 ISSN 2089-5490

RANCANGAN MODEL ALGORITMA HYBRID TEKNIK ENKRIPSI XOR DENGAN KOMBINASI MODE BLOCK *CIPHER* CBC - ECB 512-BITS DAN ALGORITMA RSA

Agung Purnomo Sidik[1], Nova Mayasari[2]
Fakultas Sains & Teknologi,
Universitas Pembangunan Panca Budi
Medan, Sumatera Utara
[1]agung@dosen.pancabudi.ac.id, [2]maya7886@pancabudi.ac.id

***Abstrak*—Penelitian ini dilakukan dengan menganalisis kombinasi dari algoritma RSA mode operasi blok cipher ECB dan CBC dengan panjang blok 512-bits. Penelitian ini bertujuan untuk mendapatkan sebuah algoritma kripografi yang cepat, hemat sumber daya, dan terbebas dari masalah key distribution. Data yang digunakan dalam pengujian merupakan data teks dengan panjang pesan dari 10 karakter hingga 1 juta karakter. Hasil penelitian menunjukkan bahwa kombinasi dari algoritma RSA dengan mode operasi CBC-ECB 512-bits mampu memberikan keamanan yang optimal terhadap data. Dimana cipher text dan cipher key yang dihasilkan akan sangat aman dari serangan exhaustive search atau bruto force. Dibutuhkan waktu hingga 1,80 x 10257 Tahun untuk memcahkan cipher text, sedangkan untuk cipher key nya dibutuhkan waktu 3,17 x 10166 Tahun dengan teknik exhaustive search atau bruto force.**

***Kata Kunci*—RSA, Enkripsi, Cipher**

I. PENDAHULUAN

Kriptografi AKAN merahasiakan informasi dengan menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya. Saat ini banyak bermunculan algoritma kriptografi yang terus dianalisis, dicoba, dan disempurnakan untuk mencari algoritma yang dianggap memenuhi standar keamanan. Setiap algoritma memiliki tingkat keamanan yang berbeda-beda, begitu juga dengan tingkat komplektifitas yang berbeda-beda pula. Algoritma dengan tingkat keamanan yang tinggi dan dengan komplektifitas yang rendah akan sangat baik untuk diterapkan, hal ini dikarenakan proses enkripsi dan deskripsi menjadi jauh lebih cepat dan memakan sumber daya yang rendah. Tetapi, setiap algoritma kriptografi yang telah ditemukan memiliki berbagai kelemahan sendiri.

Salah satu algoritma kriptografi yang paling cepat adalah algoritma kriptografi dengan teknik XOR. Walau sangat sederhana, tetapi teknik XOR menjadi pilihan untuk kebutuhan proses enkripsi yang cepat. Teknik XOR akan menjadi sangat aman jika ditetapkan dalam skema proses yang tepat, jika skema prosesnya tidak tepat, maka teknik XOR akan sangat tidak aman [1].

Teknik XOR masuk ke dalam bagian algoritma kripotgrafi simetris, di mana kunci untuk proses enkripsi dan dekripsi hanya satu buah kunci yang sama. Dikarenakan dalam proses enkripsi dan dekripsinya teknik XOR melakukan proses korespodensi satu-satu, atau satu bit kunci akan di XOR-an terhadap satu bit *plain text*, maka proses enkripsi dan dekripsi pada teknik XOR diterapkan pada mode operasi block *cipher* untuk meningkatkan keamanan dari *cipher text* yang dihasilkan [2].

Dalam operasi block *cipher*, terdapat beberapa mode yang dapat digunakan, beberapa diantaranya dalah mode operasi ECB (Electronic Code Book) dan CBC (*Cipher* Block Chaining). ECB dan CBC merupakan bentuk operasi dari mode block *cipher* pada algoritma simetris. Dalam skema block *cipher*, *plain text* terlebih dahulu akan dibagi ke dalam blok-blok sepanjang n-bits untuk kemudian setiap blok akan dilakukan pengoperasian. Jenis operasi yang dapat digunakan untuk mode operasi ini adalah teknik operasi XOR. Kelebihan dari teknik XOR adalah tingkat kecepatan dan kesederhanaanya dalam prosesnya, namun tetap menjanjikan keamanan yang tinggi terhadap *cipher text* yang dihasilkan [3].

Kekuatan atau keamanan pada mode operasi block *cipher* terutama terletak pada panjang blok yang digunakan, semakin panjang blok yang digunakan maka *cipher text*yang dihasilkan semakin aman. Pada penelitian ini, panjang blok yang diajukan adalah sepanjang 512-bits. Dengan panjang 512-bits, maka algoritma ini memiliki kemungkinan kunci dan kemungkinan variasi *cipher text* sepanjang 2512 atau sekitar 1,34 x 10154 kombinasi kunci yang berbeda [4].

Penelitian ini mencoba untuk mengkombinasikan algoritma block *cipher* Electronic Code Book (ECB) dan *Cipher* Block Chaining (CBC) menjadi satu buah mode operasi yang utuh untuk menghasilkan super *cipher*text yang jauh lebih aman dan lebih kuat dari *cipher text* sebelumnya dengan didukung dengan dua buah kunci rahasia berbeda yang masing-masing memiliki panjang kunci sepanjang 512-bits. Dengan kombinasi ini, akan dihasilkan kemungkinan kombinasi kunci sepanjang 2^{1024} yang akan meningkatkan keamanan kunci tanpa harus mengorbankan panjang blok pemrosesan. Sama seperti halnya jenis algoritma kriptografi simetris yang lain, kekurangan utama dari teknik XOR adalah pada masalah distribution *key*, di

 ISSN 2089-5490

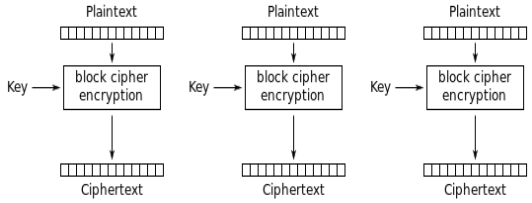
mana pengirim pesan akan sangat kesulitan dalam mendistribusikan kunci kepada penerima pesan dengan aman agar pesan tersebut dapat didekripsi kembali menjadi pesan asli (*plain text*). Jika kunci jatuh ketangan penyadap, maka proses enkripsi yang telah dilakukan akan sia-sia [2].

