# Perancangan dan Implementasi Algoritma Kriptografi Kunci Simetri "Alay-Yielded Octal"

R. Purwoko Cahyo Nugroho – 13510014[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13510014@std.stei.itb.ac.id*

***Abstract*—Makalah ini menjabarkan sebuah algoritma kriptografi kunci simetri "Alay-Yielded Octal" yang terinspirasi dari fenomena alay di Indonesia dan menggunakan prinsip-prinsip bilangan berbasis biner dan oktal, serta menggunakan dasar tombol-tombol pada papan tombol sebuah telepon selular. Algoritma AYO terbukti berhasil memenuhi kebutuhan algoritma kriptografi sesuai prinsip yang dikemukakan oleh Shannon, confusion dan diffusion.**

***Index Terms*—Alay, papan tombol telepon selular, bilangan berbasis oktal, konversi biner – oktal – keypad.**

## I. PENDAHULUAN

Sejak komputer ditemukan, terutama sejak pengiriman data secara rahasia melalui jaringan komputer menjadi penting, berbagai algoritma kriptografi seolah berlomba-lomba untuk umat manusia temukan. Salah satu yang pernah menjadi tren adalah algoritma kriptografi kunci simetri. Namun saat ini, algoritma bertipe kunci simetri tersebut sudah mulai ditinggalkan orang karena sulitnya bertukar kunci secara aman, pengaturan kunci yang menyulitkan (satu kunci untuk satu pasangan pengguna) dan kunci yang terlalu sering / lama digunakan akan meningkatkan kerentanan. Di samping itu, belakangan ini juga telah ditemukan algoritma kriptografi kunci asimetri (juga dikenal sebagai kunci publik karena satu bagian kunci dapat bebas dipublikasikan) yang lebih memudahkan pengguna dalam pertukaran kunci (satu bagian bebas dipublikasikan), penyimpanan kunci (satu pengguna untuk satu pasangan kunci), dan amat sukar untuk melacak bagian lain jika hanya memiliki bagian publiknya saja.

Di lain pihak, pengembangan algoritma kriptografi kunci simetri juga penting untuk tidak dilupakan, sebagai sebuah dasar (nenek moyang) pengembangan algoritma kriptografi, karena bisa jadi tipe / jenis algoritma lain yang mungkin lebih kuat lagi akan dibidani oleh tipe algoritma kunci ini, atau bahkan mungkin pengembangan kriptografi kunci asimetri akan berdasarkan algoritma kunci simetri. Sementara algoritma kunci publik terlihat lebih aman karena enkripsi – dekripsinya menggunakan dua bagian kunci yang berbeda, algoritma kriptografi kunci privat sudah terbukti selama Perang Dunia II digunakan untuk mengirimkan pesan rahasia di kalangan Nazi Jerman (Enigma Cipher).

Fenomena *encoding* sms secara "4L@y" – yang bermula dari kerusakan beberapa tombol papan ketik dan memaksa orang-orang untuk melakukan substitusi huruf tertentu atau disebabkan juga oleh limit sms 160 karakter sehingga membuat beberapa orang melakukan penyingkatan kata secara tidak lazim, dan bermuara menjadi tren perusakan bahasa yang sempat mewabah (disukai sekaligus dibenci) dalam masyarakat Indonesia – mengindikasikan adanya mekanisme teks kriptik yang belum dan menarik untuk dijamah.

## II. DASAR TEORI

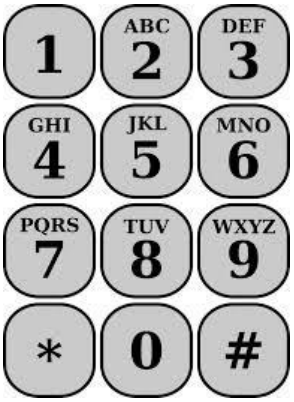
Kriptografi secara etimologis berasal dari bahasa Yunani, kriptos berarti tersembunyi, rahasia, dan graphein berarti tulisan. Secara terminologis, kriptografi adalah teknik mengacaukan pesan dan membuatnya tidak bisa dipahami oleh pihak ketiga. Metode kriptografi ada dua berdasarkan publisitas kunci, kunci publik dan kunci privat. Kriptografi kunci publik berarti sebagian kunci aman untuk dipublikasikan dan menjadi cara untuk membuat pesan teracak, sedangkan kriptografi kunci privat berarti seluruh bagian kunci harus tersembunyi dari jangkauan pihak luar karena cara untuk mengacak dan mengembalikan pesan hanya melalui kunci tersebut.

