Pencarian Karakteristik *Truncated Differential* pada 8 Iterasi Algoritme *Design* 32 Bit *Lightweight Block Cipher Algorithm* (DLBCA)

Muhamad Anis[1)], Nadia Paramita R. A.[2)]

(1) Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, muhamad.anis@bssn.go.id

(2) Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, nadia.paramita@bssn.go.id

Abstrak

Algoritme DLBCA merupakan algoritme block cipher dengan struktur Feistel yang memiliki plaintext sepanjang 32 bit dan kunci sepanjang 80 bit. Algoritme ini memiliki 15 iterasi. Salah satu pembuktian keamanan dari algoritme block cipher yaitu dengan melakukan serangan. Serangan diferensial telah diterapkan pada algoritme DLBCA dengan hasil jumlah S-box aktif cenderung lebih sedikit ketika dibandingkan dengan beberapa algoritme lainnya. Serangan diferensial dapat dikembangkan menjadi serangan truncated differential dengan menggabungkan n buah karakteristik truncated. Pada penelitian ini dilakukan pencarian karakteristik truncated untuk delapan iterasi algoritme DLBCA. Langkah yang dilakukan dalam pencarian karakteristik truncated diawali dengan menganalisis pengaruh komponen algoritme DLBCA terhadap nilai difference sampai dengan menemukan karakteristik truncated terbaik. Penentuan karakteristik truncated terbaik ini didasarkan pada probabilitas diferensial dari masing-masing karakteristik truncated. Adapun input pada karakteristik truncated yang dipilih adalah A_0={2,8,a} dan A_1={1,4,5}, serta banyak nibble aktif pada input difference adalah dua nibble. Pada penelitian ini terdapat dua karakteristik truncated terbaik untuk delapan iterasi algoritme DLBCA dengan probabilitas $2^{-67,0922814695}$. Probabilitas terbaik untuk empat iterasi adalah $2^{-33,9744694257}$ dan telah melibihi kompleksitas brute force attack yaitu 232. Berdasarkan hal tersebut maka algoritme ini tahan terhadap serangan truncated mulai dari empat iterasi. Serangan truncated dapat diterapkan pada empat iterasi algoritme DLBCA menggunakan distinguisher tiga iterasi. Pasangan plaintext dan ciphertext yang dibutuhkan untuk menerapkan serangan truncated differential pada empat iterasi algoritme DLBCA yaitu c.222,4682840384. Serangan ini mengembalikan subkunci sebanyak 16 bit.

Kata kunci: Algoritme DLBCA, block cipher, karakteristik truncated, serangan diferensial, serangan truncated differential.

1. PENDAHULUAN

Lightweight cryptography adalah suatu subbidang keilmuan yang merupakan gabungan dari ilmu elektro, kriptografi, dan komputer yang berfokus kepada desain algoritme baru dan implementasi yang efisien [1]. Tujuan dari *lightweight cryptography* adalah menciptakan layanan kriptografi dengan tingkat keamanan tinggi namun memiliki komputasi yang relatif kecil. Adapun salah satu jenis dari algoritme *lightweight cryptography* adalah algoritme *lightweight block cipher* [2].

Block cipher adalah suatu fungsi yang memetakan n bit teks terang ke n bit teks sandi dengan bit-bit tersebut disusun dalam bentuk blok dengan panjang n. Kunci yang digunakan pada *block cipher* adalah kunci yang diambil secara acak dari himpunan semua kemungkinan kunci. Fungsi yang digunakan pada *block cipher* adalah fungsi satu-satu. Fungsi ini digunakan pada proses dekripsi dan enkripsi sehingga *plaintext* dapat diperoleh setelah proses dekripsi dan enkripsi [3]. Beberapa contoh dari *lightweight block cipher* yaitu: SIMON, SPECK, SIMECK, TEA, XTEA, DLBCA dan lainnya [2].