Untuk mengatasi masalah *key* distribution yang terjadi, maka digunakanlah sebuah algoritma asimetris untuk mengenkripsi kunci yang dihasilkan. Algoritma asimetris yang digunakan adalah algoritm RSA yang dirancang oleh Ron Rivest, Adi Shamir, dan Leonard Adleman, pada tahun 1970-an [5].

II. ECB (ELECTRONIC CODE BOOK)

ECB (*Electronic Code Book*) merupakan salah satu mode pemrosesan pada block *cipher*. Mode ECB cocok untuk mengenkripsi file yang diakses secara acak karena tiap blok *plain text* dienkripsi secara independen. Bahkan jika mode ECB dikerjakan dengan prosesor paralel, maka setiap prosesor dapat melakukan enkripsi atau dekripsi blok *plain text* yang berbeda-beda [3].

Pada mode operasi *Electronic Code Book* (ECB), enkripsi dilakukan dengan meng-XOR-kan masing-masing blok dari *plain text* dengan sebuah *key* (kunci) yang memiliki panjang bit yang sama dengan masing-masing blok, seperti yang ditunjukkan pada gambar berikut:

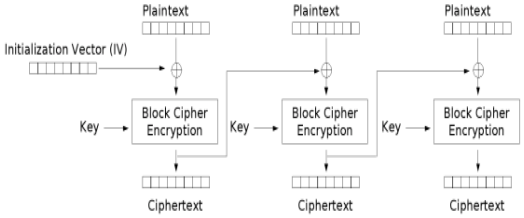


Gambar 1. Skema Proses Mode Operasi ECB

III. CBC (CIPHER BLOCK CHAINING)

Mode *Cipher Block Chaining* (CBC) menerapkan mekanisme umpan-balik (f*eedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya diumpan-balikkan ke dalam enkripsi blok yang sedang diproses. Caranya, blok *plain text* yang sedang diproses di-XOR-kan terlebih dahulu dengan blok *cipher text* hasil enkripsi sebelumnya, selanjutnya hasil peng-XOR-an ini masuk ke dalam fungsi enkripsi [3]..

Pada *Cipher Block Chaining* (CBC), enkripsi dilakukan dengan meng-XOR-kan blok pertama dari *plain text* dengan IV (*Initialization Vector*) yang terdiri dari bit 0 sepanjang n-bits, lalu setelah itu di-XOR-kan kembali dengan kunci untuk menghasilkan *cipher text* untuk blok pertama. *Cipher text* ini digunakan sebagai IV (*Initialization Vector*) untuk enkripsi blok selanjutnya, begitu seterusnya. Seperti yang ditunjukkan pada gambar berikut:



Gambar 2. Skema Proses Mode Operasi CBC

IV. RSA

RSA merupakan algoritma kriptografi asimetris memiliki dua kunci, yaitu kunci publik dan kunci privat yang dirancang oleh Ron Rivest, Adi Shamir, dan Leonard Adleman. Kekuatan dari RSA sangat bergantung rumitnya memfaktorkan bilangan bulat (integer) menjadi dua buah bilangan prima yang berbeda. RSA hanya menggunakan operasi perpangkatan untuk operasi enkripsi dan dekripsi. RSA termasuk algoritma asimetri [6].

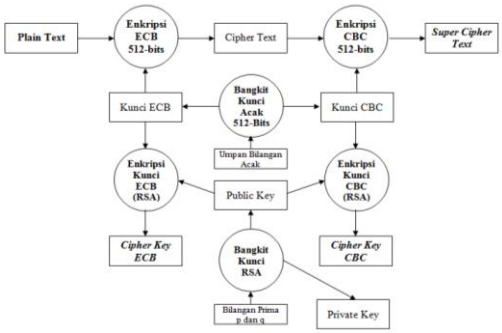
Algoritma RSA terdiri dari 3 proses, yaitu: [7]

1. Pembangkit Kunci
   1) Pilih dua bilangan prima acak, p dan q.
   2) Hitung modulus sistem
      $$n = p * q$$
   3) Cari Totient Φ (n)
      $$\Phi(n) = (p-1)(q-1)$$
   4) Pilih kunci enkripsi e secara acak
      Dimana $1 < e < \Phi(n)$, gcd $(e, \Phi(n)) = 1$
   5) Tentukan kunci dekripsi d dengan persamaan berikut:
      $$d \equiv e^{-1} \pmod{\Phi(n)}$$
      dimana persamaan di atas ekivalen dengan :
      $$e * d \equiv 1 \pmod{\Phi(n)}, \text{ dimana } 0 \leq d \leq n$$
      Hasil pembangkitan kunci:
      a. *Private key* = (d, n)
         bersifat sangat rahasia, dan hanya penerima pesan yang boleh mengetahuinya.
      b. *Public key* = (e, n)
         Bersifat tidak rahasia, dan boleh disebarkan dengan bebas.
2. Enkripsi
   Secara umum proses enkripsi dengan RSA dilakukan dengan rumus sebagai berikut :
   $$C_i = P_i^e \bmod n$$
3. Deskripsi
   Secara umum proses dekripsi dengan RSA dilakukan dengan rumus sebagai berikut:
   $$P_i = C_i^d \bmod n$$

