Desain Algoritma Block Cipher Kadek

I Made Mustika Kerta Astawa
Badan Siber dan Sandi Negara
Jakarta
kadek19_kaptainboy@yahoo.com

Abstrak—Perkembangan Teknologi Informasi dan Komunikasi setidaknya dapat menimbulkan kerawanan dalam keamanan informasi, khususnya yang bersifat rahasia. Salah satu teknik perlindungan keamanan informasi dengan menggunakan algoritma enkripsi. TEA merupakan algoritma block cipher dengan kebutuhan penggunaan memori yang kecil dan mudah diimplementasikan kedalam piranti lunak. Berdasarkan standar keamanan NIST algoritma ini dikatakan sudah tidak memenuhi tingkat keamanan karena dalam prosesnya menggunakan ukuran blok 64 bit dan panjang kunci 128 bit. Pada tahun 2010, dibuat algoritma CFN yang beroperasi dalam ukuran 128 bit dan panjang kunci 256 bit berbasis Feistel Network. Akan tetapi, algoritma CFN masih ada beberapa kelemahan salah satunya pengaruh nilai delta sebagai nilai acak dan banyaknya siklus yang digunakan dalam penyandian data.Dimana nilai acak yang dihasilakn setiap siklus lebih besar. Untuk menutupi kelemahan tersebut, dalam penelitian ini kami mendesain algoritma sandi block cipher KADEK berbasis Feistel Network sebagai perbaikan dari algoritma CFN. Algoritma KADEK memiliki input teks terang sebesar 128 bit dan panjang kunci 256 bit. Menggunakan operasi XOR, penjumlahan modulo 2^{32}, left shift (pergeseran bit ke kiri), dan right shift (pergeseran bit ke kanan). Algoritma KADEK memiliki enam siklus dimana setiap satu siklus terdiri dari dua putaran. Dari hasil pegujian SAC dapat dinyatakan bahwa algoritma KADEK memiliki tingkat difusi baik (acak). Perubahan satu bit input, maka akan menyebabkan perubahan rata-rata mendekati setengah dari bit-bit output. Dengan kata lain jika satu bit input dikomplemenkan, maka setengah dari bit-bit output berubah)

Kata kunci— algoritma KADEK; Block Cipher; Feistel Network.

I. PENDAHULUAN

Perkembangan kriptografi dewasa ini didukung oleh berbagai macam algoritma kriptografi yang diciptakan oleh para ahli dan peneliti dengan tujuan agar informasi yang penting dan rahasia dapat diamankan baik dari sisi kerahasiaan, keaslian, keutuhan dan anti penyangkalan. Informasi tersebut diamankan melalui proses enkripsi dengan menggunakan algoritma kriptografi tertentu. Enkripsi merupakan proses menyembunyikan atau menyamarkan suatu informasi sedemikian rupa sehingga substansinya tidak dapat diketahui oleh pihak lawan [1]. Pada dasarnya algoritma kriptografi dibagi menjadi dua macam yaitu algoritma simetrik dan algoritma asimetrik. Sistem kripto simetrik dibagi menjadi dua kelas, yaitu *block cipher* dan *stream cipher. Block cipher* merupakan model penyandian dimana teks terang yang akan disandi dibagi kedalam blok-blok dengan panjang tertentu sehingga menghasilkan blok-blok teks sandi [2].