Sufyan Salim Mahmood AlDabbagh [4] mengajukan sebuah algoritme *lightweight block cipher* dengan menerapkan struktur Feistel yaitu algoritme *Design 32-bit Lightweight Block Cipher Algorithm* (DLBCA). Algoritme ini memiliki dua buah *input* yaitu 80 bit kunci dan 32 bit *plaintext* serta satu buah *output* yaitu 32 bit *ciphertext*. Banyak iterasi yang dioperasikan sebanyak 15 iterasi dengan setiap iterasinya menerapkan operasi yang sama. Adapun operasi pada setiap iterasinya meliputi permutasi, XOR dengan kunci, S-box, rotasi, serta pembaruan kunci. Pada makalah yang sama AlDabbagh melakukan serangan diferensial pada algoritme DLBCA. Pada serangan tersebut diperoleh bahwa algoritme ini memiliki banyak S-box aktif lebih sedikit ketika dibandingkan dengan beberapa algoritme lainnya.

Lars R. Knudsen mengembangkan sebuah serangan baru pada *block cipher* yaitu *truncated differential cryptanalysis*. Serangan ini merupakan pengembangan dari serangan diferensial. Perbedaan *truncated differential* dengan serangan diferensial adalah pada *truncated differential* dicari n *truncated characteristic* yang hanya memprediksi bagian dari *difference* dalam pasangan teks tiap iterasi pada proses enkripsi. *Truncated differential* sendiri merupakan kumpulan dari *truncated characteristic* [5].

Ridwan Imam Syarif [6] telah melakukan serangan diferensial pada algoritme DLBCA. Pada penelitian tersebut dilakukan serangan untuk iterasi penuh algoritme DLBCA dengan pembatasan pada *input* yang digunakan. *Input* yang digunakan adalah *input* dengan 1, 2, 3, dan 4 *nibble* aktif. Pembatasan *nibble* aktif diterapkan supaya kemungkinan *input*

yang digunakan tidak terlalu banyak. Hasil yang didapatkan adalah jumlah minimal S-box aktif dari algoritme DLBCA lebih sedikit ketika dibandingkan dengan hasil yang diperoleh AlDabbagh [4]. Berdasarkan latar belakang tersebut maka dalam penelitian ini dilakukan pencarian karakteristik *truncated differential* pada delapan iterasi algoritme DLBCA. Beberapa hasil dari penelitian ini yaitu dapat ditemukan karakteristik *truncated* untuk delapan iterasi algoritme DLBCA serta perhitungan kompleksitas serangan *truncated.*

2. LANDASAN TEORI

2.1. *Block Cipher*

Block cipher merupakan algoritme enkripsi dengan menggunakan kunci simetrik yang menyediakan layanan kerahasiaan. *Block cipher* mengoperasikan satu blok *plaintext* secara keseluruhan dan menghasilkan satu blok *ciphertext* dengan panjang yang sama [5].

Secara umum *block cipher* harus memenuhi dua sifat yaitu konfusi dan difusi. Konsep konfusi dan difusi ini diperkenalkan oleh Shannon [8] yang ditujukan untuk menjadi dasar dalam konstruksi suatu *block cipher*. Difusi adalah sebuah keadaan yang mengakibatkan struktur statistik dari sebuah *plaintext* dapat tersebar merata pada struktur statistik *ciphertext*. Hal ini tercapai apabila setiap nilai dari *plaintext* mempengaruhi banyak nilai dari *ciphertext* atau sama dengan apabila masing-masing nilai dari *ciphertext* dipengaruhi oleh banyak nilai *plaintext*. Sifat ini dapat dipenuhi dengan cara membuat hubungan statistik antara *plaintext* dan *ciphertext* menjadi lebih kompleks sehingga dapat mempersulit usaha untuk memperoleh atau menebak nilai kunci *block cipher*. Tujuan dari konfusi adalah untuk membuat hubungan antara statistik dari *ciphertext* dan nilai dari kunci *block cipher* menjadi lebih kompleks sehingga dapat menghindari kemungkinan penyerang dapat memperoleh nilai kunci. Hal tersebut dapat tercapai dengan penggunaan algoritme substitusi yang kompleks [7].