 ISSN 2089-5490

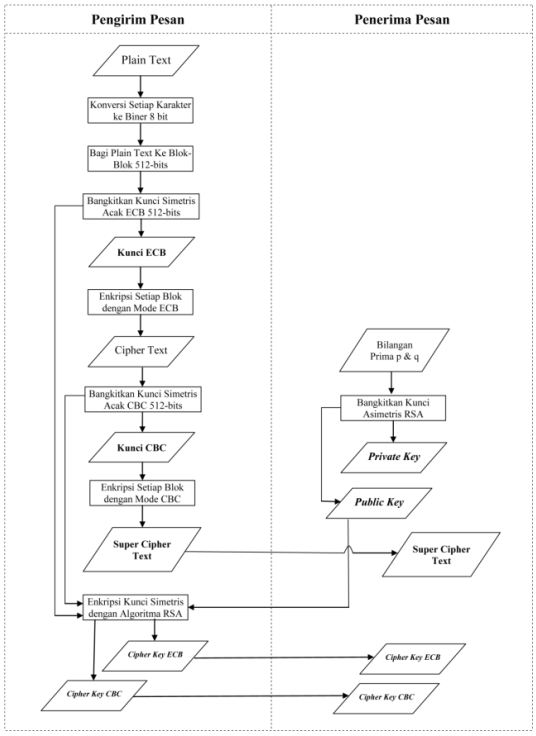
V. METODE PENELITIAN

Skema enkripsi dari algoritma hybrid yang diusulkan dapat dilihat pada gambar berikut:



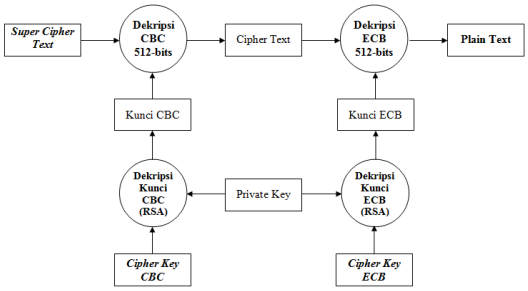
Gambar 3. Skema Enkripsi dari Algoritma Hybrid

Proses dekripsi pada algoritma *Hybrid* yang diusulkan dapat dilihat pada flowchart berikut:



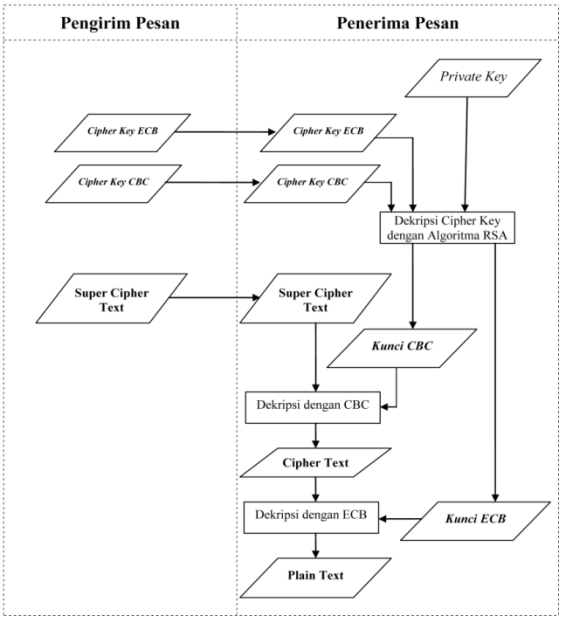
Gambar 4. Flowchart Proses Enkripsi dari Algoritma Hybrid

Skema dekripsi dari algoritma hybrid yang diusulkan dapat dilihat pada gambar berikut:



Gambar 5. Skema Dekripsi dari Algoritma Hybrid

Proses dekripsi pada algoritma Hybrid yang diusulkan dapat dilihat pada flowchart berikut:



Gambar 6. Flowchart Proses Dekripsi dari Algoritma Hybrid

Alur kerja proses pada skema algoritma *hybrid* yang diusulkan mengikuti alur berikut:

1. Tahap Pertama (Pembangkitan Kunci RSA)
2. Tahap Kedua (Enkripsi)
   a. Pembangkitan Kunci ECB
   b. Pembangkitan Kunci CBC
   c. Proses Enkripsi dengan Mode ECB
   d. Proses Enkripsi dengan Mode CBC
   e. Proses Enkripsi Kunci ECB dengan Algoritma RSA
   f. Proses Enkripsi Kunci CBC dengan Algoritma RSA
3. Tahap Ketiga (Dekripsi)
   a. Dekripsi *Cipher key* CBC
   b. Dekripsi *Cipher key* ECB
   c. Dekripsi *Super cipher text* dengan Mode Operasi CBC
   d. Dekripsi *Cipher text* dengan Mode Operasi ECB

VI. HASIL DAN DISKUSI

Pengujian dilakukan terhadap skema algoritma hybrid yang diusulkan. Pada pembahasan ini, terdapat sebuah *plain text* yang akan diuji, adapun plaintext tersebut adalah:

Plain text : Kota Medan Ibu Kota Provinsi Sumatera Utara, Kota Medan Merupakan Kota Terbesar di Luar Jawa di Indonesia.