Bilangan biner adalah sebutan untuk representasi bilangan sebagai rangkaian bit (benar – salah atau 0 – 1). Dalam konteks informatika, bilangan biner bisa merepresentasikan apapun, mulai bilangan, huruf-huruf, hingga data transmisi siaran televisi favorit. Setiap tiga bit biner dapat ditranslasi menjadi sebuah bilangan oktal, bilangan yang memiliki rentang 0 – 7.

Sesuai dengan metode enkodingnya, karakter berupa huruf-huruf, angka, tanda baca, dan sebagainya masing-masing memiliki representasi biner yang berbeda. Dalam ASCII misalnya, karakter "A" dapat ditranslasi menjadi angka 01000001 dalam biner atau 65 dalam desimal. Setiap karakter memiliki representasi masing-masing

sehingga perubahan hanya satu bit akan menyebabkan perubahan karakter tersebut.

Papan tombol pada telepon umumnya memiliki delapan tombol yang juga bertuliskan huruf, dan empat lainnya bertuliskan simbol, seperti pada gambar 1. Prinsip penggunaan tombol telepon inilah yang diaplikasikan dalam pembuatan algoritma kriptografi yang berbasis oktal ini.




*Gambar 1 Keypad telepon*

## III. PERANCANGAN

Algoritma kriptografi yang baru dirancang akan memiliki hubungan erat dengan mekanisme papan tombol telepon dan prinsip oktal karena pada dasarnya ada delapan tombol bertuliskan huruf pada papan tombol. Algoritma ini mendapatkan inspirasi dari maraknya penggunaan enkoding alay dalam bahasa Indonesia, yang pertama-tama berkembang dari para pengguna telepon genggam. Algoritma ini bernama "Alay-Yielded Octal", atau bisa disingkat AYO, karena pembuatannya terinspirasi oleh bahasa alay yang sempat marak antara 2006 – 2012, dan dengannya diimplementasikan sebuah algoritma yang menggunakan prinsip papan tombol ponsel dan bilangan berbasis oktal.

Algoritma ini merupakan algoritma kunci privat, dengan kunci untuk melakukan enciphering sama dengan kunci untuk melakukan decipher. Tingkat keamanan algoritma kriptografi kunci privat tidak setinggi algoritma kriptografi kunci publik karena metode untuk melakukan ciphering dan enciphering menggunakan satu kunci yang sama, sehingga memiliki kunci dari sisi penerima atau pengirim pesan sudah cukup untuk menlakukan pembobolan seluruh pesan yang ada.

Bagian makalah ini akan memberi penjelasan mengenai perancangan algoritma baru yang berdasar pada konversi biner – oktal, pembuatan "keypad", dan konversi oktal – "keypad".

### *A. Konversi Biner - Oktal*

Pada hakikatnya, konversi ini merupakan translasi senarai biner dari karakter-karakter dalam pesan sesuai metode enkoding menjadi senarai bilangan oktal.

Konversi representasi bilangan ini diperlukan untuk memastikan temporal ciphertext aman untk diinputkan ke dalam keypad nantinya.

### *B. Pembuatan "Keypad"*

Keypad di sini dibuat menyerupai keypad yang ada pada telepon, dengan posisi yang lebih kurang sama secara abstrak. Huruf-huruf yang ada di dalam keypad akan disubstitusi dengan beberapa huruf yang didapatkan dari pembangkitan bilangan acak pada kunci masukan, dan juga akan dimasukkan dari senarai pelengkap (komplemen) yang berisi huruf-huruf lainnya. Keypad yang sudah dibuat ini akan kemudian digunakan dalam proses selanjutnya.

