noekeon kunci-kunci tersebut tidak berpengaruh apabila diaplikasikan, seperti pada tabel dibawah ini :

Tabel 3.3 Kunci lemah pada DES (dari referensi)

DES weak-key (k)	Plaintext (P)	$E_k(P)$ - Noekeon
0101010101010101	0000000000000000	7833f85eb312e32fa66177d5d083010d
FEFEFEFEFEFEFEFE	0000000000000000	ed816d5aa111f1120ec1deedc3101330
1F1F1F1F0E0E0E0E	0000000000000000	8599052eaa02fa6814dac4ebcc1b1c4e
E0E0E0E0F1F1F1F1	0000000000000000	fc27f188020d3020ed1dfbca2317061

Tabel 3.4 Kunci Setengah lemah pada DES (dari referensi)

DES Semi weak-key (k)	$D_k(C)$	$E_k(P)$ - Noekeon
01FE01FE01FE01FE	d44cb8e7d5b15c4b933c0fc9f a30be39	9215965bd3b7c2ce45e3303ef a9348c7
FE01FE01FE01FE01		
1FE01FE00EF10EF1	5cc710d5b2a21a615d84c1e0d 8abbc02	9f4a6202ddb293ded99692e5 f969e53
E01FE01FF10EF10E		
01E001E001F101F1	8d46e1d4e5b26c79da3546d98 a3bce1a	249859ac185c5cb4563ee2b01 a57ebc7
E001E001F101F101		
1FFE1FFE0EFE0EFE	05cd49e682a12a53148d88f0a 8a0cc21	7b1a87e79e6c1411124ac220e 6ced49e
FE1FFE1FFE0EFE0E		
011F011F010E010E	d86cb4bba6912f109f3c03c68 910cd6e	1aba83424eda405d8c185de96 1dee23e
1F011F010E010E01		
E0FEE0FEF1FEF1FE	11cc7ea7c8824260528ccea1e 683a152	44d6bff6426ee5aee81e0546f fe5a15a
FEE0FEE0FEF1FEF1		

Dari tabel 3.3 tidak tampak bahwa kunci lemah yang dimiliki oleh DES berpengaruh apabila diaplikasikan kepada Noekeon, karena hasil enkripsi berbeda dengan *plaintext* yang diberikan. Sedangkan dari tabel 3.4, pasangan-pasangan kunci setengah lemah yang dimiliki oleh DES tidak menunjukkan sebagai kunci setengah lemah bagi Noekeon, karena kunci pasangan yang dimiliki satu kunci tidak berhasil memperoleh *plaintext* yang asli. Dalam Noekeon sendiri, sampai saat ini belum ditemukan kunci lemah dan kunci setengah lemah, sehingga tidak ada batasan dalam pemilihan kunci yang akan dipakai. Dalam Noekeon sendiri, sampai saat ini belum ditemukan kunci lemah dan kunci setengah lemah, sehingga tidak ada batasan dalam pemilihan kunci yang akan dipakai.

III.3.5 Efek *Avalanche*

Suatu algoritma kriptografi memenuhi kriteria *Strict Avalanche Criterion* (SAC) apabila rata-rata perubahan bit keluaran terhadap berubahnya satu bit pada masukan setidaknya adalah 50% . Perhitungan presentase avalanche suatu algoritma dipenuhi dengan persamaan berikut, dengan terlebih dahulu mengembalikan nilai hexadesimal setiap karakter menjadi nilai binernya:

Berikut ini hasil pengamatan efek avalanche pada Noekoen :

$$\frac{\sum bit_beda}{\sum bit_total} 100\% \quad (3.1)$$

1. Perubahan pada kunci :

```
P: 00000000000000000000000000000000
K: 00000000000000000000000000000000
C: B1656851699E29FA24B70148503D2DFC
K: 000000000000000000000000000000001
C: 138919FB3443DC23F7CFDEFE483142E1
```

Perbedaan bit yang dihasilkan sebanyak 73 bit. Dengan menggunakan persamaan (3.1) diperoleh perbedaan antara dua buah *ciphertext* tersebut sebanyak 57,03 %.