Berdasarkan struktur dari proses enkripsi maka *block cipher* dapat dibedakan menjadi tiga jenis yaitu: Substitution Permutation Network (SPN), Feistel, dan struktur yang tidak termasuk keduanya. Feistel adalah struktur algoritme enkripsi dengan *input* merupakan *plaintext* yang memiliki panjang blok 2w dan kunci K. *Plaintext* ini selanjutnya dibagi menjadi dua blok, yaitu $L0$,$R0$. Kedua blok ini akan melewati proses enkripsi sampai dengan beberapa iterasi. Pada struktur ini operasi substitusi hanya dilakukan untuk satu blok *plaintext* [7]. Salah satu contoh dari algoritme *block cipher* dengan struktur Feistel adalah DLBCA.

2.2. Algoritme DLBCA

DLBCA merupakan algoritme yang dibuat oleh AlDabbagh pada tahun 2017 dan dipublikasikan pada *International Journal of Computer Applications*. Algoritme ini tergolong sebagai *lightweight block cipher* sehingga algoritme ini sangat disarankan untuk diimplementasikan ke dalam perangkat keras. Sebagai contoh adalah penerapan pada kartu kredit, RFID maupun *e-passport*. Algoritme ini memproses blok *plaintext* sebanyak 32 bit dengan kunci sebanyak 80 bit. Algoritme DLBCA menggunakan struktur Feistel dengan 15 iterasi. Keseluruhan penjelasan mengenai algoritme DLBCA diambil dari penelitian yang dilakukan oleh AlDabbagh.

Algoritme DLBCA juga menyertakan algoritme pembangkitan kunci. Proses pembangkitan kunci merupakan proses yang tergolong penting dalam algortime *block cipher*. Hal tersebut dikarenakan dengan adanya pembangkitan kunci maka akan mengefisienkan data yang dikirimkan. Kunci master yang digunakan berukuran 80 bit dengan notasi $k0$,$k1$,$k2$,…..,$k79$ yang selanjutnya dimasukan ke dalam algoritme pembangkitan kunci. Proses pembangkitan kuncinya adalah sebagai berikut:

a. Memasukan bit kunci ke-1, 2, 3, 4 ke dalam S-box dengan susunan $[K3,K2,K1,K0]=S[K3,K2,K1,K0]$.
b. Melakukan pergeseran ke kiri kunci sebanyak 13 kali
c. $[K0,K1,K2,K3,….,K79]=[K13,K14,K15,K16,….,K12]$.
d. Rotasi ke kiri kunci *seed* yang disimbolkan menggunakan $\lll$ sejumlah P-bit dengan nilai awal P adalah 13. Nilai P untuk iterasi berikutnya akan ditambah dengan 2.
e. $[K0,K1,K2,K3,….,K79]\lll 13+2i$ dengan i merupakan kunci yang sedang dijalankan.

Proses pertama dari algoritme adalah melakukan XOR antara 32 bit *plaintext* dengan 32 bit kunci. Kunci ini merupakan turunan dari kunci master, proses ini dilakukan dengan cara mengambil 32 bit paling kanan dari kunci master untuk dilakukan XOR dengan 32 bit *plaintext*. Selanjutnya dilakukan fungsi dioperasikan sebanyak 15 iterasi dan langkah terakhir adalah melakukan XOR antara *ciphertext* dengan subkunci terakhir.

2.3. *Truncated Differential*

Serangan diferensial merupakan serangan yang tergolong *chosen plaintext attack*. Pada *chosen plaintext attack* penyerang dapat memilih *input* dan *output* yang sesuai dengan syarat yang telah ditentukan, sehingga dapat dilakukan penurunan kunci atau ekstraksi kunci. Selanjutnya agar dapat dilakukan konstruksi serangan diferensial dari *input difference* yang direpresentasikan menjadi ΔX dan *output difference* yang direpresentasikan menjadi ΔY maka perlu dibentuk sebuah karakteristik diferensial yang memiliki probabilitas tinggi. Karakteristik diferensial sendiri merupakan urutan dari *input difference* dan *output difference* pada sebuah iterasi sehingga *output difference* pada satu iterasi akan berkorespondensi dengan *input difference* pada iterasi berikutnya [9].

