RANCANGAN METODE ENKRIPSI UNTUK PENGAMAN DATA PADA E-KTP

Mohammad Dimas (13507059)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha nomor 10
e-mail: if17059@students.if.itb.ac.id

ABSTRAK

E-KTP merupakan tanda kependudukan baru yang akan diimplementasikan di Indonesia sebagai pengganti KTP yang lama. Pada dasarnya E-KTP ini memiliki fungsi yang sama dengan KTP sebelumnya yaitu sebagai tanda kependudukan resmi, namun pada E-KTP ini terdapat fungsi yang memungkinkan penggunaannya secara elektronik. Berkaitan dengan fungsi elektroniknya sebagai penyimpan data, maka harus diadakan pengaman terhadap data yang tersimpan di dalamnya, apalagi E-KTP berkaitan dengan ketahanan negara kita.

Dalam pengamanan E-KTP akan digunakan algoritma enkripsi modern. Algoritma enkripsi yang digunakan yaitu block cipher. Block cipher adalah contoh algoritma enkripsi kunci simetris yang beroperasi pada panjang tetap kelompok bit, yang disebut dengan blok. Sebagai contoh, sebuah algoritma enkripsi cipher blok dapat mengambil 128 bit blok plaintext sebagai masukan, dan akan mengeluarkan output berupa 128bit blok ciphertext. Transformasi yang dilakukan didasarkan atas kunci yang diberikan kepada fungsi enkripsi. Hal tersebut maksudnya, jika suatu kunci dengan panjang 128 bit blok di masukan sebagai kunci enkripsi, maka plainteks akan dibagi-bagi sesuai dengan panjang kunci yaitu 128 blok, dan kemudian tiap tiap blok plainteks akan menghasilkan blok-blok yang bersesuaian dengan panjang tiap blok sama dengan panjang kunci yaitu 128bit blok. suatu algoritma enkripsi blok cipher bersifat simetri sedemikian hingga tidak diperlukan dua buah fungsi untuk melakukan enkripsi dan dekripsi, melainkan hanya memerlukan satu fungsi. Dikarenakan proses enkripsi dan dekripsi menggunakan fungsi yang sama, maka harus diimplementasikan metode tertentu dalam enkripsi dan dekripsi agar tidak diperlukan fungsi yang berbeda antara enkripsi dan dekripsi.

Dalam makalah ini penulis bermaksud melakukan pembahasan untuk menemukan metode dan algoritma enkripsi yang tepat digunakan pada pengamanan data di E-KTP. Solusi yang penulis coba tawarkan berupa kombinasi konsep kriptografi klasik dan modern. Konstrain-konstrain yang dimiliki oleh E-KTP itu juga digunakan sebagai bahan pertimbangan metode enkripsi. Pada makalah ini akan dibahas analisis algoritma dan metode yang telah dirancang, analisis akan dilakukan berdasarkan hasil implementasi atau prototipenya.

Kata kunci: E-KTP, Block Cipher, Simetric Algorithm, kriptografi klasik, kriptografi modern.

1. PENDAHULUAN

KTP merupakan tanda yang menyatakan informasi diri sesorang sebagai warga negara indonesia. KTP diterbitkan oleh instansi pelaksana yang berglaku di wilayah Negara Kesatuan Republik Indonesia. KTP menyimpan beberapa informasi penting yang berkaitan dengan pemiliknya, yaitu nomor induk kependudukan, nama lengkap, tempat tanggal lahir, jenis kelamin, agama, status perkawinan, golongan darah, alamat, pekerjaan, kewarganegaraan, foto, masa berlaku, tempat dan tanggal dikeluarkannya KTP, tanda tangan pemilik KTP dan nomor induk pejabat yang menandatanganinya.

KTP sering digunakan untuk melakukan autentifikasi, yaitu memastikan bahwa orang yang bersangkutan berhak atas apa yang akan ia lakukan sesuai dengan identitasnya, sebagai contoh orang ingin mengambil paket atas namanya, maka KTP tersebut harus ditunjukan sebagai bukti bahwa ia adalah orang yang berhak untuk mengambil barang tersebut. KTP enjadi sangat penting dikarenakan apa yang tertulis di KTP merupakan bukti, sehingga ketika seseorang berhasil memalsukan KTP, contohnya memiliki lebih dari satu KTP dengan identitas berbeda-beda, fungsi KTP sebagai identitas tunggal menjadi tidak valid.