### *C. Konversi Oktal – "Keypad"*

Setelah semua teks dasar diubah menjadi senarai oktal dan keypad sudah terbentuk, setiap bilangan oktal akan dipetakan ke dalam keypad sesuai posisinya. Jika dianalogikan dengan keypad telepon, angka 0 – 7 pada oktal berkorespondensi satu-satu dengan angka 2 – 9 pada keypad, sehingga setiap bilangan oktal yang masuk ke dalam keypad akan dipetakan ke masing-masing huruf dalam tombol keypad tertentu secara acak.

### *D. Putaran*

Hal penting lain dari algoritma kriptografi ini adalah putaran. Dengan semakin besarnya ukuran cipherteks temporal, semakin besar pula waktu yang dibutuhkan untuk melakukan enciphering, dan di sini dilakukan beberapa kali putran terhadap cipherteks temporal (cipherteks temporal akan dienkripsi lagi, menghasilkan cipherteks temporal II, dan lagi, menghasilkan cipherteks temporal III, dst.) hingga banyaknya putaran yang diinginkan. Jadi, cipherteks yang telah dibuat akan menjadi jauh lebih besar daripada plainteks yang masuk.

Algoritma yang dibuat akan memenuhi criteria confusion dari Shannon, karena karakter-karakter dari ciphertext tidak memiliki hubungan langsung terhadap laintex. Di samping itu, prinsip diffusion juga terpenuhi, karena setiap karakter yang masuk akan dipetakan menjadi tidak tepat x karakter ciphertext (setiap karakter plaintext dibah menjadi 8/3 ciphertext dalam setiap putarannya), sehingga perubahan satu atau lebih karakter ciphertext akan menyebabkan kekacauan plaintext di sekitar ciphertext yang diubah. Algoritma ini akan membuat ciphertext menjadi sangat panjang sehingga sulit menentukan bagian mana dari ciphertext yang merupakan bagian tertentu dari ciphertext.

Algoritma ini secara khusus bergantung kepada panjangnya ciphertext yang dibuat berkat banyaknya putaransehingga membuat ciphertext luar biasa panjang jika dibandingkan dengan plaintext. Panjang cihertext yang akan terjadi dalam setipa putarannya adalah sebagai

berikut:

Ciphertext.Length = Plaintext.Length * 8 / 3 .....(1)

Dan panjang ciphertext ini akan berkembang secara eksponensial seiring dengan banyaknya putaran yang dilakukan oleh mesin pembuat ciphertext.

## IV. IMPLEMENTASI

Versi pertama dari algoritma ini dibuat dalam lingkungan bahasa Java, menggunakan IDE NetBeans versi 7.2.1. versi algoritma ini terutama menggunakan algoritma pseudorandom number generator yang disediakan oleh *library* Java. Selain pembangkit bilangan acak, versi algoritma ini tidak bergantung kepada jenis atau bahasa tertentu. (catatan penyusun: semua potongan implementasi akan dimasukkan sebagai gambar supaya tidak kehilangan format dan lebih *readable* dibandingkan jika dimasukkan sebagai tulisan)

### *A. Konversi Biner - Oktal*

Pada awalnya, semua string yang masuk akan dilakukan pengecekan apakah panjang bitnya bisa dibagi tiga karena akan dikonversi menjadi oktal yang masing-masing bilangannya membutuhkan tiga bit biner. Jika belum, maka akan dilakukan padding dengan angka random hingga panjang bilangan biner habis dibagi tiga. Setelah itu, semua bilangan akan dikonversi menjadi bilangan oktal, sebagai persiapan aplikasi pengacakan keypad di dalamnya (gambar 2).

```
public ArrayList<Integer> stringToOctal(String plain, String key) {
    char[] tempChar = plain.toCharArray();
    String binary = new String();
    ArrayList<Integer> octal = new ArrayList<>();
    for (char c : tempChar) {
        String tempStr = Integer.toBinaryString((int) c);
        while (tempStr.length() < 8) {
            tempStr = "0" + tempStr;
        }
        binary += tempStr;
    }

    //DONE: byte messing function
    r = new Random(iv / key.length());
    for (int i = 0; i < key.length(); i++) {
        int switchPos = ((int) key.charAt(i)) % binary.length();
        int messPos;
        do {
            messPos = r.nextInt(binary.length());
        } while (messPos < switchPos + 8 && messPos > switchPos - 8);
        char[] mess = binary.toCharArray();
        for (int j = 0; j < 8; j++) {
            char messTemp = mess[(messPos + j) % mess.length];
            mess[(messPos + j) % mess.length] = mess[(switchPos + j) % mess.length];
            mess[(switchPos + j) % mess.length] = messTemp;
        }
    }
    while (binary.length() % 3 != 0) {
        binary += r.nextInt(2);
    }

    for (int i = 0; i < binary.length(); i += 3) {
        octal.add(Integer.parseInt(binary.substring(i, i + 3), 2));
    }
    return octal;
}
```