Supaya dapat dibentuk karakteristik diferensial maka perlu dilakukan analisis terhadap S-box agar didapatkan karakteristik diferensial yang memiliki pasangan *input* dan *output difference* $(\Delta X,\Delta Y)$ dengan probabilitas tinggi. Keseluruhan kemungkinan probabilitas dari pasangan *difference* $(\Delta X,\Delta Y)$ tersebut dapat dilihat pada tabel *Difference Distribution Table* (DDT). Tabel DDT merupakan tabel yang berisi distribusi dari *input difference* dan *output difference* untuk semua kemungkinan pasangan pada S-box. Pada tabel ini setiap baris menunjukan masing-masing *input difference*, dan setiap kolom menunjukan masing-masing *output difference* yang berkorespondensi. Setiap nilai dari tabel DDT merupakan perhitungan dari jumlah kemungkinan pasangan setiap *input difference* dan *output difference* [10].

Truncated differential merupakan pengembangan dari serangan diferensial. Serangan ini pertama kali diperkenalkan oleh Lars Knudsen pada tahun 1994. *Truncated differential* hanya akan memprediksi beberapa bit pada sebuah pasangan *difference* $(\Delta X,\Delta Y)$ di setiap iterasinya. Berdasarkan pengertian *truncated differential* tersebut sehingga apabila (a,b) adalah diferensial pada iterasi ke-i, jika a' merupakan bagian dari a dan b' merupakan bagian dari b, maka (a',b') adalah *truncated differential* pada iterasi ke-i. Pada kondisi tersebut maka dapat dipilih suatu *output difference* sedemikian hingga dapat dijadikan sebagai *input difference* dari iterasi. Knudsen [5] memberikan penjelasan secara umum mengenai *truncated differential* ini dalam sebuah teorema berikut.

Teorema 1 Misalkan terdapat sebuah fungsi $f(x,k)$:$GF(2n)\times GF(2n)\rightarrow GF(2n)$ yang merupakan fungsi iterasi dari 5-iterasi Feistel memiliki ukuran blok adalah $2n$ bits menggunakan 5 kunci sesi, dengan setiap kunci sesi memiliki ukuran n bits. Misalkan $\alpha\neq 0$ adalah *input difference* yang merupakan sebagian W dari semua kemungkinan *output difference*. Selanjutnya serangan *truncated differential* menggunakan *chosen plaintexts* memiliki kompleksitas $2L$ dan *running time* $L\times 22n$. L merupakan bilangan bulat terkecil sedemikian sehingga $(W)L<2-2n$. Nilai dari L akan mendekati $2n+1$.

Knudsen [5] juga menjelaskan rangkuman mengenai langkah-langkah serangan *truncated differential* sebagai berikut:

a. Misalkan α merupakan nilai *difference* non-trivial dari dua buah *input* yang akan dimasukan ke dalam fungsi f, untuk setiap subbagian W dari keseluruhan *output difference* yang mungkin terjadi.
b. Menghitung nilai dari DDT dengan cara $i=0,\ldots,2n-1,T[f(i)\oplus f(i\oplus\alpha))=1$.
c. Memilih *plaintext* $P1$ secara acak dan menghitung $P2=p1\oplus(\alpha||0)$.
d. Menghitung nilai enkripsi $C1$ dan $C2$ dari $P1$ dan $P2$.
e. Untuk setiap nilai dari $k5$ yang berasal kunci sesi $RK5$ dilakukan:
   a) Mendekripsi *ciphertext* $C1,C2$ sejumlah satu iterasi menggunakan $k5$. Sehingga didapatkan nilai *ciphertext* $D1,D2$.
   b) Untuk setiap nilai dari $k4$ dari kunci sesi $RK4$ dilakukan:
      (1) Menghitung $ti=f(DiR\oplus k4)$ untuk $i=1,2$
      (2) Jika $T[t1\oplus t2\oplus D1L\oplus D2L]>0$, maka keluarkan nilai $k5$ dan $k4$.

3. METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini adalah metode kajian kepustakaan dan metode eksperimen. Kajian kepustakaan dilakukan melalui studi literatur pada makalah, buku, jurnal, dan sumber lain yang dapat mendukung pelaksanaan penelitian ini. Pengkajian ini bertujuan untuk memperoleh pemahaman terkait teori *block cipher*, algoritme DLBCA, *differential cryptanalysis*, dan *truncated differential cryptanalysis*.