Bangkitkan kunci RSA sehingga didapatkan sebuah *public key* yaitu (3149, 2159869) dan *private key* yaitu (1010249, 2159869). Lalu bangkitkan kunci simetris ECB dan CBC secara acak sepanjang 512-bits. Kunci acak ECB 512-bits yang digunakan sebagai berikut:

0001010101100101110001111011100110000101011010110001001111111010110110100110110001000100010101001110011000110110010101011010011101000001000111101111

 ISSN 2089-5490

01111111000110111110000010110011000101110101011000111000010011111100011000111001111100110010100110001100101101100110001010111100101001001110010010111111000111100010101001011110011100000001111010001111110001100110101001011010101101010001111101100101011010110011110011101000110111000100000000000010000001010011010001010111100100000001111000111001011000010000101101

Kunci acak CBC 512-bits yang digunakan sebagai berikut:

00111010000111010100000110111001000100000100001110001110111101100110110100001110100010110110010000100001011111101110011100111110101011110111100111110000110101110100101010001000110111001110110101101100011111110101100101000100110011011110100010010101111001110110011000000100111001101001110101101010000010101000100010111111011000000010000101010000001101011100101000101011000111101011101010111001000011011011010110000111000001101111001111011010111000011111001011101100010001010010110011010100001110001110011001001000101

Panjang *plain text* yang akan dienkripsi adalah 106 karakter atau 848 bit. Konversi *plain text* dalam bentuk biner lalu bagi ke dalam blok-blok dengan panjang 512-bits. Jika pada blok terakhir ternyata panjangnya kurang dari 512-bits, maka tambahkan bit padding ‘1’ hingga panjang blok terakhir menjadi 512-bits. Proses pembagian *plain text* ke dalam blok-blok 512-bits adalah sebagai berikut:

Plain Blok ke - 0

0100101101101111011101000110000100100000010011
0101100101011001000110000101101110001000000100100
1011000100111010100100000010010110110111101110100
01100001001000000010100000111001001101111011101100
11010010110111001110011011010010010000000101001101
1101010110110101100001011101000110010101110010011
00001001000000010101010111010001100001011100100110
00010010110000100000001001011011011110111010001100
00100100000001001101011001010110010001100001011011
10001000000010011010110010101110010011101010111000
001100001011010101101100001

Plain Blok ke - 1

0110111000100000010010110110111101110100011000
01001000000010101000110010101110010011000100110010
10111001101100001011100100010000000110010001101001
00100000001001100011101010110000101110010001000000
10010100110000101110111011000010010000000110010001
10100100100000001001001011011100110010001101111011
011100110010101110011011010010110000100101110**1111**
1111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111
1111111111111111111111111

Lakukan operasi XOR dengan mode ECB 512-bits seperti tabel berikut:

Tabel 1. Enkripsi Plain text Blok ke - 0 dengan Kunci ECB

***Plain text* Blok ke – 0**	**Kunci ECB**	**Hasil Enkripsi**
01001011011011110	00010101011001011	01011110000010101
11101000110000100	10001111011100110	01100111101100010
10000001001101011	00010101101011000	10010100100110011
00101011001000110	10011111110101101	10110100111101011
00010110111000100	10100110110001000	10110000001001100
00001001001011000	10001010100111001	10000011101100001
10011101010010000	10001101100101011	00010000110111011
00100101101101111	00101011001110100	00001110100011011
01110100011000010	00010001111011111	01100101100011101
01000000101000001	01111111000110111	00111111101110110
11001001101111011	11000001011001100	00001000110110111
10110011010010110	01011101010110001	11101110000100111
11100111001101101	11000010011111100	00100101010010001
00100100000010100	01100011100111110	01000111100101010
11011101010110110	01100101001100011	10111000011010101
10110000101110100	00101101100110001	10011101001000101
01100101011100100	01011110010100100	00111011001000000
11000010010000001	11100100101111110	00100110111111111
01010101110100011	00111100010101001	01101001100001010
00001011100100110	01111001110000001	01110010010100111
00010010110000100	11101000111111000	11111010001111100
00001001011011011	11001101010010110	11000100001001101
11011101000110000	10101101010001111	01110000010111111
10010000001001101	10110010101101011	00100010100100110
01100101011001000	00111100111010001	01011001100011001
11000010110111000	10111000100000000	01111010010111000
10000001001101011	00010000001010011	10010001000111000
00101011100100111	01000101011110010	01101110111010101
01010111000001100	00000111100011100	01010000100010000
00101101011011000	10110000100001011	10011101111010011
01	01	00

Tabel 2. Enkripsi Plain text Blok ke - 1 dengan Kunci ECB

***Plain text* Blok ke – 1**	**Kunci ECB**	**Hasil Enkripsi**
01101110001000000	00010101011001011	01111011010001011
10010110110111101	10001111011100110	00011001101011011
11010001100001001	00010101101011000	11000100001010001
00000010101000110	10011111110101101	10011101011101011
01010111001001100	10100110110001000	11110001111000100
01001100101011100	10001010100111001	11000110001100101
11011000010111001	10001101100101011	01010101110010010
00010000001100100	00101011001110100	00111011000010000
01101001001000000	00010001111011111	01111000110011111
10011000111010101	01111111000110111	11100111111100010
10000101110010001	11000001011001100	01000100101011101
00000010010100110	01011101010110001	01011111000010111
00010111011101100	11000010011111100	11010101000010000
00100100000011001	01100011100111110	01000111100100111
00011010010010000	01100101001100011	01111111011110011
00100100101101110	00101101100110001	00001001001011111
01100100011011110	01011110010100100	00111010001111010
11011100110010101	11100100101111110	00111000011101011
11001101101001011	00111100010101001	11110001111100010
00001001011101111	01111001110000001	01110000101101110
11111111111111111	11101000111111000	00010111000000111
11111111111111111	11001101010010110	00110010101101001
11111111111111111	10101101010001111	01010010101110000
11111111111111111	10110010101101011	01001101010010100
11111111111111111	00111100111010001	11000011000101110
11111111111111111	10111000100000000	01000111011111111
11111111111111111	00010000001010011	11101111110101100
11111111111111111	01000101011110010	10111010100001101
11111111111111111	00000111100011100	11111000011100011
11111111111111111	10110000100001011	01001111011110100
11	01	10