*Gambar 2 Implementasi pemetaan biner*

### *B. Pembuatan "Keypad"*

Pembuatan keypad didasarkan kepada algoritma pembangkit bilangan acak pseudorandom. Pemberian seed (bibit / umpan bilangan) dilakukan secara acak berdasarkan putaran yang sedang berlaku (gambar 3), sehingga masing-masing putaran akan memiliki rangkaian keypad yang berbeda satu sama lain. Setelah itu akan dilakukan pengambilan karakter-karakter dari dalam kunci secara random dan distinct (gambar 4). Posisi keypad diandaikan sesuai dengan posisi keypad telepon, dengan urutan huruf dimulai dari angka 2, berturut hingga 9, sehingga akan didapatkan sebuah string yang panjangnya 26 dengan isi karakter kunci atau (inclusive or) komplemen (sebuah senarai sepanjang 26 karakter yang berisi semua huruf dalam alfabet).

```
public String keyGenerator(String key, int putaran) {
    String tempKey = "";

    //DONE: key scrambler
    r = new Random(key.charAt((iv + putaran * 26) % key.length()) * iv);
    while (tempKey.length() < 26) {
        tempKey += key.charAt(r.nextInt(key.length()));
    }

    //DONE: unique
    for (int i = 0; i < tempKey.length(); i++) {
        if (tempKey.indexOf(tempKey.charAt(i)) < i) {
            tempKey = tempKey.substring(0, i) + tempKey.substring(i + 1);
            i--;
        }
    }

    return tempKey;
}
```

*Gambar 3 Key generator*

```
public String keypadCryption(String keyPad) {
    //DONE: keyPad messing function
    int length = (keyPad.length() / 26 < 1 ? keyPad.length() : 26);
    String complement = "ZVWMBXKJQSNTHDIUEOALRCGFYP".toLowerCase();
    char[] chars = new char[26];
    int charPos = 5;
    for (int i = 0; i < 26; i++) {
        char inputKey;
        if (i < length) {
            inputKey = keyPad.charAt(i);
        } else {
            complement = complement.trim();
            inputKey = complement.charAt(0);
        }
        if (i == 25) {
            chars[i] = inputKey;
        } else {
            chars[charPos * 3 + (charPos > 5 ? 1 : 0) + (i / 8)] = inputKey;
            complement = complement.replace(inputKey, ' ');
            charPos = (charPos + 3) % 8;
        }
    }
    String tempKey = "";
    for (char c : chars) {
        tempKey += c;
    }
    return tempKey;
}
```

*Gambar 4 Key to Keypad Mapping*

### *C. Konversi Oktal – "Keypad"*

Setelah keypad selesai dibuat, senarai bilangan oktal yang masuk akan diinputkan kepada keypad "seperti mengetikkan sms di ponsel", hanya bedanya, setiap

pengetikan kali ini digunakan algoritma pembangkit bilangan acak (gambar 5) sehingga karakter dalam keypad yang muncul akan diacak (seperti menekan tombol angka 5 sekali, namun bisa mengeluarkan huruf ‘K’, atau ‘L’, tidak harus ‘J’).

```
public String octalToKeypadMapper(String keypad, ArrayList<Integer> octal) {
    //DONE: octal to keypad mapping
    String cipher = new String();
    for (int i : octal) {
        int rand = r.nextInt(i == 7 || i == 5 ? 4 : 3);
        cipher += keypad.charAt(i * 3 + rand + (i > 5 ? 1 : 0));
    }
    return cipher;
}
```