E-KTP merupakan bentuk kependudukan warga negara Indonesia di masa yang akan datang. E-KTP pada dasarnya memiliki fungsi yang sama dengan KTP terdahulu.

Pengajuan E-KTP diawali atas banyaknya masalah yang timbul dikarenakan KTP tidak dapat menjadi tanda kependudukan tunggal dimana KTP dapat dengan mudahnya dibuat. Pembuatan KTP itu bukan hanya

melibatkan warga negara indonesia yang bisa memiliki lebih dari satu KTP saja, melainkan juga melibatkan warga negara asing yang tidak berhak atas KTP namun dapat membuat KTP di indonesia, dengan demikian orang yang bersangkutan dapat menikmati seluruh fasilitas yang disediakan untuk warga negara indonesia.

Contoh-contoh kejahatan yang telah terjadi yaitu, pemalsuan identitas oleh para teroris dengan cara memiliki banyak KTP dengan identitas yang berbeda-beda, kejahatan pada kasus Bank Century dimana pelaku memalsukan identitas dengan membuat banyak KTP dan kemudian mengambil uang dari bank sedikit demi sedikit agar tidak mencurigakan, dll.

Hal yang menyebabkan E-KTP berbeda dengan KTP ada pada awalannya yaitu huruf 'E' yang berarti elektronik. Pada kartu identitas warga negara yang baru ini akan ditanamkan fitur elektronik untuk menyimpan data. Data yang disimpan yaitu data sidik jari yang mencirikan pemiliknya, dan disimpan pada chip yang tertanam pada E-KTP.

Gambar 1Contoh smart card, yaitu kartu perbankan yang mengimplementasikan chip

E-KTP secara garis besar dapat disamakan dengan kartu perbankan. Kartu perbankan telah terlebih dahulu mengaplikasikan penggunaan chip atau magnetic card di dalamnya sebagai media penyimpanan data. Perbedaannya terdapat pada ukuran dan data yang disimpan.

Chip yang digunakan E-KTP berukuran 4kb, dan dipergunakan untuk menyimpan data dua buah sidik jari, yaitu sidik jari jempol dan telunjuk kanan. Walaupun pada E-KTP hanya disimpan dua buah data sidik jari, pada database pusat disimpan kesepuluh data sidik jari pemiliknya.

Dengan adanya sidik jari yang disimpan akan mengubah proses autentifikasi yang sebelumnya hanya dengan mencocokan dengan mata saja antara foto dengan orang yang menggunakannya, menjadi dengan autentifikasi yang lebih kompleks dan aman.

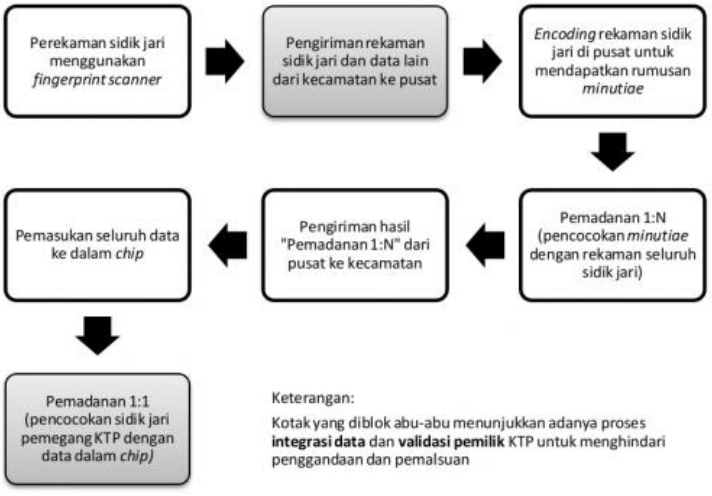
Cara autentifikasinya E-KTP dengan cara melakukan pemindaian orang yang bersangkutan dan kemudian mencocokan dengan data yang ada di E-KTP dan juga database pusat.

Dalam penggunannya E-KTP tidak menggunakan *magnetic card*, melainnkan RFID, yaitu radio frequency identification. Dengan diterapkannya RFID memungkinkan E-KTP dapat bertahan lebih lama dikarenakan tidak terjadi gesekan antara *reader* dengan E-KTP, *contactless*.