2. Perubahan pada *plaintext* :

```
K: eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
P: 00000000000000000000000000000000
C: 108A1D565253FCA42374FD5AC3666923
P: 000000000000000000000000000000001
c: F34DF3F41D5725139269B2092B22FA5D
```

Jumlah perbedaan bit yang dihasilkan sebanyak 63 bit, sehingga efek avalanche yang dihasilkan antara kedua buah *ciphertext* diperoleh sebanyak 53,91 %.

Percobaan efek avalanche juga dilakukan terhadap 32 buah masukan acak.. Hasil secara lengkap terdapat pada lampiran. Untuk percobaan perubahan satu bit pada *plaintext* menghasilkan efek avalanche rata-rata 52,78 %. Sedangkan perubahan satu bit pada kunci, menghasilkan efek avalanche sebesar 52,37 %.

Dari hasil diatas, efek avalanche yang dihasilkan Noekeon diatas 50%, berarti Noekeon memenuhi kriteria SAC, sebagai algoritma yang cukup baik.

III.3.6 Kesalahan Propagasi

Dimungkinkan terjadi kesalahan baik pada proses enkripsi maupun saat proses pertukaran data melalui media tertentu yang disebabkan oleh adanya interferensi baik oleh gelombang frekuensi maupun yang dilakukan oleh penyadap, sehingga terjadi perubahan informasi atau bit-bit tertentu pada *ciphertext*. Untuk melihat sejauh mana kesalahan tersebut berpengaruh terhadap *plaintext* hasil dekripsi, pada analisa ini dilakukan terhadap satu blok data yang

dipresentasikan dengan sebuah file sebesar 128 bit. Data hasil enkripsi, *ciphertext* yang diperoleh dilakukan perubahan beberapa bit pada blok pertama (16 karakter ASCII) sebagai simulasi terjadinya kesalahan. Kemudian *ciphertext* tersebut didekripsi kembali untuk memperoleh *plaintext*.

Dari Hasil yang diperoleh, ternyata *plaintext* yang diperoleh kembali mengalami kerusakan pada blok pertama saja. Hal ini terjadi karena pada algoritma Noekeon yang diimplementasikan ini menggunakan mode ECB, yaitu masing-masing blok dioperasikan secara independen. Kesalahan yang terjadi pada sebuah blok *plaintext*, tidak akan mempengaruhi kepada blok lainnya saat dilakukan enkripsi. Sehingga saat dilakukan proses dekripsi, kesalahan yang terjadi hanya pada blok yang bersangkutan saja.

III.3.7 Kekuatan Terhadap Jenis Serangan *Brute-Force*

Serangan bertipe *brute-force* adalah dengan menerapkan percobaan setiap kemungkinan kunci yang ada satu per satu sampai diperoleh *plaintext* yang diharapkan. Waktu yang diperlukan untuk memperoleh kunci yang diharapkan, selalu berbanding lurus dengan panjang bit kunci yang dimiliki oleh algoritma kriptografi, dan makin cepat prosesor yang dipakai, makin cepat waktu yang pula dibutuhkan untuk dapat memperoleh kunci tersebut.

Dengan panjang kunci 128 bit, berdasarkan perkiraan pakar kriptografi maka algoritma Noekeon akan bertahan mampu terhadap serangan bertipe *brute-force* selama ± 100 tahun.

BAB IV

KESIMPULAN

Dari hasil analisa pada tugas ini, dapat diambil kesimpulan sebagai berikut :

1. Pada tugas ini berhasil membuat program simulasi algoritma kriptografi Noekeon dengan bahasa pemrograman Borland C++6 dan berjalan dengan baik sehingga dapat menerangkan cara kerjanya.
2. Pencampuran antara *state* dari kunci *state* dan *plaintext* pada proses theta akan menghasilkan nilai *state* yang jauh berbeda dari *state* awal, dengan diikuti pula proses pergeseran bit dan operasi gamma, kemudian dilakukan perputaran 16 kali, menghasilkan enkripsi data yang kuat.
3. Semua kunci lemah dan kunci setengah lemah yang dimiliki oleh DES, tidak berpengaruh apabila diterapkan pada algoritma Noekeon, sehingga pada algoritma ini tidak ada pembatasan kunci yang digunakan ataupun pemanfaatan kunci-kunci tersebut untuk melakukan eksploitasi terhadap *ciphertext*. Tidak tampak kelemahan algoritma Noekeon dibanding algoritma DES yang mengacu pada analisa kunci lemah dan setengah lemah
4. Efek *avalanche* yang diperoleh memenuhi kriteria yang cukup baik sesuai kriteria *Strict Avalanche Criterion (SAC)* yaitu pada perubahan satu bit kunci ataupun *plaintext* menyebabkan lebih dari separuh perubahan bit pada *ciphertext*.
5. Kesalahan atau kerusakan yang terjadi pada sebuah blok *plaintext*, tidak akan mempengaruhi kepada blok lainnya saat dilakukan enkripsi.
6. Dengan panjang kunci 128 bit, algoritma Noekeon akan bertahan mampu terhadap serangan bertipe *brute-force* selama ± 100 tahun.