Lalu *cipher text* dienkripsi kembali dengan melakukan operasi XOR dengan mode CBC 512-bits menggunakan kunci CBC seperti yang dapat dilihat pada tabel berikut:

Tabel 3. Enkripsi Cipher text Blok ke - 0 dengan Kunci CBC

***Cipher text* Blok ke - 0**	**IV (*Initial vector*)**	**Hasil XOR Blok ke - 0 dengan IV**	**Kunci CBC**	***Super Cipher Text***
0101111000	0000000000	0101111000	0011101000	0110010000
0010101011	0000000000	0010101011	0111010100	0101111111
0011110110	0000000000	0011110110	0001101110	0010011000
0010100101	0000000000	0010100101	0100010000	0110110101
0010011001	0000000000	0010011001	0100001110	0110010111
1101101001	0000000000	1101101001	0011101111	1110000110
1110101110	0000000000	1110101110	0110011011	1000110101
1100000010	0000000000	1100000010	0100001110	1000001100
0110010000	0000000000	0110010000	1000101101	1110111101

 ISSN 2089-5490

Cipher text Blok ke - 0	IV (*Initial vector*)	Hasil XOR Blok ke - 0 dengan IV	Kunci CBC	*Super Cipher Text*
0111011000	0000000000	0111011000	1001000010	1110011010
0100010000	0000000000	0100010000	0001011111	0101001111
1101110110	0000000000	1101110110	1011100111	0110010001
0001110100	0000000000	0001110100	0011111010	0010001110
0110110110	0000000000	0110110110	1011110111	1101000001
0101100011	0000000000	0101100011	1001111100	1100011111
1010011111	0000000000	1010011111	0011010111	1001001000
1101110110	0000000000	1101110110	0100101010	1001011100
0000100011	0000000000	0000100011	0010001101	0010101110
0110111111	0000000000	0110111111	1100111011	1010000100
0111000010	0000000000	0111000010	0101101100	0010101110
0111001001	0000000000	0111001001	0111111010	0000110011
0101001000	0000000000	0101001000	1100101000	1001100000
1010001111	0000000000	1010001111	1001100110	0011101001
0010101010	0000000000	0010101010	1111010001	1101111011
1110000110	0000000000	1110000110	0010101111	1100101001
1010110011	0000000000	1010110011	0011101100	1001011111
1010010001	0000000000	1010010001	1100000010	0110010011
0100111011	0000000000	0100111011	0111001101	0011110110
0010000000	0000000000	0010000000	0011101011	0001101011
0100110111	0000000000	0100110111	0101000001	0001110110
1111110110	0000000000	1111110110	0101000100	1010110010
1001100001	0000000000	1001100001	0101111101	1100011100
0100111001	0000000000	0100111001	1000000100	1100111101
0010100111	0000000000	0010100111	0010101000	0000001111
1111101000	0000000000	1111101000	0001101011	1110000011
1111100110	0000000000	1111100110	1001010001	0110110111
0010000100	0000000000	0010000100	0101100011	0111100111
1101011100	0000000000	1101011100	1101011101	0000000001
0001011111	0000000000	0001011111	0101110010	0100101101
1001000101	0000000000	1001000101	0001101101	1000101000
0010011001	0000000000	0010011001	1010110000	1000101001
0110011000	0000000000	0110011000	1110000011	1000011011
1100101111	0000000000	1100101111	0111100111	1011001000
0100101110	0000000000	0100101110	1011010111	1111111001
0010010001	0000000000	0010010001	0000111110	0010101111
0001110000	0000000000	0001110000	0101110110	0100000110
1101110111	0000000000	1101110111	0010001010	1111111101
0101010101	0000000000	0101010101	0101100110	0000110011
0000100010	0000000000	0000100010	1010000111	1010100101
0001001110	0000000000	0001001110	0001110011	0000111101
1111010011	0000000000	1111010011	0010010001	1101000010
00	00	00	01	01

Pada proses enkripsi untuk *cipher text* blok ke - 1, IV (*Initial vector*) lama akan digantikan dengan hasil akhir dari proses enkripsi pada blok ke – 0. Begitu seterusnya, hasil enkripsi dari blok sebelumnya akan dijadikan sebagai IV untuk blok selanjutnya. Hal ini terus berulang sampai akhir dari blok

Tabel 4. Enkripsi Cipher text Blok ke - 1 dengan Kunci CBC

Cipher text Blok ke - 1	IV (*Initial vector*)	Hasil XOR Blok ke - 1dengan IV	Kunci CBC	*Super Cipher Text*
0111101101	0110010000	0001111101	0011101000	0010010101
0001011000	0101111111	0100100111	0111010100	0011110011
1100110101	0010011000	1110101101	0001101110	1111000011
1011110001	0110110101	1101000100	0100010000	1001010100
0000101000	0110010111	0110111111	0100001110	0010110001
1100111010	1110000110	0010111100	0011101111	0001010011
1110101111	1000110101	0110011010	0110011011	0000000001
1100011110	1000001100	0100010010	0100001110	0000011100
0010011000	1110111101	1100100101	1000101101	0100001000
1100011001	1110011010	0010000011	1001000010	1011000001
0101010101	0101001111	0000011010	0001011111	0001000101
1100100100	0110010001	1010110101	1011100111	0001010010
0111011000	0010001110	0101010110	0011111010	0110101100
0100000111	1101000001	1001000110	1011110111	0010110001
1000110011	1100011111	0100101100	1001111100	1101010000
1111110011	1001001000	0110111011	0011010111	0101101100
1111100010	1001011100	0110111110	0100101010	0010010100
0100010010	0010101110	0110111100	0010001101	0100110001
1011101010	1010000100	0001101110	1100111011	1101010101
1111100001	0010101110	1101001111	0101101100	1000100011
0111110101	0000110011	0111000110	0111111010	0000111100
0100001000	1001100000	1101101000	1100101000	0001000000
0010001111	0011101001	0001100110	1001100110	1000000000
0010011101	1101111011	1111100110	1111010001	0000110111
1111110111	1100101001	0011011110	0010101111	0001110001
1001100001	1001011111	0000111110	0011101100	0011010010
0010010111	0110010011	0100000100	1100000010	1000000110
1100111010	0011110110	1111001100	0111001101	1000000001
0011110100	0001101011	0010011111	0011101011	0001110100
0111000011	0001110110	0110110101	0101000001	0011110100
1010111111	1010110010	0000001101	0101000100	0101001001
0001111100	1100011100	1101100000	0101111101	1000011101
0100111000	1100111101	1000000101	1000000100	0000000001
0101101110	0000001111	0101100001	0010101000	0111001001
0001011100	1110000011	1111011111	0001101011	1110110100