Arsitektur dari *block cipher* diklasifikasikan menjadi tiga kelompok utama yaitu struktur SPN (*Substitution Permutation Network*), DES struktur, dan struktur lainnya seperti *Square block cipher* [3]. Struktur berbasis *Feistel Network* merupakan kelas *block cipher* yang pada dasarnya menggunakan struktur algoritma penyandian *Feistel* yang diciptakan oleh Horst Feistel [4]. Pada tahun 1994, David Wheeler dan Roger Needham dari *Computer Laboratory*, Cambridge *University* England menciptakan suatu algoritma *block cipher* yang dinamakan *Tiny Encryption Algorithm* (TEA) [5]. Sistem penyandian TEA menggunakan proses *Feistel Network* dengan menambahkan fungsi matematik berupa penambahan dan pengurangan sebagai operator pembalik selain XOR. Hal ini dimaksudkan untuk menciptakan sifat nonlinearitas. Hal yang paling menonjol dari algoritma TEA adalah kesederhanaan implementasi, ketiadaan S-BOX maupun P-BOX, dan kecepatan yang tinggi. Sehingga TEA dikatakan memiliki beberapa keunggulan antara lain keringanan prosesnya, operasi-operasi yang digunakan hanya berupa operasi bit biasa tanpa substitusi, permutasi, ataupun operasi matriks. Terlepas dari keunggulan TEA tersebut, berdasarkan standar yang telah ditetapkan oleh NIST, algoritma ini dikatakan sudah tidak aman lagi karena dalam prosenya masih menggunakan ukuran blok 64 bit dan panjang kunci 128 bit yang tidak memenuhi tingkat keamanan sekarang ini. Pada Tahun 2010, I Made Mustika Kerta Astawa dan Sri Rosdiana dari Sekolah Tinggi Sandi Negara menciptakan suatu algoritma CFN (*Cipher Feistel Network*) yang dalam prosenya menggunakan ukuran blok 128 bit dan panjang kunci 256 bit [6]. Namun masih terdapat kelemahan diantaranya nilai delta yang digunakan sebagai nilai acak dan banyaknya siklus yang digunakan dalam penyandian data sehingga mempengaruhi performa waktu yang dibutuhkan dalam proses penyandian data. Untuk mengatasi permasalahan tersebut, peneliti mencoba mendesain suatu algoritma yang aman secara kriptografis dengan tetap memperhatikan kecepatan penyandiannya. Nilai delta merupakan nilai konstanta yang dapat mempengaruhi tingkat difusi suatu algoritma penyandian.

Salah satu tujuan dari dibuatnya makalah ini adalah mendesain algoritma KADEK yang aman secara kriptografis sehingga dapat dijadikan sebagai alternatif dalam penyandian data serta mempunyai kecepatan dalam proses penyandian data yang relatif cepat. Algoritma KADEK merupakan algoritma *block cipher* berbasis *Feistel Network* yang memiliki *input* blok teks terang dan *output* blok teks sandi sebesar 128-bit, serta memiliki *input* kunci sebesar 256-bit. Penelitian ini diharapkan dapat bermanfaat untuk memberikan inspirasi dan tambahan referensi pengetahuan bagi pembaca dalam mengembangkan ilmu persandian serta dapat memberikan kontribusi untuk

memajukan ilmu persandian khususnya pada perkembangan pembuatan desain algoritma *block cipher*.

II. TINJAUAN PUSTAKA

A. Block Cipher dan Feistel Network

Block cipher merupakan suatu teknik penyandian dimana teks terang yang akan disandi dibagi kedalam blok-blok dengan panjang tertentu sehingga menghasilkan blok-blok teks sandi [7].

Feistel Network merupakan suatu struktur penyandian yang menggabungkan prinsip konfusi dan difusi yang dilakukan secara berulang-ulang dalam sebuah sistem sandi dengan kombinasi yang berbeda-beda. Prinsip konfusi dan difusi pertama kali diperkenalkan oleh Claude Shannon, yang digunakan untuk menyulitkan usaha kriptanalisis pihak lawan melalui analisis statistik. Istilah *Feistel Network* sering dikenal juga dengan nama *Feistel Cipher*.

Di dalam *Feistel Network*, blok teks terang dibagi menjadi dua bagian yang sama besar. Kemudian salah satu blok dioperasikan dengan fungsi *F* dan diputar. Inti dari *Feistel Network* adalah fungsi *F*, yang memetakan sebuah *input* bit menjadi *output* bit. Untuk melakukan operasinya, fungsi *F* membutuhkan adanya kunci internal. Kunci internal ini dibangkitkan dari kunci eksternal yang sebelumnya telah didefinisikan. Kekuatan penggunaan *Feistel Network* ini berada pada putaran di fungsi *F*.