DAFTAR PUSTAKA

- [1] Brickell Ernest F., Odlyzko, Andrew M. *Cryptanalysis: A Survey of Recent Result*.
- [2] Daemen, Joan, Peeters, Michaël. Van Assche, Gilles. and Rijmen, Vincent. *NOEKEON Block Cipher, Nessie Proposal*. October 27, 2000.
- [3] Daemen, Joan, Peeters, Michaël. Van Assche, Gilles. and Rijmen, Vincent. *NOEKEON, Slide*. September 13, 2000.
- [4] Daemen, Joan, *Propagation and Correlation, Chapter 5 of Cipher and Hash Function Design*, distributed as PDF file as annex to AES submission Rijndael.
- [5] Daemen, Joan. L. Knudsen. Rijmen, Vincent. *The Block Cipher Square*, Paper.
- [6] Feistel, H. *Cryptography and Computer Privacy*. Scientific American, 1973.
- [7] Menezes, P. Van Oorschot, and S. Vanstone. *The Handbook of Applied Cryptography*, CRC Press, 1996.
- [8] Mohammad Sbastian Widodo, *Simulasi Algoritma Kriptografi Noekoen dalam Penyandian Data* , Tugas Akhir STT Telkom, 2002
- [9] N.P. Smart, *Physical Side-Channel Attacks On Cryptographic Systems*.
- [10] P. Michael. *Secure Data Networking*, Artech House, Inc. 1993.
- [11] Schneier, B. *Applied cryptography Second Edition*. John Wiley & Sons, Inc. 1996.
- [12] St Dennis, Tom, *Block Cipher, an Introduction to Modern Cryptanalysis*, Algonquin College, 2001.
- [13] St Dennis, Tom, *Differential Cryptanalysis of NOEKEON – DRAFT*, Algonquin College, 2000.
- [14] Webster, A. and S. Tavares. *On the Design of S-Boxes*. Advances in Cryptology, CRYPTO 1985.

LAMPIRAN A:

PROSES ENKRIPSI LENGKAP

```
Key          : KUNCI YG RAHASIA
Key          : 4b554e43492059472052414841534941
PlainText   : PESAN YG RAHASIA
PlainText   : 504553414e2059472052414841534941

State       : 50455341 4e205947 20524148 41534941

Theta[0]    : 1c171d85 e9758ceb 07070007 ee758ceb
Pi1[0]      : 1c171d85 d2eb19d7 e0e000e0 b9d633af
Gamma[0]    : e9f4e26a f6ead5f5 5123fbcd dcf71d45
Pi2[0]      : e9f4e26a fb756afa 6a891fde 773dc751
a Round[0]  : b0a387a6 e3561c6c 58d97502 676da1c1

Theta[1]    : 949f8774 8716728e 17e27ac0 0b5edf25
Pi1[1]      : 949f8774 0e2ce51d fc4f5802 2d7b7c94
Gamma[1]    : 29737880 2ec85e7d 450b8596 9893c774
Pi2[1]      : 29737880 97642f3e b2285c2c 2624f1dd
a Round[1]  : 0d622e67 84a0935c fd3e05f6 3d935db9

Theta[2]    : 54181d9e c6071a67 cf433932 7747c484
Pi1[2]      : 54181d9e 8c0e34cf e8672659 dd1f1211
Gamma[2]    : db97e271 8e8fffb61 67e8f0f6 dc1e39d7
Pi2[2]      : db97e271 c747fdb0 b33f4787 f7078e75
a Round[2]  : 58aa3fc6 d40233a0 5b059557 ec315063

Theta[3]    : a6fc876f cc2492b3 ce542299 fc64e176
Pi1[3]      : a6fc876f 98492567 ca845339 f19385db
Gamma[3]    : 79938898 cc202d53 89ca5dcf 2efc864e
Pi2[3]      : 79938898 e61016a9 7c4e52ee 8bbf2193
a Round[3]  : b758017d d932e69f d982d4d8 bceec1a3

Theta[4]    : 04f8ed86 5973cd1e 012537f0 34dcfa24
Pi1[4]      : 04f8ed86 b2e79a3c 24a6fe00 d373e890
Gamma[4]    : fb1772d1 286e9b6b 6964fa45 245e7786
Pi2[4]      : fb1772d1 94374db5 2b4b27d2 89179de1
a Round[4]  : a4c4fdbe f9ceb5df 1f9fa71d ec9d758d

Theta[5]    : 867c17cf 5854e571 562042cc 45743525
Pi1[5]      : 867c17cf b0a9cae2 c408598a 15d0d495
Gamma[5]    : 1dd2fc15 2a0768a0 4d223ad0 06745f4d
Pi2[5]      : 1dd2fc15 1503b450 826911d6 419d17d3
a Round[5]  : 62856971 f6ea35eb 96398bf4 aa07866e

Theta[6]    : 60b9990b 3f60fa72 ff0274c8 6bfe59f1
Pi1[6]      : 60b9990b 7ec1f4e4 e04e991f aff967c5
Gamma[6]    : 8f7867c5 b0c10024 218f0311 00f9090f
Pi2[6]      : 8f7867c5 58608012 890c7818 c03e4243
a Round[6]  : 2e428109 242df50b 43319145 b400275c

Theta[7]    : f21302db a50319e7 f4671d06 3d5ddbb6
Pi1[7]      : f21302db 4a0633cf 8ce3a0de f5776ed8
Gamma[7]    : bd7d6c34 4c6a33ec cb8bcefd fa112215
Pi2[7]      : bd7d6c34 263519f6 ee5c5e77 7e844885
a Round[7]  : 11fbd43c 71553fbc 29dde95b 21977ec9

Theta[8]    : c016092a 6e5635b6 93373b69 36e764c5
Pi1[8]      : c016092a dcac6b6c 66e76d32 db9d9314
Gamma[8]    : 1fb9f295 c6a561d1 e564f589 84b2600a
Pi2[8]      : 1fb9f295 e352b0e8 4f2b27ac a12c9802
a Round[8]  : ff936516 0f65c94d c406bf7a 4568f1a1
```

```
Theta[9]      : 62d2978b da31edd9 324042b2 984fc533
Pi1[9]       : 62d2978b b463dbb3 48085646 613f14ce
Gamma[9]     : 413d644c b68a78f6 6946f583 62d2c589
Pi2[9]       : 413d644c 5b453c7b 1b4a37ac 58b4b162
a Round[9]   : 9c7c2a3a 631b1d69 ad0c766d 68998076

Theta[10]    : 3c560152 802683a8 662152b2 83d70eb1
Pi1[10]     : 3c560152 004d0751 c42a564c 0f5c3ac6
Gamma[10]   : 0f98bea6 30cd9665 c7e4edf8 3c5e0712
Pi2[10]     : 0f98bea6 9866cb32 c63f276f 8f1781c4
a Round[10] : a98a5833 15b1068f 0b2ace92 0ab35c7f

Theta[11]    : 72f3dbb8 9c05ab3e bb544271 8b74e1c8
Pi1[11]     : 72f3dbb8 380b567d 6a884e37 2dd38722
Gamma[11]   : 8ddca507 a82f26e7 b58f322d 5afb9d8d
Pi2[11]     : 8ddca507 d4179373 6dac7991 56bee763
a Round[11] : 80bd437a 5d7b3658 0bca9021 d7a1524e

Theta[12]    : f0b7c019 757456da 10c71c8f f7dd22ca
Pi1[12]     : f0b7c019 eae8adb4 18e391e2 df748b2b
Gamma[12]   : 1f540bab c868e5a6 f52092d0 f85741b9
Pi2[12]     : 1f540bab 643472d3 87a90496 7e15d06e
a Round[12] : aff31e64 e27e73a9 5c091ec5 f02cc112

Theta[13]    : 623f6eb5 9457e62b fac26188 8e764496
Pi1[13]     : 623f6eb5 28afcc57 584c311f 39d9125a
Gamma[13]   : 9dd0124a be4990ba a52b4d10 6a336ea2
Pi2[13]     : 9dd0124a 5f24c85d 85295a68 9a8cdba8
a Round[13] : 33907701 e0ad375d 406e301d 2d7634ae

Theta[14]    : e0bb7754 0d47fe07 f8423f76 c8efedf2
Pi1[14]     : e0bb7754 1a8ffc0e 0847eedf 23bfb7cb
Gamma[14]   : 2f348beb dc8fbcab 0dcb3e60 e8bc9b5a
Pi2[14]     : 2f348beb ee47de55 006e59f3 ba2f26d6
a Round[14] : a6ceb3f3 a0c07d9d e2936e8a fcdb9518

Theta[15]    : 525feef5 e324b750 7d053ced b74c4fd3
Pi1[15]     : 525feef5 c6496ea1 a0a79daf dd313f4e
Gamma[15]   : 9d30114e c0213f1a 6bc96e04 d25ee254
Pi2[15]     : 9d30114e 60109f8d 235e4b70 3497b895
a Round[15] : 60fa82ac 13ba18c4 b593d74d 4f4e2fda

State       : 60fa82ac 13ba18c4 b593d74d 4f4e2fda

cipherText  : 60fa82ac13ba18c4b593d74d4f4e2fda
cipherText  : `ú,-□°□Åµ"×MON/Ú
```