Setelah dilakukan kajian kepustakaan maka dapat dilakukan eksperimen yang dimulai dengan mengobservasi komponen-komponen pada algoritme DLBCA. Selanjutnya dilakukan pembentukan karakteristik satu iterasi menggunakan komponen-komponen algoritme yang memiliki pengaruh terhadap nilai *difference*. Setelah ditemukan karakteristik untuk satu iterasi, pencarian dilanjutkan untuk menemukan *truncated differential* delapan iterasi algoritme DLBCA. Apabila sudah ditemukan truncated differential untuk delapan iterasi algoritme DLBCA maka dapat dihitung probabilitas dilakukan serangan pada algoritme DLBCA. Perhitungan probabilitas ini menggunakan perhitungan nilai *input* yang mengaktifkan komponen nonlinier pada algoritme DLBCA yaitu S-box. Probabilitas dari S-box dapat direpresentasikan dalam DDT. Hal ini dapat memudahkan ketika dilakukan perhitungan probabilitas serangan. Setelah ditemukan probabilitas dilakukan serangan pada karakateristik yang telah ditemukan, selanjutnya dapat ditemukan kompleksitas dalam melakukan serangan.

4. HASIL DAN PEMBAHASAN

4.1. Pengaruh Komponen Algoritme DLBCA Terhadap Nilai *Difference*

Analisis komponen merupakan sebuah tahapan yang dibutuhkan sebelum melakukan pencarian karakteristik *truncated differential*. Analisis ini dibutuhkan karena beberapa komponen dari suatu algoritme mempengaruhi nilai *input difference* sehingga nantinya akan dihasilkan nilai *output difference* yang berbeda dari nilai *input difference* yang digunakan.

Nilai *difference* merupakan hasil dari operasi dua

buah nilai yang dapat dipengaruhi oleh komponen pada suatu algoritme. Pengaruh ini terjadi apabila terdapat dua buah nilai *input* algoritme yaitu X dan X' yang ketika dioperasikan menghasilkan suatu *input difference* $\Delta X=X\oplus X'$.

Kedua nilai *input* tersebut ketika dijalankan pada suatu algoritme akan menghasilkan dua buah nilai *output* yaitu Y dan Y'. Ketika kedua nilai *output* tersebut dioperasikan maka dihasilkan suatu nilai *output difference* $\Delta Y=Y\oplus Y'$. Komponen pada suatu algoritme disebut mempengaruhi nilai *difference* apabila nilai $\Delta X\neq\Delta Y$. Sebaliknya apabila nilai $\Delta X=\Delta Y$ maka komponen pada algoritme tidak mempengaruhi nilai *difference*.

Komponen dari algoritme DLBCA adalah sebanyak enam komponen yaitu XOR kunci, fungsi S-box, fungsi permutasi, rotasi, XOR kanan dan kiri, serta *swap*. Sebelum dilakukan pencarian karakteristik pada algoritme DLBCA terlebih dahulu dianalisis keseluruhan komponen algoritme DLBCA. Berdasarkan analisis yang telah dilakukan didapatkan hasil terkait komponen-kompen yang berpengaruh pada nilai *difference*. Hasil analisis ini dapat dilihat pada Tabel 1.

Tabel 1 pengaruh komponen algoritme DLBCA terhadap nilai *difference*

Komponen	Pengaruh
XOR Kunci	Tidak berpengaruh
Fungsi *S-box*	Berpengaruh
Fungsi Permutasi	Berpengaruh
Fungsi Rotasi	Berpengaruh
Fungsi XOR Kanan dan Kiri	Berpengaruh
Fungsi *Swap*	Berpengaruh