B. Fungsi F

Fungsi F merupakan inti dari arsitektur Feistel Network. Pada umumnya fungsi F tergantung pada fungsi matematis yang digunakan. Fungsi F dalam Feistel Network adalah fungsi nonlinear dan merupakan elemen konfusi. Definisi: Fungsi F dari sebuah Feistel Network konvensional dapat diekspresikan sebagai:

$$F : \{0,1\}^{n/2} \times \{0,1\}^{k} \rightarrow \{0,1\}^{n/2} \quad (1)$$

dimana pada definisi ini, *n* adalah ukuran blok. *F* adalah sebuah fungsi yang menggunakan *n*/2 bit dan *k*–bit dari kunci sebagai *input*, dan menghasilkan sebuah *output* dengan panjang *n*/2 bit [8]. Terdapat beberapa kriteria yang perlu diperhatikan dalam membangun suatu fungsi *F*, antara lain mempunyai sifat *Avalanche Criterion* (AC) dan *Strict Avalanche Criterion* yang baik.

C. Strict Avalanche Criterion

Konsep ini dikemukakan oleh Webster dan Tavares yang mencoba meneliti hubungan antara *completeness* dengan *avalanche* hingga akhirnya pada tahun 1985 mereka mampu menemukan suatu konsep baru yang lebih dikenal dengan istilah *Strict Avalanche Criterion* (SAC) [9]. Jika bit-bit *output* hanya bergantung pada sedikit bit-bit *input*, maka dengan mengamati pasangan *input-output* dalam jumlah yang signifikan (sama seperti pada *chosen plaintext attack*), dapat mendeteksi hubungan dan menggunakan informasi ini untuk mencari kunci. Oleh karena itu, terdapat sifat SAC yang merupakan gabungan antara AC dan *Completeness*. AC merupakan sifat yang jika terdapat perubahan satu bit *input*, maka akan menyebabkan perubahan rata-rata setengah dari bit-bit *output*. Dengan kata lain jika satu bit *input* dikomplemenkan, maka setengah dari bit-bit *output* berubah.

III. METODE

Pada penelitian ini, penulis menggunakan metode penelitian kepustakaan berupa deskripsi penelitian yang dihasilkan atas kajian referensi pustaka dan eksperimen. Tahapan proses kajian ini adalah sebagai berikut:

1. Pengumpulan data

   Melakukan pengumpulan referensi dari literatur terkait mengenai desain algoritma block cipher.

2. Desain

   Melakukan desain algoritma block cipher KADEK berbasis Feistel network. Desain dilakukan dengan perubahan nilai delta, an operasi matematik pada fungsi F algoritma CFN

3. Pengujian

   Melakukan pengujian algoritma block cipher KADEK. Pengujian dilakukan dengan uji SAC untuk mendapatkan hasil tingkat difusi (keacakan).

4. Analisis data

   Analisis hasil pengujian terhadap algoritma block cipher KADEK. Analisis idapat berdasarkan hasil uji setiap siklus algoritma KADEK dan elemen-elemen operasi matematik yang digunakan di fungsi F.

5. Pengambilan Kesimpulan

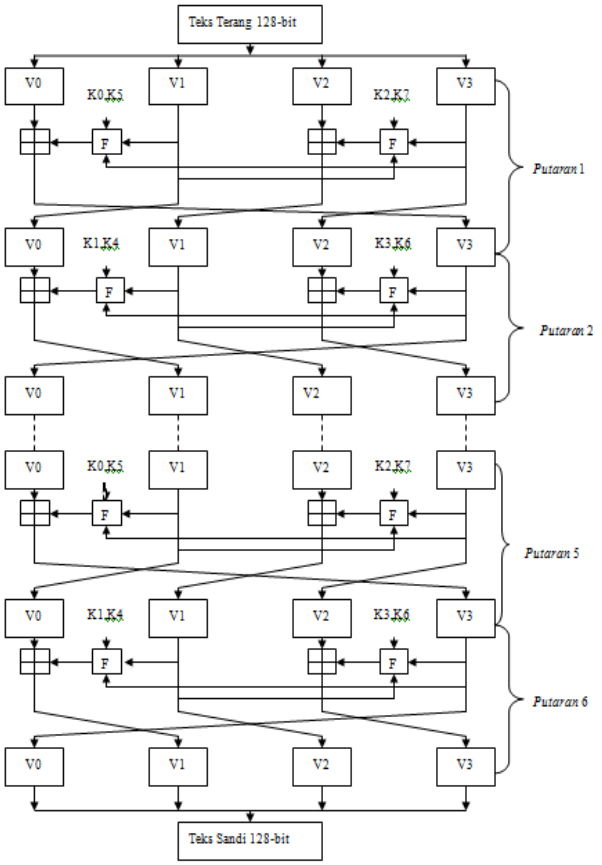
   Pengambilan simpulan hasil penelitian.