PROSES DEKRIPSI LENGKAP

Key : KUNCI YG RAHASIA
Key : 4b554e43492059472052414841534941
CipherText : `ú,¬□°□Äµ"xMON/Ú
CipherText : 60fa82ac13ba18c4b593d74d4f4e2fda

State : 60fa82ac 13ba18c4 b593d74d 4f4e2fda

Theta[0] : 9d30119a 60109f8d 235e4b70 3497b895
Pi1[0] : 9d30114e c0213f1a 6bc96e04 d25ee254
Gamma[0] : 525fee5 c6496ea1 a0a79daf dd313f4e
Pi2[0] : 525fee5 e324b750 7d053ced b74c4fd3
a Round[0] : a6ceb3f3 a0c07d9d e2936e8a fcdb9518

Theta[1] : 2f348b81 ee47de55 006e59f3 ba2f26d6
Pi1[1] : 2f348beb dc8fbcab 0dcb3e60 e8bc9b5a
Gamma[1] : e0bb7754 1a8ffc0e 0847eedf 23bfb7cb
Pi2[1] : e0bb7754 0d47fe07 f8423f76 c8efedf2
a Round[1] : 33907701 e0ad375d 406e301d 2d7634ae

Theta[2] : 9dd0127f 5f24c85d 85295a68 9a8cdba8
Pi1[2] : 9dd0124a be4990ba a52b4d10 6a336ea2
Gamma[2] : 623f6eb5 28afcc57 584c311f 39d9125a
Pi2[2] : 623f6eb5 9457e62b fac26188 8e764496
a Round[2] : aff31e64 e27e73a9 5c091ec5 f02cc112

Theta[3] : 1f540b3c 643472d3 87a90496 7e15d06e
Pi1[3] : 1f540bab c868e5a6 f52092d0 f85741b9
Gamma[3] : f0b7c019 eae8adb4 18e391e2 df748b2b
Pi2[3] : f0b7c019 757456da 10c71c8f f7dd22ca
a Round[3] : 80bd437a 5d7b3658 0bca9021 d7a1524e

Theta[4] : 8ddca5c1 d4179373 6dac7991 56bee763
Pi1[4] : 8ddca507 a82f26e7 b58f322d 5afb9d8d
Gamma[4] : 72f3dbb8 380b567d 6a884e37 2dd38722
Pi2[4] : 72f3dbb8 9c05ab3e bb544271 8b74e1c8
a Round[4] : a98a5833 15b1068f 0b2ace92 0ab35c7f