Cipher text Blok ke - 1	IV (*Initial vector*)	Hasil XOR Blok ke - 1dengan IV	Kunci CBC	*Super Cipher Text*
0000111001	0110110111	0110001110	1001010001	1111011111
1001010110	0111100111	1110110001	0101100011	1011010010
1001010100	0000000001	1001010101	1101011101	0100001000
1010111000	0100101101	1110010101	0101110010	1011100111
0010011010	1000101000	1010110010	0001101101	1011011111
1001010011	1000101001	0001111010	1010110000	1011001010
0000110001	1000011011	1000101010	1110000011	0110101001
0111001000	1011001000	1100000000	0111100111	1011100111
1110111111	1111111001	0001000110	1011010111	1010010001
1111101111	0010101111	1101000000	0000111110	1101111110
1101011001	0100000110	1001011111	0101110110	1100101001
0111010100	1111111101	1000101001	0010001010	1010100011
0011011111	0000110011	0011101100	0101100110	0110001010
1000011100	1010100101	0010111001	1010000111	1000111110
0110100111	0000111101	0110011010	0001110011	0111101001
1011110100	1101000010	0110110110	0010010001	0100100111
10	01	11	01	10

Kemudian gabungkan seluruh hasil enkripsi yang didapat untuk masing-masing blok dan konversikan ke dalam bentuk heksadesimal. Hasil akhir ini disebut sebagai *super cipher text*. Seperti yang ditunjukkan pada tabel berikut:

Tabel 5. Konversi Super cipher text ke Bentuk Heksadesimal

***Super Cipher Text* dalam Bentuk Biner**	***Super Cipher Text* dalam Bentuk Heksadesimal**
011001000001011111110010011000011011010101100101111110000110100011010110000011001110111011110011010010100111101100100010010001110110100000111000111111001001000100101110000101011101010000100001010111000001100111001100000001110100111011110111100101001100101111101100100110011110110000110101100011101101010110010110001110011001111010000000111111100000110110110111011110011100000000010100101101100010100010001010011000011011101100100011111110010010101111010000011011111111010000110011101010010100001111011101000010010010010101001111001111110000111001010100001011000100010100110000000001000000111000100001000101100000100010001010001010010011010110000101100011101010000010110110000100101000100110001110101010101100010001100001111000001000000010000000000000110111000111000100110100101000000110100000000010001110100001111010001010010011000011101000000000010111001001111011010011110111111011010010010000100010111001111011011111101100101001101010011011100111101001000111011111110110010100110101000110110001010100011111100111101001010010011110	6417f261b565f868d60cef79a53d9123b41c7e48970aea10ae0ce603a77bca65f64cf61ac76acb1ccf40fe0db779c014b6288a61bb23f92bd06ff433a943dd09254f3f0e542c4530041c422c1114526b0b1d416c25131d56230f040800371c4d281a011d0f45261d005c9ed3dfb4908b9edfb29a9b9e91dfb29a8d8a8f9e949e

Konversi kunci ECB dan CBC ke dalam bentuk heksadesimal sehingga menjadi:

Kunci ECB :

1565c7b9856b13fada6c4454e63656567411efbf8df0598bab1c27e31cf994c65b315e52725f8f152f381e8fc66a5ab51f656b3ce8dc40040a68af2078e5842d

Kunci CBC:

3a1d41b910438ef66d0e8b64217ee73eaf79f0d74a88dced6c7eb2899bd12bcecc09cd3ad415117d810a81ae5158f5d5c86dac38379ed70f97622966a1c73245

Lalu lakukan enkripsi untuk setiap karakter heksadesimal dari kunci ECB dengan menggunakan *public key Public key* (e,n) = (3149, 2159869) yang sebelumnya telah diterima dari pengirim pesan sehingga menjadi:

Cipher key ECB:

1.bcb0c.1e5123.bcb0c.a6001.204442.7142a.1839bc.949e9.bcb0c.1e5123.7142a.1.1cf8a9.514bc.52302.1283c9.52302.1e5123.a6001.4ca25.4ca25.bcb0c.4ca25.196a21.1e5123.1cf8a9.1e5123.bcb0c.1e5123.bcb0c.1e5123.204442.4ca25.1.1.196a21.514bc.7142a.514bc.949e9.1283c9.514bc.0.bcb0c.1839bc.949e9.7142a.52302.7142a.1.a6001.5a931.204442.196a21.1

 ISSN 2089-5490

cf8a9.1.a6001.514bc.1839bc.1839bc.4ca25.a6001.1e5123.bcb0c.7142a.1cf8a9.1.bcb0c.196a21.bcb0c.5a931.204442.5a931.bcb0c.514bc.949e9.514bc.1.bcb0c.5a931.514bc.1cf8a9.949e9.1.196a21.949e9.514bc.a6001.1e5123.1e5123.52302.bcb0c.52302.7142a.bcb0c.1.514bc.1e5123.bcb0c.1e5123.7142a.1cf8a9.a6001.196a21.949e9.1283c9.a6001.4ca25.0.0.4ca25.0.52302.1e5123.949e9.52302.514bc.5a931.0.204442.949e9.196a21.bcb0c.949e9.4ca25.5a931.1283c9