IV. HASIL DAN PEMBAHASAN

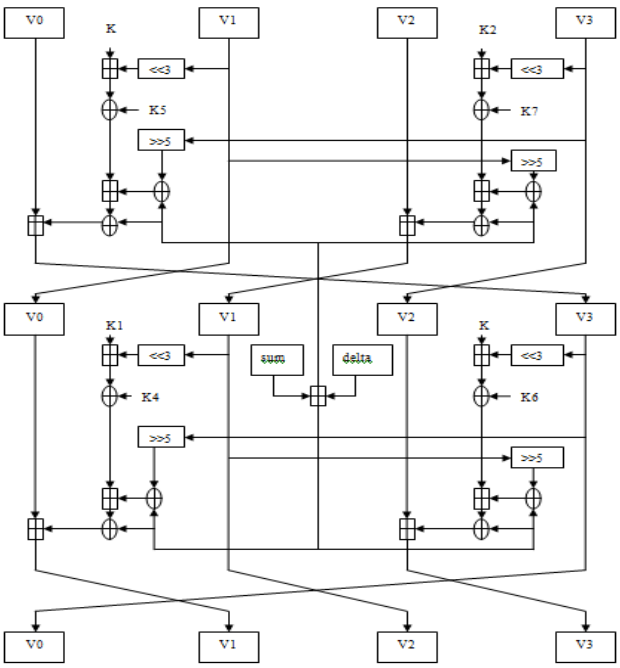
A. Deskripsi Algoritma KADEK

Algoritma KADEK mengadopsi nilai *delta* yang digunakan pada algoritma TEA yaitu 0xc6ef3720 untuk mencegah *output* yang tidak acak terhadap *input*-an yang ekstrim. Berbeda dengan algoritma TEA yang memproses 64-bit teks terang terhadap 64-putaran sekali waktu, algoritma KADEK di desain agar dapat memproses 128-bit *input* terhadap 12 putaran sekali waktu. Secara umum, arsitektur KADEK dapat dilihat pada Gambar 1. *Input* blok teks terang dari algoritma KADEK akan dibagi menjadi empat subblok (V_0, V_1, V_2, V_3) dimana masing-masing subblok berisi 32-bit teks terang yang terurut dari kiri ke kanan. Setiap subblok tersebut akan dioperasikan dengan fungsi *F* secara *Feistel*. Proses ini akan berulang terus menerus sampai pada putaran terakhir. *Output* subblok pada putaran terakhir (putaran-6) akan digabung menjadi 128-bit *output* blok teks sandi. Pada setiap putaran *i* (i=1,2,...,6) dalam algoritma KADEK, dua buah subblok yaitu V_1,V_3 akan menjadi *input* fungsi *F*. *Output* dari fungsi *F* akan dioperasikan penjumlahan modulo 2^{32} dengan subblok V_0 pada fungsi *F* sebelah kiri dan operasi penjumlahan modulo 2^{32} dengan subblok V_2 pada fungsi *F* sebelah kanan. Proses ini terus berulang sampai pada putaran terakhir. *Output* yang dihasilkan pada putaran terakhir

akan di gabung menjadi 128-bit *output* blok teks sandi. Lebih detilnya, alur pemrosesan dan struktur dari algoritma KADEK pada satu siklus dapat dilihat pada Gambar 2.