Secara singkat, ketika sesorang akan membuat E-KTP untuk pertamakalinya, maka diperlukan data sidik jarinya untuk kemudian direkam pada basis data pusat dan pada E-KTP itu sendiri. Salah satu fungsi penyimpanan data pada basis data pusat adalah untuk memeriksa apakah sidik jari yang baru masuk tersebut sudah pernah disimpan atau belum. Hal tersebut dilakukan untuk mencapai tujuan utama pembuatan E-KTP yaitu sebagai identitas tunggal, jika data sidik jari telah ada, berarti orang yang bersangkutan telah memiliki E-KTP sebelumnya, dan jika belum tersimpan berarti ia berhak untuk memperoleh E-KTP.

Tahapan yang dilakukan yaitu :

1. Perekaman sidik jari menggunakan *fingerprint scanner*
2. Pengiriman data rekaman sidik jari dari daerah setempat ke pusat
3. Encoding rekaman sidik jari di pusat
4. Pembandingan sidik jari hasil kiriman(sidik jari pembuat KTP baru) dengan data yang telah ada(sidik jari lain), 1:N
5. Pengiriman haril pemadanan ke daerah lokal
6. Memasukan data ke dalam *chip* E-KTP
7. Memadankan sidik jari orang yang bersangkutan dengan data yang baru dimasukan ke E-KTP.



Gambar 2Proses perekaman sidik jari pada E-KTP

Dilihat dari ketujuh tahapan pembuatan E-KTP tersebut, terdapat kemungkinan celah keamanan. Celah keamanannya yaitu berupa memasukan secara ilegal data sidik jari, dan informasi lain yang diinginkan ke dalam E-KTP. Dengan demikian memungkinkan seseorang yang

bukan pemilik E-KTP untuk memalsukan identitasnya dikarenakan sidik jari yang disimpan di dalam E-KTP dapat ia manipulasi.

Solusinya ialah dengan cara mengimplementasikan algoritma enkripsi pada saat penulisan data di chip. Dengan demikian ketika data yang tertulis di dalam E-KTP diperlukan, maka perlu dilakukan proses dekripsi, dimana ketika data yang dimasukan secara paksa tidak mengimplementasikan algoritma yang sama, maka saat proses dekripsi dilakukan, justru data tersebut tidak dapat dibaca.

Algoritma enkripsi yang akan diterapkan ialah algoritma enkripsi kunci simetrik atau biasa dikenal dengan block cipher. Kontribusi penulis dalam makalah ini yaitu merancang fungsi enkripsi, merancang fungsi pembangkitan kunci, dan analisis terhadap implementasinya.

2. DASAR TEORI

Kriptografi adalah suatu cabang ilmu yang dipergunakan untuk menyembunyikan pesan dari orang yang tidak berhak. Pesan disembunyikan dengan cara mengubah bentuk pesan menjadi bentuk yang tidak bisa dimaknai oleh orang atau biasa disebut cipertext, proses tersebut dikenal dengan nama enkripsi. Pesan yang telah disembunyikan dapat dimaknai kembali oleh orang yang berhak dengan menggunakan suatu proses yang disebut dengan dekripsi, dekripsi mengembalikan bentuk pesan dari ciphertext kembali menjadi plaintext. Proses pembalikan tersebut memerlukan suatu kunci, dimana ketika kunci salah maka pesan tidak dapat dikembalikan.

Dalam perkembangan ilmu kriptografi terdapat dua algoritma kriptografi yang dikenal, yaitu algoritma kriptografi klasik dan algoritma kriptografi modern. Terdapat perbedaan yang mendasat dari keduanya, yaitu algoritma kriptografi klasik bekerja pada mode operasi karakter, sedangkan algoritma kriptografi beroperasi pada mode bit.