Cipher key CBC:

1cf8a9.52302.1.1283c9.4ca25.1.7142a.1839bc.1.0.4ca25.1cf8a9.949e9.196a21.514bc.1e5123.1e5123.1283c9.0.196a21.949e9.7142a.1e5123.4ca25.5a931.1.204442.196a21.196a21.204442.1cf8a9.196a21.52302.514bc.204442.1839bc.514bc.0.1283c9.204442.4ca25.52302.949e9.949e9.1283c9.a6001.196a21.1283c9.1e5123.a6001.204442.196a21.7142a.5a931.949e9.1839bc.1839bc.7142a.1283c9.1.5a931.7142a.a6001.196a21.a6001.a6001.0.1839bc.a6001.1283c9.1cf8a9.52302.1283c9.4ca25.1.bcb0c.1.1.204442.1283c9.949e9.1.0.52302.949e9.1.52302.196a21.bcb0c.1.bcb0c.949e9.514bc.bcb0c.1283c9.bcb0c.a6001.949e9.1e5123.1283c9.52302.a6001.1cf8a9.949e9.1cf8a9.204442.1839bc.196a21.1283c9.204442.0.514bc.1839bc.204442.1e5123.5a931.5a931.1839bc.1e5123.1e5123.52302.1.a6001.204442.1cf8a9.5a931.4ca25.bcb0c.

Untuk melakukan dekripsi terhadap *super cipher text*, maka terlebih dahulu dekripsi *cipher key* CBC dan *cipher key* ECB menjadi kunci ECB dan kunci CBC yang digunakan untuk mendekripsi *super cipher text* menjadi *plain text* dengan mode CBC dan ECB 512 bits.

Berikut adalah hasil pengujian algoritma yang dirancang untuk beberapa pesan dengan panjang yang berbeda.

Tabel 6. Hasil Pengujian Terhadap Beberapa Pesan Dengan Panjang Berbeda

No	Panjang Pesan (Karakter)	*Public key*	*Private key*	Waktu Enkripsi (detik)	Waktu Dekripsi (detik)	Panjang *Cipher* (Karakter)
1	10	777,2f6d5b	f5e07,2f6d5b	0.02078	0.01929	128
2	100	4a9,29adf3	99879,29adf3	0.03393	0.03667	256
3	1000	22d,44db39	1effa1,44db39	0.17556	0.24275	2048
4	10000	1039,512579	2a98e9,512579	1.82661	2.28857	20096
5	100000	3a1,7121e7	55e9a1,7121e7	26.8484	42.6393	200064
6	1000000	1f7,35a425	2a8147,35a425	230.764	354.982	1000064

VII. KONTRIBUSI PENELITIAN

Kontribusi yang diberikan pada penelitian ini terletak pada diberikannya alternatif algoritma kriptografi yang cepat, hemat sumber daya, dan terbebas dari masalah *key* distribution. Dengan algoritma yang diusulkan, seluruh kunci simetris yang digunakan tidak lagi menjadi rahasia, sehingga pengirim pesan tidak perlu khawatir dalam proses pengiriman kunci kepada penerima pesan. Selain itu, algoritma ini menggunakan proses yang sederhana sehingga memiliki waktu proses yang sangat cepat dan sumber daya komputer yang hemat.

VIII. ANALISIS KEAMANAN

Algoritma yang diusulkan menggunakan mode blok *cipher* dengan panjang blok 512-bits, sehingga kemungkinan kunci yang dapat diterjadi adalah sebanyak $2^{512} = 1{,}34 \times 10^{154}$.

Jika diasumsikan bahwa *intruder* memiliki teknologi komputer berkecepatan tinggi yang mampu mencoba 1 juta triliun kunci perdetik (10^{18} operasi perdetik), maka waktu yang dibutuhkan untuk memecahkan kunci simetris yang dibutuhkan pada mode operasi blok *cipher* dengan panjang blok 512-bits dapat dilihat pada perhitungan berikut:

$$time = \frac{2^{512}}{(10^{18})(3600)(24)(365)(1000)} = 4{,}25\ x\ 10^{128}\ Tahun$$

Sehingga berdasarkan perhitungan di atas, *intruder* membutuhkan waktu 4,25 x 10^{128} ribu tahun untuk mencoba seluruh kunci yang ada untuk menemukan kunci yang tepat.

Mengingat pada algoritma digunakan dua buah mode operasi dengan panjang blok yang sama, maka waktu yang dibutuhkan *intruder* adalah

Time = $(4{,}25 \times 10^{128})^2 = 1{,}80 \times 10^{257}$ Tahun

Seluruh kunci yang digunakan dilindungi dengan algoritma RSA, sehingga untuk dapat membuka perlindungan ini dibutuhkan *private key* yang cocok. Kesulitan dalam menembus keamanan algoritma RSA adalah sulitnya melakukan pemaktoran terhadap suatu bilangan bulat menjadi dua buah bilangan prima. Mengalikan dua buah bilangan prima adalah perkara yang sangat mudah, tetapi memfaktorkan suatu bilangan bulat menjadi dua buah bilangan prima tanpa mengetahui salah satu faktor primanya bukanlah perkara yang mudah. Hal ini sesuai dengan rumus pembangkitan kunci algoritma RSA. *Private key* dapat dibentuk dari dua buah bilangan prima yaitu p dan q yang sangat rahasia. Lalu bilangan n yang tidak rahasia dapat dicari dengan n= p.q.