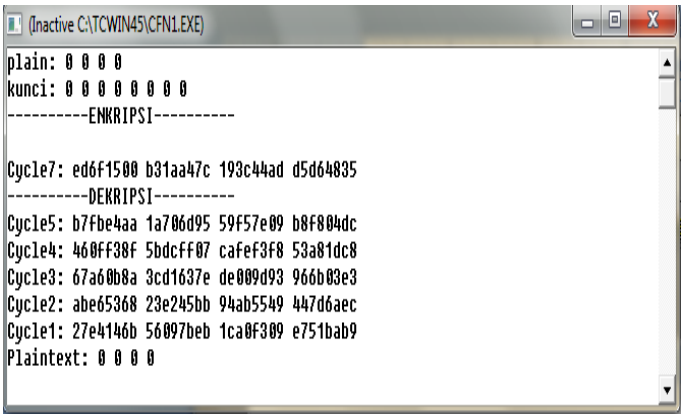
Gambar 1. Arsitektur algoritma KADEK



Gambar 2. Struktur satu siklus algoritma KADEK

B. *Proses Enkripsi dan Dekripsi*

Proses enkripsi pada algoritma KADEK diawali dengan input teks terang sebanyak 128-bit. Kemudian 128 bit teks terang tersebut dibagi menjadi empat subblok (V0, V1, V2, V3) masing-masing sebanyak 32-bit. Setiap subblok teks terang tersebut akan dioperasikan dengan fungsi F disetiap putarannya. Hasil penyandian satu blok teks terang (128-bit) menjadi blok teks sandi (128-bit) dalam satu siklus pada algoritma KADEK adalah dengan menggabungkan antara V0, V1, V2, dan V3. Untuk penyandian pada siklus berikutnya dilanjutkan proses seperti diatas sampai dengan 12-putaran. Sedangkan proses dekripsi algoritma KADEK pada prinsipnya sama halnya seperti pada proses enkripsi yang berbasis Feistel Network lainnya. Namun hal yang berbeda adalah penggunaan teks sandi sebagai input, dan output-nya adalah teks terang. Serta operator substraction sebagai operator pembalik terhadap addition. Contoh proses enkripsi dan dekripsi pada Algoritma KADEK dapat dilihat pada gambar 3.



Gambar 3. Proses enkripsi dan dekripsi

V. ANALISIS DESAIN ALGORITMA KADEK

Berikut adalah hasil analisis algoritma KADEK berdasarkan beberapa kriteria evaluasi desain block cipher :

A. *Ukuran Kunci*

KADEK memiliki ukuran kunci sebesar 256-bit untuk menyandi 128-bit teks terang. Menurut Zener pada tahun 2005 [10], ukuran ini cukup untuk memenuhi 128-bit tingkat keamanan dimana panjang kunci yang digunakan adalah sebanyak dua kali panjang teks terang yang disandi, sehingga akan mempersulit pihak lawan dalam melakukan *exhaustive key search attack* (*brute force attack*).

B. *Ukuran Blok*

Ukuran blok KADEK adalah 128-bit yang merupakan ukuran standar penyandian block cipher dewasa ini, seperti halnya AES. Operasi setiap subblok pada KADEK adalah sebesar 32-bit. Hal ini dapat memudahkan dan mempercepat proses perhitungan pada processor terutama prosesor 32-bit, karena dapat meminimalisir penggunaan memori pada saat pemrosesan.

C. *Jumlah Putaran*

Jumlah putaran yang digunakan pada algoritma KADEK adalah 12-putaran. Hal ini dimaksudkan untuk meningkatkan kecepatan proses enkripsi/dekripsinya. Perbandingan jumlah putaran pada algoritma KADEK berdasarkan uji SAC dapat dilihat pada Tabel 1.