2.1 KONSEP ONE-TIMEPAD

Contoh algoritma kriptografi klasik ialah Vigenere Cipher. Vigenere cipher melakukan substitusi suatu berkas plainteks dengan suatu kunci yang berulang sesuai kebutuhannya dalam memenuhi panjang berkas plainteks. Maksud dari substitusi yaitu ketika plainteks ‘A’ bertemu dengan kunci ‘B’ maka akan dihasilkan cipherteks ‘C’. Cipherteks C diperoleh dengan memberi nilai alfabet yaitu dari 0-25, yang kemudian akan dijumlahkan antara plainteks dan kunci sehingga diperoleh nilai untuk cipherteks, nilai akan dikenai operasi mod dengan tujuan menjaga nilai akan tetap berada pada range yang terdefinisi, dalam hal ini 0-25.

Dalam dunia kriptografi hal yang dicari yaitu algoritma enkripsi dan dekripsi yang tidak dapat dipecahkan, yaitu dimana ciphertext tidak dapat dimaknai tanpa mengetahui kunci yang digunakan, dan kunci yang digunakan tidak dapat dianalisis berdasarkan berkas ciphertext.

Hal tersebut diimplementasikan pada algoritma enkripsi one-time pad, dimana digunakan kunci yang sangat panjang dalam hal ini sepanjang berkas plainteks. Metode enkripsi dan dekripsi yang digunakan sendiri serupa dengan metode yang digunakan oleh vigenere cipher.

Dengan kunci yang sangat panjang dan dibangkitkan secara acak, one-pad termasuk algoritma kriptografi yang tidak terpecahkan, namun algoritma ini memiliki kelemahan yaitu kunci yang terlalu panjang sangat sulit untuk disimpan dan didistribusikan.

2.2 KONSEP BLOK CIPHER

Contoh dari algoritma kriptografi modern adalah cipher block yang mengenkripsi block plainteks dengan cara membagi-baginya menjadi panjang bit tertentu. Panjang block yang diproses disesuaikan dengan panjang kunci. Algoritma kriptografi ini beroperasi pada empat mode yang berbeda yaitu ECB(Electronic Code Block), CBC(Cipher Block Chaining), CFB(Cipher Feedback), dan OFB(Output Feedback).

Perbedaan keempatnya terdapat pada proses peng-enkripsian dan pendekripsian.

Untuk mode ECB, setelah plainteks dibagi menjadi block-block, tiap block akan di proses secara independen, tidak terpengaruh oleh proses pada block lain, hal tersebut berlaku untuk proses enkripsi dan dekripsi.

Pada mode CBC, hampir mirip dengan ECB, namun untuk proses enkripsi hasil proses suatu block akan digunakan untuk block setelahnya. Jadi ketika dilakukan proses enkripsi, hasil enkripsi blok ke i akan menjadi blok ciper ke i, dan blok cipher ke i itu akan digunakan dalam proses pembuatan blok cipher ke i+1.

Pada mode CFB proses enkripsi dilakukan untuk setiap n bit dimana n <= m, dengan m ukuran blok, pada mode ini kesalahan 1 bit pada blok plainteks akan berefek kepada blok yang berkorespodensi dan juga kepada blok blok setelahnya.

Pada mode OFB serupa dengan CFB namun kesalahan 1bit pada blok plainteks hanya mempengaruhi blok yang berkorespodensi saja.

Mode operasi kriptografi berbasis bit relatif lebih aman dan sulit dipecahkan dikarenakan perubahan suatu berkas dari plainteks menjadi cipher teks menjadi tidak terbatas, dibandingkan dengan mode operasi yang berbasiskan karakter dimana pesan disimpan sebagai suatu karakter yang menyebabkan wilayah substitusi sangat terbatas, yaitu sejumlah karakter ASCII saja contohnya.

3. BATASAN MASALAH

Pada makalah ini masalah dibatasi pada penggunaan dua konsep algoritma yaitu metode modern dan klasik. Berkas yang akan dienkripsi pada E-KTP akan dienkripsi bit per bit yaitu mengikuti konsep kriptografi modern, dan penggunaan kunci sepanjang-panjangnya, maksimal sepanjang berkas plainteks mengikuti konsep kriptografi klasik one-time pad.

Berdasarkan suatu sumber, menyebutkan bahwa data yang disimpan pada E-KTP hanya berupa dua buah jari, dimana ukuran satu buah jari sebesar 0,4kb saja. Melihat hal tersebut penulis berasumsi algoritma dapat dibuat sekompleks mungkin dikarenakan data yang akan dienkripsi berukuran kecil.