*Gambar 5 Konversi oktal - keypad*

Setiap plaintext yang telah masuk akan keluar sebagai sebuah ciphertext yang ukurannya lebih besar dari plaintext sebelumnya. Banyaknya putaran dalam algoritma ini mempengaruhi lama komputasi, karena semakin besar senarai yang harus diurus akan membuat konversi jadi memakan waktu semakin banyak.

### *D. Penyatuan*

Semua tahap persiapan pembuatan ciphertext di atas kemudian akan disatukan dengan cara sebagai berikut:

- Ditentukan banyaknya putaran yang akan dilakukan dalam enkripsi (dibangkitkan bilangan acak integer antara 1 – 5 [inclusive])
- Untuk setiap putaran, dilakukan:
  - pembangkitan sebuah kunci dengan sumber asal kunci yang telah masuk sepanjang maksimum 26 karakter (gambar 3)
  - pemetaan kunci kepada papan kunci (gambar 4)
  - translasi string pesan menjadi biner, kemudian menjadi senarai bilangan oktal (gambar 2)
  - pemetaan oktal – keypad (gambar 5).

Setelah proses enkripsi selesai, ada lagi satu hal yang penting: proses dekripsi. Proses dekripsi dari sebuah algoritma kriptografi kunci simetri sebenarnya hanyalah melakukan putaran balik terhadap proses enkripsinya, yang dalam kasus ini berarti melakukan pemetaan string ciphertext terhadap keypad untuk menghasilkan bilangan oktal, kemudian memetakannya kambali sebagai bit biner dan mengembalikan menjadi string plantext.

## V. HASIL IMPLEMENTASI

Setelah mengalami proses implementasi, program hasil kemudian dicobakan untuk melakukan enkripsi – dekripsi, dengan hasil sebagai berikut:

- Contoh plaintext kecil
  Plaintext: aku adalah anak gembala
  Kunci: nyemmm
  Ciphertext:
  odvawlvwjzdnnxrnovgcrlhbygaqpsvxyjtnjzanjvtctxn
  molankftzonrpjswbjoqqyfiyjsuaifcdyrvcxlmdygrcoln
  zojrqyshkjgtcilxjjxdpuiigoodcjsizjyqahmrdotbwfsnjo
  mvaqzhzooaaplnzjgnhlb

- Contoh plaintext sedang
  Plaintext: aku adalah anak gembala selalu riang serta gembira
  Kunci: nyemmm
  Ciphertext:
  ojnammvdjednnxffozgaamnyyorqfmvxyjtptzabjxtap
  xmnolankftzoyupjswbjadqyfqxjsuaifqjyivanlwsygrcg
  lvfodaqnxrqjadczxxqjgqauiigoakcjsiijyqahmrcosbwl
  mwdomepmzxpoaghbfvfjgnhgzngjjvamsndorgckivdj
  zrnviaajgvaxitcykgneipgjvgntihbofdnwstoyqrnazxxjr
  ahriniyzankfahyjucjlqkjouqmiqmyvepflpuytvnnmigo
  rrpglrjyeraasntjtaczinfyganuxihybgposrqoonnpxtvjtb
  cziqjogdhisigjftcdlrsodqnen