TABEL I PERBANDINGAN JUMLAH PUTARAN ALGORITMA KADEK

Siklus	Waktu Enkripsi (detik)	SAC		KETERANGAN
		MIN	MAX	
1.	0.0022	0.00%	100.00%	TIDAK LULUS
2.	0.0038	0.00%	100.00%	TIDAK LULUS
3.	0.0057	0.00%	100.00%	TIDAK LULUS
4.	0.0070	31.24%	71.28%	TIDAK LULUS
5.	0.0093	49.21%	50.60%	LULUS
6.	**0.0117**	**49.15%**	**50.65%**	**LULUS**
7.	0.0122	49.25%	50.65%	LULUS
8.	0.0147	49.17%	50.75%	LULUS
9.	0.0159	49.28%	50.76%	LULUS
10	0.0176	49.15%	50.86%	LULUS
11	0.0214	49.20%	50.72%	LULUS
12	0.0228	49.26%	50.77%	LULUS
13	0.0238	49.25%	50.79%	LULUS
14	0.0263	49.27%	50.80%	LULUS
15	0.0287	49.17%	50.81%	LULUS
16	0.0292	49.29%	50.80%	LULUS

D. *Fungsi F Algoritma KADEK*

Pada algoritma KADEK terdapat fungsi *F* yang digunakan secara berulang di setiap putarannya. Operasi yang digunakan pada fungsi *F* adalah operasi XOR, penjumlahan modulo 2^{32}, dan operasi pergeseran bit (*shift*) serta nilai *sum*. Nilai *sum* didapat dari penjjumlahan *sum* awal dengan nilai *delta* (nilai acak pada algoritma TEA yaitu 0xc6ef3720). Nilai *sum* tersebut diperbarui pada setiap siklus.

E. *Algoritma Pembangkit Subkunci (Key Schedule)*

Key schedule pada algoritma KADEK cukup kompleks, hal ini dimaksudkan untuk meningkatkan keamanan dalam kesederhanaan desainnya. Kunci *K* sepanjang 256-bit dipecah menjadi delapan sub kunci *K*0, *K*1, *K*2, *K*3, *K*4, *K*5, *K*6, *K*7 dimana penjadwalan kuncinya dapat dilihat pada Tabel 2.

TABEL II. *KEY SCHEDULE* ALGORITMA KADEK

JENIS PUTARAN	SUB KUNCI YANG DIGUNAKAN
Putaran ganjil	*K*0, *K*2, *K*5, *K*7
Putaran genap	*K*1, *K*3, *K*4, *K*6

Pada setiap siklus, subkunci *K*0-*K*7 di-*update* menggunakan fungsi *G*. Untuk fungsi *G* yang digunakan pada *key schedule* algoritma KADEK adalah sebagai berikut:

K[0]^=(K[1]<<3^K[2]>>7)^sum+K[3]

K[1]^=(K[2]<<3^K[3]>>7)^sum+K[4]

K[2]^=(K[3]<<3^K[4]>>7)^sum+K[5]

K[3]^=(K[4]<<3^K[5]>>7)^sum+K[6]

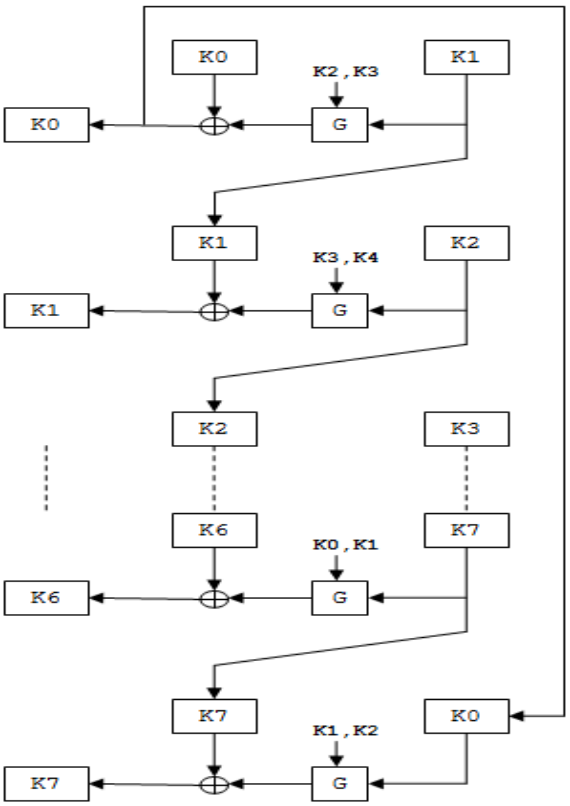
K[4]^=(K[5]<<3^K[6]>>7)^sum+K[7]