4. RANCANGAN ALGORITMA

Algoritma yang penulis ajukan dibuat berdasarkan kelebihan dua buah konsep, yaitu konsep kriptografi modern dan kriptografi klasik. Implementasi hasilnya yaitu berupa algoritma kriptografi kunci simetrik dengan panjang kunci yang merupakan panjang blok sama dengan panjang ukuran plainteks. Pada dasarnya panjang kunci yang juga merupakan panjang blok yang akan diproses menentukan performa algoritma enkripsi, semakin panjang performanya semakin menurun, namun dengan asumsi algoritma akan diimplementasikan pada E-KTP dengan data yang dienkripsi berkisar 0,4kb-4kb, waktu eksekusi seharusnya menjadi relatif tidak terlalu lama.

Mode operasi fungsi yang penulis ajukan adalah ECB, hal tersebut didasarkan pada anggapan bahwa berkas hanya akan menjadi satu buah blok, dengan demikian mode lain yang melibatkan blok sebelumnya untuk proses enkripsi dan dekripsi menjadi tidak terlalu berpengaruh.

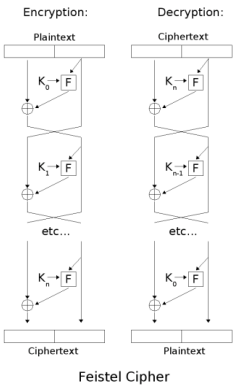
Untuk mewujudkan algoritma tersebut dibutuhkan dua hal, yaitu perancangan fungsi enkripsi dekripsi, dan juga fungsi pembangkitan kunci.

4.1 PERANCANGAN FUNGSI

Fungsi yang akan diimplementasikan akan dilewatkan pada suatu jaringan Feistel sehingga hanya diperlukan satu buah fungsi yang dapat berlaku untuk enkripsi maupun dekripsi.

Feistel network ditemukan oleh fisikawan dan kriptografer Horst Feistel. Ia menemukan jaringan feistel ketika sedang bekerja untuk IBM. Sangat banyak algoritma blok cipher yang menggunakan skema pemrosesan feistel network ini.

Semakin banyak dan semakin rumit fungsi yang dipakai maka akan membuat semakin kuat suatu algoritma kriptografi yang dihasilkan. Namun dengan bertambahnya jumlah dan kerumitan fungsi yang digunakan akan mengurangi performa dalam prosesnya.

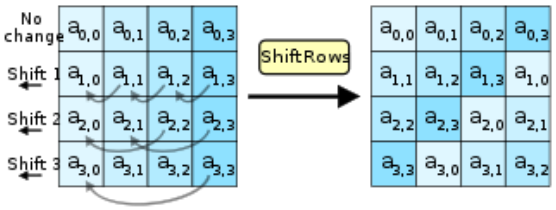


Gambar 3 skema umum jaringan feistel

Pada makalah ini penulis mengajukan tiga buah fungsi. Yaitu menjumlahkan blok dengan kunci, memindahkan posisi elemen blok, dan mengoperasikan blok dengan dirinya sendiri.

4.1.1 Scramble Row

Pemindahan posisi elemen pada blok merupakan tahapan yang dilakukan untuk membuat urutan blok berbeda dari sebelumnya. Proses ini didasarkan pada operasi Shift Row pada algoritma Rijndael yang juga merupakan algoritma kriptografi modern.



Gambar 4operasi shift row pada Rijndael

Pada algoritma tersebut pemrosesan dilakukan terhadap array bytes berukuran 4x4. Array byte akan dikenai operasi shift row, pada operasi tersebut setiap kolom akan mengalami pergeseran berdasarkan barisnya. Sebagai contoh, pada gambar di atas array pada baris pertama tidak mengalami pergeseran, array pada baris ke dua mengalami pergeseran sebesar 1 kolom ke sebelah kiri, array pada baris ke tiga mengalami pergeseran sebesar 2 kolom ke sebelah kiri, dan begitu seterusnya untuk kolom kolom setelahnya, jumlah pergeseran akan semakin besar. Jika pergeseran elemen array menyebabkan index elemen menjadi tidak terdefinisi, maka elemen akan digeser ke sebelah kanan dengan perhitungan index yang dituju ditambah dengan jumlah kolom.