- Contoh plaintext besar
  Plaintext: aku adalah anak gembala selalu riang serta gembira karena aku senang bekerja tak pernah malas ataupun lengah
  Kunci: nyemmm
  Ciphertext:
  jgrhbxvwytdnjzpeozgnaihgybnaixtmygnqqfqnoxtncif
  qyzaagftzjyuhlzmtondassfnjzqpgmayjieczihljjnholal
  ybaqosixoqdaqsiajdqadffgyggwdsckondwtxtwokgqn
  zpdyfuwflmbongpgxvfyfdwysavygqqhxqqytgauitvyst
  noznqogtawlttjvbquxmroudnozqjjbaqhzvjjavqazpejy
  kaisxyykvnuixuooeclzfujqtapsqijunnnstaoivpvzmwy
  xupglrooseclxaqyjkhlixyygnaumxdjzvwllfsoqgacmv
  hjedhzinjjdgcsxipjitpdltxodahsznvjidqpmtrjedhuixno
  ogauitrjagclivwoebhuzitjfkpmlfroorpgfvsjigprzrnyyq
  pvxtqosuqjixqjtaneicljagqnzrtyrdatzmvjiepzxwzydtc
  zztljoqpjlntyjrprmfyyieabivwoquprliboynaasrvjfncfl
  xvonengzxgoorqzzttymgpmsvmjrrawiqxjzuqbxifymr
  pdsiyobtcjzqiyvbndlxfozdqgzmdoyacolnlyfdnosvqoy
  vclmxnjdaauxvlonkpsswbonbaisfqozgpbmiljvgawim
  vombnhsnsoxgqrxazytvnmzxjjlkqrsrcojdpexfnjbrnq
  mvgyzbntsfjyfuwdiiwjjqncsnlourhrlxxozdhrmmtykb
  nufafolvptzqm

Berdasarkan hasil implementasi dan percobaan enkripsi di atas, terlihat tidak ada kesamaan antara satu ciphertext dengan ciphertext yang lain pada permulaan stringnya, padahal semua awal string dan kunci sama. Ini menunjukkan bahwa antara plaintext dengan ciphertext tidak ada hubungan (*resemblance*) dan membuktikan terimplementasikannya prinsip Shannon pertama dalam pembuatan algoritma kriptografi : confusion.

Di samping itu, ukuran ciphertext yang berbeda dengan ukuran plaintext juga menunjukkan tdak adanya hubungan satu-satu antara plaintext dan ciphertext. Ini selain membuktikan lagi adanya prinsip confusion juga akan membuat para penyerang menjadi bingung akan apa maksud dari ciphertext.

## VI. PENGUJIAN KRIPTANALISIS

Di atas, hasil implementasi algoritma pada program telah dibuktikan, dan menghasilkan bukti bahwa prinsip confusion dari Shannon terpenuhi. Namun, prinsip kedua belum terbukti apakah algoritma yang baru dibuat ini merupakan implementasi dari prinsip diffusion. Untuk membuktikannya, diperlukan percobaan-percobaan serangan terhadap ciphertext yang dihasilkan oleh algoritma ini.

Untuk percobaan jenis serangan, akan digunakan pengujian terhadap serangan brute force attack, penyisipan blok cipher semu, perubahan blok cipher yang ada, dan penghapusan sebuah blok cipher.

### *A. Brute-Force Attack*

Pada serangan brute-force, seorang kriptanalis harus mampu menebak algoritma kriptografi yang digunakan, baru kemudian menebak kunci yang digunakan dalam proses enkripsi algoritma ini.

Dengan asumsi kriptanalis sudah bisa menebak bahwa algoritma enkripsi yang digunakan adalah AYO, kriptanalis selanjutnya harus menebak tiga hal:

- Jumlah putaran
- Pengacakan biner dalam string plaintext
- Papan kunci yang dibuat dalam setiap putaran.

Dengan jumlah karakter dalam papan kunci berjumlah 26, banyaknya karakter dalam sebuah system enkoding (dimisalkan ASCII) 255, dan putaran yang bervariasi antara 0 – 6 (exclusive), maka kriptanalis harus menemukan dalam setiap putaran sebuah senarai kunci ${}_{255}P_{26}$, atau sekitar 9.33e+61 urutan, atau jika ada satu kalkulasi setiap satu detik, kunci akan ditemukan maksimum setelah sekitar 2e+58 jam. Itu baru dihitung untuk putaran pertama. Jika algoritma enkripsi itu memiliki 5 putaran, maka akan ada sekitar 4.5e+62 pilihan urutan kunci, dan akan lebih lama dalam mencapai hasil yang diinginkan.