K[5]^=(K[6]<<3^K[7]>>7)^sum+K[0]

K[6]^=(K[7]<<3^K[0]>>7)^sum+K[1]

K[7]^=(K[0]<<3^K[1]>>7)^sum+K[2]

Secara umum, skema *key schedule* algoritma KADEK tampak pada gambar 4.



Gambar 4. *Key schedule* algoritma KADEK

F. *Perluasan Data*

Ukuran Input blok teks terang pada algoritma KADEK adalah 128-bit, dan blok outputnya adalah 128-bit teks sandi. Tidak terdapat perluasan data pada saat penyandian sehingga dapat dikatakan cukup baik untuk mencegah pihak lawan memperoleh informasi yang lebih mengenai teks terang jika diketahui teks sandinya.

G. *Propagasi Error*

Pada algoritma KADEK, proses dekripsi satu blok teks sandi yang di dalamnya terdapat bit *error* akan mengakibatkan teks terangnya memiliki nilai perubahan yang sangat jauh dari teks terang aslinya sebesar satu blok, hal ini dikarenakan hasil difusi bit yang cukup baik sehingga mengakibatkan jika terdapat satu bit saja yang error, maka akan menyebabkan bit teks terangnya juga akan berubah dengan probabilitas setengah.

VI. KESIMPULAN

Algoritma KADEK merupakan algoritma block cipher yang cukup baik, karena KADEK telah memenuhi beberapa

kriteria yang diperlukan untuk mengevaluasi bahwa suatu algoritma block cipher itu dapat dikatakan cukup baik. Algoritma KADEK secara keseluruhan memenuhi uji SAC dengan taraf error maximum sebesar 0,68 % yang berarti perubahan satu bit *input*, maka akan menyebabkan perubahan rata-rata mendekati setengah dari bit-bit *output*. Dengan kata lain jika satu bit *input* dikomplemenkan, maka setengah dari bit-bit *output* berubah. Sehingga dapat dikatakan algoritma KADEK memiliki difusi yang cukup baik (nilai keacakan yang sangat baik).

REFERENSI

[1] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C Second Edition. John Wiley & Sons, Inc. New York, 1996.

[2] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press LLC. Boca Raton, 1996.

[3] A. M. Youssef. *Analysis And Design of Block Cipher*. Queen's University. Canada, 1997.

[4] H. Feistel, "Cryptography and Computer Privacy," Scientific American, vol. 228, no. 5, May 1973

[5] D. J. Wheeler, and R. M. Needham, "TEA, a tiny encryption algorithm," *International Workshop on Fast Software Encryption*, pp. 363-366, 1994.

[6] I. M. M. K. Mustika and S. Rosdiana, "Desain Algoritma Block Cipher Feistel Network (CFN)," *Konferensi Nasional Matematika (KNM) XV*, pp. 141-152, 2010.

[7] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press LLC. Boca Raton, 1997.

[8] B. Schneier and J. Kelsey, "Unbalanced Block Cipher and Block Cipher Design," in: *Gollmann D. (eds) Fast Software Encryption. FSE 1996. Lecture Notes in Computer Science*, vol. 1039, pp. 121-144, 1996.

[9] A.F.Webster and S.E.Tavares, "*On the Design of S-Boxes", Advances in Cryptology*, Crypto'85, Lect. Notes. Comp. Sci:218, Springer Verlag, 1986

[10] A. K. Lenstra and A. R. Verheul, "Selecting Cryptographic Key Size," *Journal of Cryptography*, 2001.