Tujuan pergeseran adalah agar blok semakin sulit untuk ditebak polanya, dengan demikian algoritma enkripsi dapat menjadi lebih kuat.

Untuk proses pergeseran ini penulis mengajukan pergeseran dengan mekanisme sebagai berikut, merepresentasikan blok ke dalam bentuk array dua dimensi, array yang dibentuk ialah array dengan ukuran 4 * n, dimana n bergantung dengan total panjang blok. pemilihan angka 4 sebagai pengali didasarkan pada perhitungan bahwa panjang setiap blok pasti habis dibagi 8 dan dikarenakan blok dibagi menjadi dua untuk dimasukan ke jaringan fiestel, maka bilangan yang pasti habis membagi panjang blok adalah 4.

Pergeseran dilakukan dengan cara mengiterasi kolom array, untuk tiap kolom array dilakukan iterasi dari index ke offset menuju index ke 0, dan dilanjutkan dengan iterasi dari index ke offset menuju index ke 3. Untuk setiap iterasi dilakukan penambahan elemen yang dikunjungi pada blok lama ke blok baru dengan penambahan yang terurut. Offset untuk kolom setelahnya akan ditambah jika offset pada kolom sebelumnya kurang dari tiga, dan offset akan menjadi nol jika offset pada elemen sebelumnya telah mencapai tiga.

Berikut adalah ilustrasi dari pergeseran:

1	2	3	4	5	6	7	8	9	10	11	12	...	16

Tabel 1bitarray awal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Tabel 2 bitarray direpresentasikan sebagai matriks

1	5	9	13
2	6	10	14
7	3	11	15
12	8	4	16

Tabel 3matriks yang telah dikenai operasi

1	5	9	13	2	6	10	14	7	3	11	15	...	16

Tabel 4 bitarray hasil

Blok direpresentasikan sebagai suatu bitarray. Bitarray diubah bentuknya menjadi suatu matriks atau array 2 dimensi. Dilakukan operasi perpindahan dengan iterasi seperti diatas, hasilnya ialah matriks yang baru. Matriks yang baru diubah kembali menjadi suatu bitarray.

4.1.2 Mix Own

Operasi selanjutnya ialah operasi terhadap dirinya sendiri. Operasi ini bertujuan sama dengan operasi sebelumnya, yaitu untuk menghilangkan atau menyamarkan pola-pola yang mungkin terbentuk.

Operasi ini, sama dengan operasi sebelumnya, blok akan direpresentasikan dalam bentuk array dua dimensi dengan ukuran 4 * n, dimana n adalah bilangan yang disesuaikan dengan panjang blok. Pemilihan angka empat sama juga memiliki alasan yang sama, yaitu dikarenakan tiap blok pasti habis dibagi 8 dan blok yang dimasukan ke dalam fungsi ialah setengah dari blok yang berarti pasti habis dibagi 4.

Operasi mixown akan mengiterasi kolom array, untuk setiap kolom yang diiterasi akan di xor kan dengan tetangganya pada array dengan index kolom dan baris lebih satu darinya. Jika penambahan kolom atau baris menyebabkan index yang dituju tidak terdefinisi maka index yang dituju akan kembali ke index 0. Sebagai contoh ketika index baris yang sedang ditunjuk ialah index 3, maka elemen yang seharusnya di xor dengan dirinya yaitu elemen pada array dengan index kolom+1 dengan baris+1, namun dikarenakan index baris+1 itu tidak terdefinisi, atau tidak ada di dalam array, maka index barisnya menjadi 0, sehingga elemen yang dituju memiliki index kolom +1 dan index baris 0. Hal yang sama untuk index kolom, ketika index kolom yang dituju lebih dari index yang terdefinisi maka index kolom menjadi 0.

Berikut adalah ilustrasi dari operasi:

1	2	3	4	5	6	7	8	9	10	11	12	...	16

Tabel 5 bitarray awal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Tabel 6 bitarray direpresentasikan sebagai matriks

1 xor 6	2 xor 7	3 xor 8	4 xor 5
5 xor 10	6 xor 11	7 xor 12	8 xor 9
9 xor 14	10 xor 15	11 xor 16	12 xor 13
13 xor 2	14 xor 3	15 xor 4	16 xor 1

Tabel 7 matriks yang telah dikenai operasi

Serupa tahap sebelumnya blok direpresentasikan sebagai suatu bitarray. Bitarray diubah bentuknya menjadi suatu matriks atau array 2 dimensi. Dilakukan operasi xor terhadap tetangga elemen yang sedang diiterasi, hasilnya disimpan pada matriks yang baru . Matriks yang baru diubah kembali menjadi suatu bitarray.