Langkah yang paling masuk akal untuk mendapatkan *private key* dengan pasti adalah dengan mengetahui nilai p dan q dengan memfaktorkan n, tetapi itu bukan perkara yang mudah, apalagi jika nilai p adalah prima yang besar hingga mencapai 10 digit lebih misalnya.

Cara lain yang dapat dilakukan adalah dengan teknik *bruto force* terhadap kemungkinan *private key* yang ada, dalam algoritma RSA diketahui bahwa:

Public key = (e, n), dimana e dan n tidak rahasia.

Private key = (d, n). Dimana $0 \leq d \leq n$. n tidak rahasia tapi d sangat rahasia

Dengan melihat syarat diatas, diketahui bahwa nilai n tidak rahasia, tetapi nilai d sangat rahasia, sedangkan nilai d ada diantara 0 dan n. Oleh karena itu banyaknya kemungkinan *private key* yang dapat terjadi adalah sebanyak n. sebagai contoh, jika:

P = 1987 dan q = 2779

n = p.q

n = 5.521.873

 ISSN 2089-5490

Sehingga dengan asumsi *intruder* dapat melakukan 1 juta operasi perdetik, maka asumsi waktu terburuk yang dimiliki *intruder* untuk dapat mendekripsikan *cipher key* hanya 6 detik, tetapi *intruder* akan mendapatkan 5.521.873 kunci yang berbeda-beda sehingga akan membingungkan *intruder* untuk menentukan kunci mana yang benar. Dikarenakan algoritma yang diusulkan menggunakan 2 buah mode operasi blok *cipher* (dua buah kunci) maka dihasilkan (5.521.873)2 atau 30.491.081.428.129 kombinasi kunci yang berbeda-beda.

Dengan analisa tersebut, maka semakin panjang bilangan prima yang digunakan, maka akan semakin aman kunci simetris yang digunakan untuk proses enkripsi dan dekripsi. Hal ini dikarenakan semain banyak kemungkinan *private key* yang dapat terbentuk. Sebagai contoh jika p terdiri dari 7 digit, dan q terdiri dari 7 digit, maka nilai n yang terbentuk bisa mencapai 14 digit, dengan asumsi bahwa *intruder* dapat melakukan 1 juta operasi perdetik, maka dibutuhkan setidaknya tiga tahun untuk mencoba seluruh kombinasi dari *private key* yang mungkin. Ini waktu yang cukup lama untuk menjamin keamanan dari kunci. Hal ini ditambah bahwa, *intruder* akan memiliki kemungkinan kunci sebanyak 14 digit angka atau sebanyak puluhan triliun kunci yang mungkin untuk setiap kunci. Dikarenakan algoritma yang dirancang menggunakan dua mode blok *cipher* dan dua kunci, maka kemungkinan kunci yang dibentuk dari kombinasi *private key* yang mungkin adalah sebanyak $(14)^2$ atau 192 digit atau sebanyak 10^{192} kunci yang mungkin. Maka seluruh kunci ini harus dicoba kembali untuk menemukan mana kunci yang tepat digunakan untuk medekripsikan *cipher text*. Jika diasumsikan *intruder* mampu mencoba 1 juta triliun kunci perdetik (10^{18} operasi perdetik), maka waktu yang dibutuhkan untuk memecahkan *cipher key* dapat dilihat pada perhitungan berikut:

$$time = \frac{10^{192}}{(10^{18})(3600)(24)(365)}$$
$$= 3{,}17\ x\ 10^{166}\ Tahun$$

Sehingga untuk memecahkan *cipher key* yang dihasilkan jika bilangan prima yang digunakan dalam proses pembangkitan kunci dengan RSA menggunakan bilangan p dan q dengan panjang minimal 7 digit, maka dibutuhkan waktu 3,17 x 10^{166} Tahun.

KESIMPULAN

1. Dengan mengkombinasikan mode operasi blok *cipher* ECB dan CBC sepanjang 512-bits dengan algoritma RSA, akan dihasilkan algoritma kriptografi yang cepat, ringan, dan aman. Dimana *cipher text* dan *cipher key* yang dihasilkan sangat tidak memungkinkan untuk dipecahkan secara exhaustive search.
2. *Cipher text* dan *cipher key* yang dihasilkan dari algoritma yang diusulkan sangat aman dari serangan *exhaustive search* atau *bruto force*. Dimana waktu yang dibutuhkan untuk memecahkan *cipher text* yang dihasilkan dibutuhkan waktu 1,80 x 10^{257} Tahun, sedangkan untuk *cipher key* nya dibutuhkan waktu 3,17 x 10^{166} Tahun

REFERENSI

[1] Wairya, S., Kumar. R., Nagaria., & Tiwari, S. (2012). *Comparative Performance Analysis of XORXNOR Function Based High-Speed CMOS Full Adder Circuits For Low Voltage VLSI Design*. International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2, April 2012

[2] Kumar, S., Suneetha, C. H., & Chandrasekhar, A. (2011). *A Block Cipher Using Rotation and Logical XOR Operations*. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011 ISSN (Online): 1694-0814

[3] Dashti, A., Kheradmand, H. A., & Jazi, M. D. (2016). *Comparison Of Three Modes Of Cryptography Operation For Providing Security and Privacy Based on Important Factors*. International Journal of Information Technology and Electrical Engineering Volume 5, Issue 3 ISSN: -2306-708 X June 2016

[4] Sridevi. (2014). *Construction of Stream Ciphers from Block Ciphers and their Security*. IJCSMC, Vol. 3, Issue. 9, September 2014, pg.703 – 714

[5] Munir, R. (2006). *Kriptografi*. Informatika: Bandung.

[6] Kallam, R. B. (2011. *An Enhanced RSA Public key Cryptographic Algorithm*. International Journal of Advanced Research in Computer Science (IJARCS)

[7] Singh, S. (2013). *A Performance Analysis of DES and RSA Cryptography*. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)