Theta[5] : 0f98bec5 9866cb32 c63f276f 8f1781c4
Pi1[5] : 0f98bea6 30cd9665 c7e4edf8 3c5e0712
Gamma[5] : 3c560152 004d0751 c42a564c 0f5c3ac6
Pi2[5] : 3c560152 802683a8 662152b2 83d70eb1
a Round[5] : 9c7c2a3a 631b1d69 ad0c766d 68998076

Theta[6] : 413d64f0 5b453c7b 1b4a37ac 58b4b162
Pi1[6] : 413d644c b68a78f6 6946f583 62d2c589
Gamma[6] : 62d2978b b463dbb3 48085646 613f14ce
Pi2[6] : 62d2978b da31edd9 324042b2 984fc533
a Round[6] : ff936516 0f65c94d c406bf7a 4568f1a1

Theta[7] : 1fb9f2cb e352b0e8 4f2b27ac a12c9802
Pi1[7] : 1fb9f295 c6a561d1 e564f589 84b2600a
Gamma[7] : c016092a dcac6b6c 66e76d32 db9d9314
Pi2[7] : c016092a 6e5635b6 93373b69 36e764c5
a Round[7] : 11fbd43c 71553fbc 29dde95b 21977ec9

Theta[8] : bd7d6c1b 263519f6 ee5c5e77 7e844885
Pi1[8] : bd7d6c34 4c6a33ec cb8bcefd fa112215
Gamma[8] : f21302db 4a0633cf 8ce3a0de f5776ed8
Pi2[8] : f21302db a50319e7 f4671d06 3d5ddbb6
a Round[8] : 2e428109 242df50b 43319145 b400275c

Theta[9] : 8f78675f 58608012 890c7818 c03e4243
Pi1[9] : 8f7867c5 b0c10024 218f0311 00f9090f
Gamma[9] : 60b9990b 7ec1f4e4 e04e991f aff967c5
Pi2[9] : 60b9990b 3f60fa72 ff0274c8 6bfe59f1
a Round[9] : 62856971 f6ea35eb 96398bf4 aa07866e

Theta[10] : 1dd2fc58 1503b450 826911d6 419d17d3
Pi1[10] : 1dd2fc15 2a0768a0 4d223ad0 06745f4d
Gamma[10] : 867c17cf b0a9cae2 c408598a 15d0d495
Pi2[10] : 867c17cf 5854e571 562042cc 45743525
a Round[10] : a4c4fdbf f9ceb5df 1f9fa71d ec9d758d

Theta[11] : fb17727a 94374db5 2b4b27d2 89179de1
Pi1[11] : fb1772d1 286e9b6b 6964fa45 245e7786
Gamma[11] : 04f8ed86 b2e79a3c 24a6fe00 d373e890
Pi2[11] : 04f8ed86 5973cd1e 012537f0 34dcfa24
a Round[11] : b758017d d932e69f d982d4d8 bceec1a3

Theta[12] : 79938840 e61016a9 7c4e52ee 8bbf2193
Pi1[12] : 79938898 cc202d53 89ca5dcf 2efc864e
Gamma[12] : a6fc876f 98492567 ca845339 f19385db
Pi2[12] : a6fc876f cc2492b3 ce542299 fc64e176
a Round[12] : 58aa3fc6 d40233a0 5b059557 ec315063

Theta[13] : db97e21d c747fdb0 b33f4787 f7078e75
Pi1[13] : db97e271 8e8ffb61 67e8f0f6 dc1e39d7
Gamma[13] : 54181d9e 8c0e34cf e8672659 dd1f1211
Pi2[13] : 54181d9e c6071a67 cf433932 7747c484
a Round[13] : 0d622e67 84a0935c fd3e05f6 3d935db9

Theta[14] : 297378b6 97642f3e b2285c2c 2624f1dd
Pi1[14] : 29737880 2ec85e7d 450b8596 9893c774
Gamma[14] : 949f8774 0e2ce51d fc4f5802 2d7b7c94
Pi2[14] : 949f8774 8716728e 17e27ac0 0b5edf25
a Round[14] : b0a387a6 e3561c6c 58d97502 676da1c1