4.1.3 Key Operated

Operasi selanjutnya ialah operasi terhadap kunci. Tahapan ini bisa dibilang tahapan yang paling wajib untuk dilakukan. Hal tersebut disebabkan tahapan ini satu satunya tahap yang melibatkan kunci dalam proses peng-enkripsian maupun pendekripsian.

Kunci dalam proses di dalam algoritma ini menentukan panjangnya blok. Kunci yang dioperasikan terhadap blok bukanlah kunci yang dimasukan oleh user, melainkan kunci yang dibangkitkan berdasarkan masukan user. Di dalam proses ini fungsi akan memproses tiap blok sebanyak 10 kali. Dalam tiap proses akan digunakan kunci internal yang berbeda-beda.

Operasi blok yang dikenai fungsi ialah operasi xor. Operasi xor yang digunakan memanfaatkan fungsi bawaan c# yaitu Xor yang dapat melakukan xor antara dua bitarray dengan panjang yang sama. Bitarray merupakan representasi blok-blok yang akan diproses.

4.2 Pembangkitan Kunci

Seperti yang telah disinggung di atas bahwa pada algoritma yang penulis usulkan akan menggabungkan kelebihan dua algoritma. Pada sub bab sebelumnya telah dibuat 3 fungsi yang telah mengimplementasikan konsep block cipher, maka selanjutnya yang harus dilakukan ialah mengimplementasikan konsep one-time pad. One-time pad, yaitu konsep dimana dibangkitkan suatu kunci secara random dengan panjang sama dengan panjang blok, namun implementasi pada algoritma ini ialah kunci dengan panjang sama dengan panjang berkas dibangkitkan dengan suatu string lain yang panjangnya bebas. Setelah memperoleh string dengan panjang yang diinginkan, akan dibuat sepuluh buah kunci internal berdasarkan suatu fungsi.

Tahapan pembentukan kunci:

1. Membangkitkan string dengan panjang sama dengan panjang berkas yang akan diproses
2. Membangkitkan sepuluh buah kunci internal

4.2.1 Pembangkitan String kunci

Proses ini pada dasarnya akan membangkitkan string sesuai dengan panjang yang diinginkan. Dalam hal ini panjang yang diinginkan yaitu sama dengan panjang berkas yang akan diproses. String kunci yang dihasilkan harus benar-benar acak.

Untuk membangkitkan string tersebut penulis membuat fungsi dengan dua parameter, string kunci lain sebagai umpan dan panjang string kunci keluaran yang diharapkan.

Algoritma fungsi memanfaatkan fungsi random yang dibawa oleh C#. Fungsi random diumpankan integer bebas, dalam hal ini panjang kunci yang diumpankan ditambah dengan panjang kunci keluaran yang diharapkan. Fungsi random sebenarnya tidak sepenuhnya random, ia akan menghasilkan suatu deret bilangan yang sama jika umpan integer yang diberikan sama.

String yang dihasilkan ialah, iterasi string kunci yang diumpankan kemudian ditambah dengan bilangan random yang dihasilkan dan di mod 128. Operasi mod dilakukan untuk membatasi karakter yang dihasilkan adalah karakter yang printable.

Pembagian operasi algoritma fungsi menjadi dua, yaitu ketika kunci yang diumpankan panjangnya kurang dari panjang kunci keluaran yang diharapkan dan ketika panjang kunci yang diumpankan lebih panjang dari panjang kunci keluaran yang diharapkan.

Untuk kasus pertama, penanganan dilakukan dengan mengulang kunci yang diumpankan untuk memenuhi panjang yang dibutuhkan. Untuk kasus kedua, string kunci yang diumpankan akan dipotong menjadi string-string kecil dengan panjang sesuai dengan panjang kunci string keluaran yang diharapkan, setelahnya string-string tersebut dijumlahkan.