4.2. Pencarian Karakteristik *Truncated* Berdasarkan Pengaruh Komponen Algoritme DLBCA

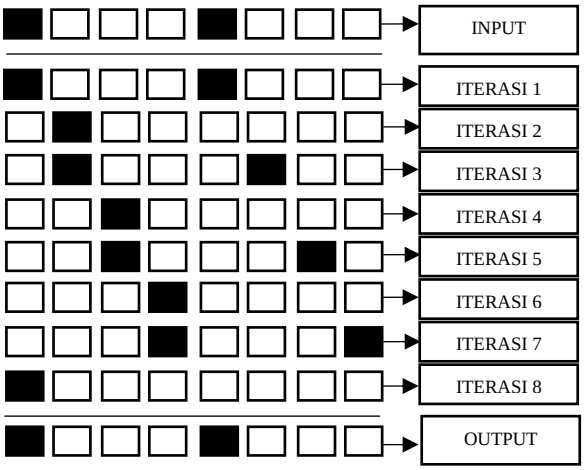
Pada analisis sebelumnya telah diperoleh komponen-komponen algoritme DLBCA yang memiliki pengaruh terhadap nilai *difference*. Setelah hasil analisis tersebut diperoleh maka dapat dilakukan observasi terhadap komponen-komponen tersebut. Observasi ini bertujuan untuk mendapatkan karakteristik *truncated* yang memiliki nilai probabilitas tinggi.

Pada proses observasi ini komponen-komponen algoritme DLBCA dikelompokkan menjadi dua. Kelompok pertama yaitu komponen linier meliputi permutasi, rotasi, XOR kanan dan kiri, serta *swap*. Kelompok kedua adalah komponen nonlinier pada algoritme DLBCA yaitu S-box. Berdasarkan observasi ini didapatkan dua buah sifat. Sifat ini yaitu Sifat 1 dan Sifat 2.

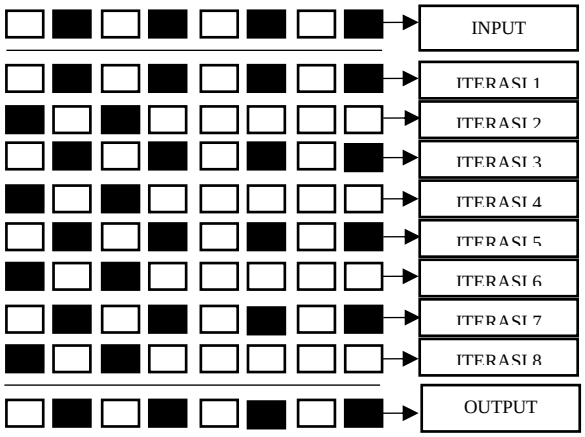
Sifat 1. Misalkan *input* permutasi yang direpresentasikan ke dalam empat buah *nibble* yaitu $M0,M1,M2,M3$. Apabila hanya terdapat dua *nibble* aktif yaitu Mi, $Mj\in A0=\{2,8,a\}$ atau $A1=\{1,4,5\}$ dengan $j=(i+2) \bmod 4$, maka dihasilkan *output* dua *nibble* aktif.

Sifat 2. Banyaknya *nibble* aktif pada *output* sama dengan *Hamming weight* dari *nibble* aktif pada *input* .

Berdasarkan observasi yang sudah dilakukan digunakan dua buah pola untuk membangun karakteristik *truncated* delapan iterasi. Ke dua pola tersebut ditunjukkan pada Gambar 1 dan Gambar 2.



Gambar 1. karakteristik *input* dua *nibble* aktif



Gambar 2. karakteristik *input* empat *nibble* aktif

Setelah dilakukan observasi pada komponen linier diperoleh bahwa karakteristik-karakteristik yang dapat mengaktifkan sedikit S-box adalah karakteristik yang pada *input difference*-nya memiliki *nibble* aktif dengan *Hamming weight* satu atau dua. Contoh dari karakteristik ini dapat dilihat pada Gambar 1 dan Gambar 2. Berdasarkan kondisi *input difference* yang memiliki *nibble* aktif dengan *hamming weight* satu atau dua maka terdapat beberapa kemungkinan yang dapat dijadikan sebagai karakteristik *truncated*.

Beberapa kemungkinan ini yaitu *input difference* dengan satu *nibble* aktif yang disimbolkan dengan S_1, kondisi pertama dari *input difference* dengan dua *nibble* aktif yang disimbolkan dengan S_2, dan kondisi ketiga *input difference* dengan empat *nibble* aktif yang disimbolkan dengan S_3. Alasan digunakan kemungkinan S_1 dikarenakan kemungkinan ini memiliki jumlah *nibble* aktif paling sedikit dibandingkan pola lainnya. Kemungkinan ini juga

sudah mewakili kemungkinan-kemungkinan *input difference* dengan dua, tiga, dan empat *nibble* aktif yang iterasi kedua sampai dengan kedelepan hanya terdapat satu atau dua *nibble* aktif.