Hasil di atas baru merupakan hasil pencarian kunci menggunakan brute-force tanpa memperhitungkan adanya pengacakan biner dalam string plaintext, dan karena jumlah pengacakan biner adalah sejumlah panjang kunci yang diinputkan, maka banyaknya pengacakan biner bisa berapapun dan mempersulit pemecahan kriptanalisis dengan menggunakan brute – force attack.

### *B. penyisipan blok cipher*

Penyisipan blok cipher tidak bertujuan untuk mengetahui isi pesan, tetapi memodifikasinya. Hal ini dilakukan untuk membuat orang yang dimaksudkan untuk menerima pesan akan mempersepsikan hal yang berbeda dari yang dimaksud oleh pengirim pesan. Serangan ini termasuk dalam man-in-the-middle attack, karena penyerang tidak hanya menangkap pesan, tetapi juga mengubahnya.

Sebelum seorang kriptanalis melakukan serangan dengan menyisipkan sebuah blok cipher semu yang akan menyebabkan perubahan seluruh bagian dari ciphertext, hal yang harus diketahui oleh penyerang adalah panjang blok sehingga mengurangi kemungkinan adanya kerusakan ciphertext, tetapi benar bahwa ia mengisikan blok cipher yang baru. Pada bagian sebelumnya, dijelaskan bahwa panjang bit yang dikonversi adalah 3 bit menjadi oktal, dari 8 bit representasi biner dari karakter. Berarti bisa diasumsikan perubahan sepanjang 24 bit akan membuat tiga karakter berubah tanpa merusak seluruh ciphertext. Perubahan sepanjang 24 bit berarti penambahan 8 karakter baru dalam ciphertext.

Contoh (diambil contoh dari hasil implementasi, plaintext berukuran kecil):

Plaintext: aku adalah anak gembala
Kunci: nyemmm
Ciphertext:
odvawlvwjzdnnxrnovgcrlhbygaqpsvxyjtnjzanjvtctxn
molankftzonrpjswbjoqqyfiyjsuaifcdyrvcxlmdygrcoln
zojrqyshkjgtcilxjjxdpuiigoodcjsizjyqahmrdotbwfsnjo
mvaqzhzooaaplnzjgnhlb

Sekarang akan ditambahkan sebuah blok cipher baru pada posisi ke 24, dengan isi sama dengan 24 bit sebelumnya (duplikasi blok sebelumnya, sebagai pengecekan).

Ciphertext baru:
odvawlvwjzdnnxrnovgcrlhbodvawlvwjzdnnxrnovgcr
lhbygaqpsvxyjtnjzanjvtctxnmolankftzonrpjswbjoqqy
fiyjsuaifcdyrvcxlmdygrcolnzojrqyshkjgtcilxjjxdpuiig
oodcjsizjyqahmrdotbwfsnjomvaqzhzooaaplnzjgnhlb
Hasil dekripsi:
aku,-n¤,Œ-Œ- -Ì-dì-¬L-Œ

terjadi kerusakan pada plaintext, dan karena plaintext yang masih aman tidak mengalami perubahan adalah tiga karakter pertama, ketiga karakter pertama masih bisa dibaca, sedangkan karakter lain setelah pemasukan blok cipher baru menjadi rusak.

### *C. pengubahan blok cipher*

Sebagai usaha kedua untuk mengubah ciphertext yang dibuat oleh algoritma ini, akan dilakukan kriptanalisis dengan mengubah blok cipher yang memiliki posisi tertentu dalam ciphertext. Dalam kasus ini akan ditukar blok ciphertext pertama dan kedua.

Contoh (diambil contoh dari hasil implementasi, plaintext berukuran kecil):

Plaintext: aku adalah anak gembala
Kunci: nyemmm
Ciphertext:
odvawlvwjzdnnxrnovgcrlhbygaqpsvxyjtnjzanjvtctxn
molankftzonrpjswbjoqqyfiyjsuaifcdyrvcxlmdygrcoln
zojrqyshkjgtcilxjjxdpuiigoodcjsizjyqahmrdotbwfsnjo

mvaqzhzooaaplnzjgnhlb

Sekarang akan ditukarkan blok cipher baru pada posisi ke 24, dengan 24 bit sebelumnya (pertukaran blok 0 dan blok 1).