Theta[15] : e9f4e271 fb756afa 6a891fde 773dc751
Pi1[15] : e9f4e26a f6ead5f5 5123fbcd dcf71d45
Gamma[15] : 1c171d85 d2eb19d7 e0e000e0 b9d633af
Pi2[15] : 1c171d85 e9758ceb 07070007 ee758ceb
a Round[15] : 50455341 4e205947 20524148 41534941

State : 50455341 4e205947 20524148 41534941

plainText : 504553414e2059472052414841534941
plainText : PESAN YG RAHASIA

LAMPIRAN B

Efek Avalanche dengan perubahan satu bit pada plaintext secara random

Kunci : **iiiiiiiiiii (eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee Hexa)**

No	Plaintext 1	Plaintext 2	prosentase
1	00000000000000000000000000000000	0000000000000000000000000000000 1	53.91
2	4142434445464748494a4b4c4d4e4f50	414 3 434445464748494a4b4c4d4e4f50	61.72
3	77424344453233333465666566653534	77 5 24344453233333465666566653534	50.78
4	6d6f68616d616420736261737469616e	6d6f68616d61642 1 736261737469616e	50
5	6d736261737469616e207769646f646f	6d736261737469616e2 1 7769646f646f	53.91
6	73656b6f6c61682074696e6767692074	5 3656b6f6c61682074696e6767692074	50
7	73656b6f6c61682074696e6767692074	736 7 6b6f6c61682074696e6767692074	54.69
8	74747274727435202020202079797579	7 6747274727435202020202079797579	54.69
9	00000000000000000000000000000000	10000000000000000000000000000000	49.22
10	00000000000000000000000000000000	000000 1 000000000000000000000000	53.91
11	11111111111111111000000000000000	1 01111111111111111000000000000000	46.09
12	00000000000000111111111111111111	000000000000000111111111111111 1 0	55.47
13	616a64686475383933373934756a7269	616 e 64686475383933373934756a7269	47.66
14	45d0bf100f07590bd9ef0fdc632d89cd	6 5d0bf100f07590bd9ef0fdc632d89cd	50
15	71d0bf100f07590bd9676867632d89cd	71d0bf100f07590bd9676867632d89 c f	50.78
16	bd9676867632d89cd71d0bf100f07590	B 99676867632d89cd71d0bf100f07590	48.44
17	efefefefefefefefefefefefefefefef	efefefefefefefefefefefefefef e cef	53.12
18	abababababababababababababababab	aaababababababababababababababab	54.69
19	cfefcfefcfefcfefcfefcfefcfefcfef	cfefcfefcfefcfef d cfefcfefcfefcfef	57.81
20	34356c6b6a3738757967686a68676a37	34356c6b6a373875 7 d67686a68676a37	50.78
21	74656b6e696b20656c656b74726f2073	74656b6e696b20656c6 1 6b74726f2073	50
22	56c656b74726f207374656b6e696b206	56c656b74726f207374656b6e69 6 9206	53.91
23	16e207769646f646f6d7362617374696	16e217769646f646f6d7362617374696	49.22
24	f100f07590b99676867632d89cd71d0b	f100f07590 b 9676867632d89cd71d0b	53.12
25	64748494a4b4c4d4e4f5041434344454	64748 6 94a4b4c4d4e4f5041434344454	51.56
26	967686a68676a3734356c6b6a3738757	967686a68676a3734356c6a6a3738757	56.25
27	a3738753967686a68676a3734356c6a6	a3738 3 57967686a68676a3734356c6a6	61.72

28	02020207979757974747274727435202	12020207979757974747274727435202	50
29	c61682074696e07473656b6f66767692	c61682074696e07473656b6f667676d2	55.47
30	6300f090bd9ef0f245d0b75f1d89cdde	6300f090bc9ef0f245d0b75f1d89cdde	50.78
31	011011110000000001110000110111e1	011011110000000001110000100111e1	56.25
32	deafdeafdeafdeadfdeafdeafdeafdea	deafdeafdeafdeadddeafdeafdeafdea	53.12
Rata-rata efek avalanche			52.78 %

Efek Avalanche dengan perubahan satu bit pada kunci secara random

Plaintext : *null string* (0000000000000000000000000000 Hexa)