Selanjutnya alasan dipilih kemungkinan $S2$ dan $S3$ karena dua kemungkinan tersebut memiliki karakteristik yang dapat mempertahankan jumlah S-box aktif dan memiliki jumlah S-box aktif yang sedikit lebih banyak. S-box aktif pada kemungkinan ini yaitu 24 S-box aktif aktif. $S2$ dan $S3$ digunakan apabila saat observasi komponen nonlinier kemungkinan $S1$ tidak dapat diterapkan.

Alasan lain dari digunakannya dua kemungkinan ini karena kedua kemungkinan ini memiliki hubungan pada nilai *input* dan *output* nya. Apabila digunakan untuk mengkonstruksi karakteristik *truncated* dengan jumlah iterasi genap maka penempatan kemungkinan kedua dan ketiga tidak berpengaruh. Selanjutnya apabila kedua kemungkinan ini digunakan untuk mengontruksi karakteristik *truncated* dengan jumlah iterasi ganjil maka kemungkinan kedua harus digunakan pada iterasi pertama untuk meminimalisir jumlah S-box aktif.

Berdasarkan Tabel DDT tersebut dapat dilihat bahwa semua pemetaan dari *input difference* yang memiliki *Hamming weight* satu ke *output difference* yang memiliki *Hamming weight* satu yaitu 1, 2, 4, dan 8 memiliki nilai probabilitas 0 atau $P(S1 \rightarrow S1)=0$ keadaan tersebut menyebabkan kemungkinan $S1$ pada *input difference* tidak dapat digunakan.

S-box pada algoritme DLBCA dioperasikan sebelum fungsi permutasi. Oleh sebab itu *output difference* dari fungsi S-box harus merupakan nilai pada kemungkinan kondisi pertama dari kemungkinan $S2$ atau kondisi ketiga dari kemungkinan $S3$.

Sifat dari *input difference* yang dapat digunakan pada kemungkinan S_1 sudah jelas yaitu *nibble* aktif harus memiliki *Hamming weight* sejumlah satu buah. Pada kemungkinan S_2 dan kemungkinan S_3 terdapat beberapa kondisi. Misalkan terdapat dua buah himpunan *input difference* yaitu $A_0 = \{2,8,a\}$ dan $A_1 = \{1,4,5\}$ serta terdapat dua buah pola yaitu $(M_0, 0, M_1, 0)$ yang kemudian dapat disebut H_0 dan $(0, M_0, 0, M_1)$ yang kemudian dapat disebut H_1 .

4.3. Konstruksi Karakteristik Truncated Untuk 8 Iterasi Algoritme DLBCA

Pada iterasi pertama algoritme DLBCA digunakan *input* dengan dua buah *nibble* aktif yang merupakan kondisi $S2$. Karakteristik *truncated* ini terdiri atas empat buah kemungkinan pola. Hal tersebut ditujukan supaya dapat diteruskan menjadi karakteristik untuk iterasi selanjutnya maka masing-masing pola akan menghasilkan dua buah kemungkinan *output* .

Hasil dua buah kemungkinan *output* ini didapatkan karena adanya operasi pada fungsi S-box. Pada operasi fungsi S-box akan terdapat dua kemungkinan pemetaan yaitu terhadap nilai dari himpunan A_0 atau nilai dari himpunan A_1. Proses ini dilakukan sampai dengan iterasi ke delapan. Setelah dilakukan konstruksi didapatkan kompleksitas serangan *truncated differential* pada delapan iterasi Algoritme DLBCA. Kompleksitas ini ditunjukkan pada Tabel 2.