Ciphertext baru:
ygaqpsvxyjtnjzanjvtctxnmodvawlvwjzdnnxrnovgcrlh
bolankftzonrpjswbjoqqyfiyjsuaifcdyrvcxlmdygrcolnz
ojrqyshkjgtcilxjjxdpuiigoodcjsizjyqahmrdotbwfsnjo
mvaqzhzooaaplnzjgnhlb
Hasil dekripsi:
# -n¥lah anak gembala

Ternyata pertukaran blok menyebabkan kerusakan pada 7 karakter, sedangkan seharusnya terjadi kerusakan hanya pada karakter 0 sampai dengan 6.

Selanjutnya akan dilakukan pengujian dengan menghapus sebuah blok cipher.

### *D. penghapusan blok cipher*

Sebagai usaha terakhir untuk mengubah ciphertext yang dibuat oleh algoritma ini, akan dilakukan kriptanalisis dengan menghapus sebuha blok cipher yang memiliki posisi tertentu dalam ciphertext. Dalam kasus ini akan dihapus ciphertext blok pertama..

Contoh (diambil contoh dari hasil implementasi, plaintext berukuran kecil):

Plaintext: aku adalah anak gembala
Kunci: nyemmm
Ciphertext:
odvawlvwjzdnnxrnovgcrlhbygaqpsvxyjtnjzanjvtctxn
molankftzonrpjswbjoqqyfiyjsuaifcdyrvcxlmdygrcoln
zojrqyshkjgtcilxjjxdpuiigoodcjsizjyqahmrdotbwfsnjo
mvaqzhzooaaplnzjgnhlb

Kemudian blok pertama dihapus.

Ciphertext baru:
ygaqpsvxyjtnjzanjvtctxnmolankftzonrpjswbjoqqyfiyj
suaifcdyrvcxlmdygrcolnzojrqyshkjgtcilxjjxdpuiigoo
dcjsizjyqahmrdotbwfsnjomvaqzhzooaaplnzjgnhlb
Hasil dekripsi:
#cA sY ;+k c

Ternyata penghapusan sebuah blok cipher menyebabkan kerusakan parah dalam plaintext.

Dari tiga contoh serangan terhadap blok di atas, dapat disimpulkan bahwa prinsip diffusion oleh Shannon dalam membuat algoritma kriptografi terpenuhi.

## VII. SIMPULAN

Seperti segala yang ada di dunia ini, algoritma ini pun tidak sempurna. Peningkatan selanjutnya yang paling penting untuk dilakukan adalah membuat aloritma ini menjadi tidak terikat oleh bahasa, karena saat ini masih digunakan algoritma pembangkitan bilangan random yang didukung oleh Java, belum dilakukan porting sehingga algoritma ini menjadi bebas-bahasa dan akhirnya bisa diterapkan di mana saja.

Algoritma "Alay-Yielded Octal" ini telah berhasil menjadi algoritma yang memenuhi kebutuhan confusion and diffusion, seperti yang dikemukakan oleh Shannon. Algoritma ini menunjukkan bahwa masih banyak sisi kehidupan yang bila ditelaah lebih jauh akan bisa menjadikan dunia ini semakin berkembang menuju ke arah yang lebih baik, walaupun dimulai dengan cara yang tidak disukai oleh sebagian orang. "Alay-Yielded Octal", yang secara harfiah berarti Oktal yang disebabkan / diberi jalan / dibuatkan jalan oleh Alay, berusaha menunjukkan bahwa selama manusia bisa diam merenung untuk beberapa saat tanpa menghakimi hal yang tidak disetujui, ia pasti bisa menemukan sesuatu yang membangun atau mengambil hikmah dari kejadian yang ada.

## REFERENCES


[1] C.E Shannon, "Communication Theory of Secrecy Systems", 1949.

[2] http://commons.wikimedia.org/wiki/File:Telephone-keypad.png, diakses 24 Maret 2013.


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Maret 2013

R. Purwoko Cahyo Nugroho – 13510014