No	Kunci 1	kunci 2	prosentase
1	00000000000000000000000000000000	00000000000000000000000000000001	57.03
2	4142434445464748494a4b4c4d4e4f50	4143434445464748494a4b4c4d4e4f50	50.78
3	77424344453233333465666566653534	77524344453233333465666566653534	50
4	6d6f68616d616420736261737469616e	6d6f68616d616421736261737469616e	52.34
5	6d736261737469616e207769646f646f	6d736261737469616e217769646f646f	53.12
6	73656b6f6c61682074696e6767692074	43656b6f6c61682074696e6767692074	60.94
7	73656b6f6c61682074696e6767692074	73676b6f6c61682074696e6767692074	49.22
8	74747274727435202020202079797579	76747274727435202020202079797579	49.22
9	00000000000000000000000000000000	10000000000000000000000000000000	53.12
10	00000000000000000000000000000000	00000001000000000000000000000000	53.12
11	1111111111111111110000000000000000	1011111111111111110000000000000000	50
12	00000000000000011111111111111111	00000000000000011111111111111110	51.56
13	616a64686475383933373934756a7269	616e64686475383933373934756a7269	46.09
14	45d0bf100f07590bd9ef0fdc632d89cd	65d0bf100f07590bd9ef0fdc632d89cd	54.69
15	71d0bf100f07590bd9676867632d89cd	71d0bf100f07590bd9676867632d89cf	50.78
16	bd9676867632d89cd71d0bf100f07590	B99676867632d89cd71d0bf100f07590	50.78
17	efefefefefefefefefefefefefefefef	efefefefefefefefefefefefefefefec	51.56
18	abababababababababababababababab	aaababababababababababababababab	53.91
19	cfefcfefcfefcfefcfefcfefcfefcfefcf	cfefcfefcfefcfefcfefcfefcfefcfefcf	56.25
20	34356c6b6a3738757967686a68676a37	34356c6b6a3738757d67686a68676a37	48.44
21	74656b6e696b20656c656b74726f2073	74656b6e696b20656c616b74726f2073	53.12
22	56c656b74726f207374656b6e696b206	56c656b74726f207374656b6e6969206	45.31
23	16e207769646f646f6d7362617374696	16e217769646f646f6d7362617374696	57.81
24	f100f07590b99676867632d89cd71d0b	f100f07590bb9676867632d89cd71d0b	49.22
25	64748494a4b4c4d4e4f5041434344454	64748694a4b4c4d4e4f5041434344454	51.56
26	967686a68676a3734356c6b6a3738757	967686a68676a3734356c6a6a3738757	49.22
27	a3738753967686a68676a3734356c6a6	a3738357967686a68676a3734356c6a6	59.38
28	02020207979757974747274727435202	12020207979757974747274727435202	51.56
29	c61682074696e07473656b6f66767692	c61682074696e07473656b6f667676b2	53.91

30	6300f090bd9ef0f245d0b75f1d89cdde	6300f090be9ef0f245d0b75f1d89cdde	53.12
31	011011110000000001110000110111e1	011011110000000001110000100111e1	58.59
32	deafdeafdeafdeafdeafdeafdeafdea	deafdeafdeafdeadddeafdeafdeafdea	50
Rata-rata efek avalanche			52.37 %

LAMPIRAN C

Percobaan Kesalahan Propagasi :

Isi *plaintext* sebesar satu blok (1024 bit) :

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF012
3456789ABCDEF0123456789ABCDEF0123456789ABCDEF012345
6789ABCDEF0123456789ABCDEF

Isi *Ciphertext* Hasil Enkripsi dengan kesalahan terjadi pada 128 bit pertama (blok awal) :

.txtÃp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ-Oeç¹oÏš Ãp~ô²ÿ-
Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ
-Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ -
Oeç¹oÏš }82) □\@;Ð□!€z;@

Simulasi Terjadinya Kesalahan :

.txtbitsalahOeç¹oÏš Ãp~ô²ÿ-Oeç¹oÏš Ãp~ô²ÿ -
Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ
-Oeç¹oÏš Ãp~ô²ÿ -Oeç¹oÏš Ãp~ô²ÿ -
Oeç¹oÏš }82) □\@;Ð□!€z;@

Hasil Dekripsi Setelah Terjadi Kesalahan :

ÎÅ□ê Y»ú E²çž2•S0123456789ABCDEF0123456789ABCDEF012345678
9ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCD
EF0123456789ABCDEF