Tabel 2. Kompleksitas serangan masing-masing iterasi

Iterasi	**1**	**2**	**3**	**4**
Kompleksitas	$c.2^{5,10691}$	$c.2^{12,3293}$	$c.2^{22,4682}$	$c.2^{33,9744}$
Iterasi	5	6	7	8
Kompleksitas	$c.2^{38,9577}$	$c.2^{50,4639}$	$c.2^{55,4182}$	$c.2^{67,0922}$

5. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan didapatkan beberapa hasil, yaitu:

a. Hasil dari analisis komponen linier pada empat kemungkinan *input* yaitu kemungkinan *input* dengan 1, 2, 3, dan 4 *nibble* aktif adalah ditemukan dua buah kemungkinan *input difference* yang dapat digunakan untuk membangun karakteristik *truncated*. Karakteristik truncated dapat dibangun dari dua kemungkinan tersebut apabila nilai *input* yang digunakan merupakan bagian dari himpunan $A0=\{2,8,a\}$ atau $A1=\{1,4,5\}$. Adapun total karakteristik *truncated* yang dapat dibangun untuk delapan iterasi algoritme DLBCA adalah 1024 karakteristik *truncated*. Dari 1024 karakteristik *truncated* ditemukan dua buah karakteristik.
b. Setelah dilakukan penggabungan pada karakteristik yang memiliki nilai *input* dan *output difference* yang sama maka dihasilkan probabilitas terbaik dari karakteristik delapan iterasi algoritme DLBCA adalah $2^{-67,0922814695}$ sehingga dapat dihitung kompleksitas serangannya adalah c.267,0922814695. Adapun probabilitas dari karakteristik *truncated* untuk empat iterasi algoritme DLBCA adalah $2^{-33,9744694257}$. Usaha yang dibutuhkan untuk menyerang algoritme DLBCA menggunakan *distinguisher* empat iterasi sudah melebihi usaha apabila dilakukan *brute force attack* yaitu c. 232. Berdasarkan hal tersebut maka algoritme DLBCA tahan terhadap serangan *truncated* mulai dari empat iterasi maupun pada iterasi kedelapan atau iterasi penuh.
c. Berdasarkan perhitungan kompleksitas yang telah dilakukan maka serangan *truncated differential* dapat dijalankan apabila menggunakan *distinguisher* tiga iterasi algoritme DLBCA. Pada *distinguisher* tiga iterasi ini dapat diterapkan untuk melakukan serangan *truncated* pada algoritme DLBCA. Serangan ini diterapkan untuk memulihkan 16 bit subkunci. Kompleksitas data dari serangan ini adalah 222,4682840384. Berdasarkan kompleksitas tersebut, dibutuhkan

pasangan *plaintext* dan *ciphertext* yang berkorespondensi dengan banyak 222,4682840384. Pasangan *plaintext* dan *ciphertext* ini digunakan untuk mendapatkan pasangan *plaintext* dan *ciphertext* yang sesuai dengan *distinguisher* untuk tiga iterasi algoritme DLBCA.

REFERENSI

[1] A. J. Menezes, P. C. v. Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, USA: CRC press, Inc. Boca Raton, FL, 1996.

[2] A. Y. Poschmann, "LIGHTWEIGHT CRYPTOGRAPHY Cryptographic Engineering for a Pervasive World," Faculty of Electrical Engineering and Information Technology Ruhr-University, Bochum, 2009.

[3] A. Biryukov and L. Perrin, "State of the Art in Lightweight Symmetric Cryptography," 2017.

[4] L. R. Knudsen, "Truncated and Higher Order Differentials," Springer, pp. 196-211, 1994.

[5] W. Stallings and M. P. Tahiliani, Cryptography and network security principles and practice, London: Pearson, 2014.

[6] S. S. M. AlDabbagh, "Design 32-bit Lightweight Block Cipher Algorithm (DLBCA)," International Journal of Computer Application, 2017.

[7] H. M. Heys, "A Tutorial on Linear and Differential Cryptanalysis," Cryptologia, pp. 189-221, 2002.

[8] E. Biham and A. Shamir, "Differential Cryptanlysis of DES-like Cryptosystems," Journal of CRYPTOLOGY, pp. 3-72, 1991.

[9] R. I. Syarif, "Differential Cryptanalysis pada Full Round Design 32 Bit Lightweight Block Cipher Algorithm (DLBCA)," Sekolah Tinggi Sandi Negara (STSN), Bogor, 2018.

[10] S. Hong, D. Hong, Y. Ko, D. Chang, W. Lee and S. Lee, "Differential Cryptanalysis of TEA and XTEA," Springer, pp. 402-417, 2003.