Dengan demikian dapat dipastikan seluruh kemungkinan telah dipenuhi, dengan kata lain tidak mungkin ada dua string umpan yang beda menghasilkan satu string kunci keluaran yang sama.

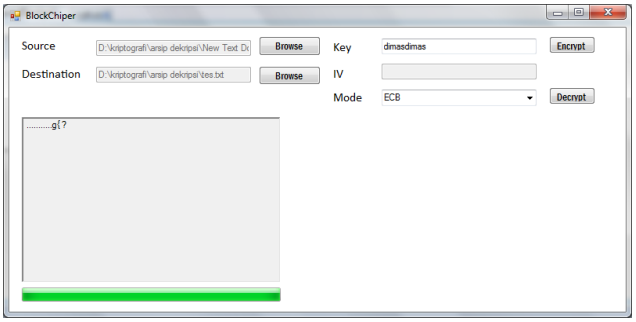
4.2.1 Pembangkitan Kunci internal

Kunci internal yang dibangkitkan akan berbentuk list of bitarray yang berisi 10 elemen. Kunci yang digunakan untuk tiap iterasi proses enkripsi maupun dekripsi berbeda. Untuk proses enkripsi kunci yang dipakai dari kunci pertama hingga kunci ke 10, dan untuk proses dekripsi kebalikannya yaitu dari 10 hingga yang pertama.

Kunci yang dibangkitkan didasarkan atas string masukan yang diperoleh dari hasil sub bab sebelumnya. Untuk key pertama digunakan setengah dari bit array kunci bagian kiri. Untuk kunci selanjutnya digunakan fungsi penjumlahan. Fungsi penjumlahan akan menjumlahkan kunci sebelumnya dengan bagian kanan kunci yang pertama. Fungsi penjumlahan yaitu fungsi yang meng xor kan dua bitarray pada posisi yang bersesuaian.

5. Implementasi

Sejauh ini penulis telah berhasil mengimplementasikan kedua konsep kriptografi, block cipher dan one time pad. Pada bagian ini penulis akan menganalisis hasil dari metode kriptografi yang akan diujikan tehadap file gambar. Dalam implementasinya gambar yang digunakan ialah gambar dengan ukuran 0,4 kb. Sebagai pembanding performa aplikasi penulis juga akan memproses berkas-berkas dengan ukuran kecil dan sedang.



Gambar 5 antarmuka program blok cipher+one-timepad

Pengujian pertama dilakukan kepada gambar berikut.
. Gambar tersebut diasumsikan sebagai gambar yang akan disimpan pada chip E-KTP. Gambar berukuran sekitar 0,4 kb.

Gambar 6gambar sample

Kunci yang dimasukan yaitu, “dimasdimas”, kunci yang dihasilkan tersebut memiliki panjang sama dengan panjang file, string kunci yang dibangkitkan yaitu

```
Yi_9zvC;1v$CQA>,-
O'y'vax`}yD0v,m_"FYxASdqwo\u)6B2iY_eO@o#zV
%!@7uD]cxn9e1F36"9!ZuZ?JGS#E^_]!wQcwYek\XTVH["ISB
-uk(x-W\$tFH;3>4@hkg>i>ZI-P
,sH1P1Vs!y,C6JQ}D7D<L2>@_x"M^,kqe^RCJ8'&hH=LXUV}?
;K8j;T'S3^Q<(8z1MaY}0`*:MU2xzn&V)8R[h%N8N|=c[dn5E
J-
(#*6TU#Gs>;\Ng|GtnNC\9DW5WJaoa2YTowh$%%)yb]".Ts}l
?/Cx/Q$[U0}8FZ@/m'`v7QtlMJ/YFWZwJw_Uz]PFu;R}.qh&G
"Bb2+chFSVjD^3)\)zL&x.DTE/v3sihb60HDysD>}*T&g0:<O
E3pJ&W"w<p'sW^mv6)\hgo/k3+uTLp8$]oTY
;D\PS`7<LMEr3[5W_eW2!Kxc[MWV-
Ei?:z*f4=?>r]k_^[F3.R                    8yR-jXuw-
$l4+p)m6\U?OEDF>l>/')XN\SLRdwF0QB<A(]^qdgE[[AR'Z6
u Nk{2MS[aRP=;AS
```

pada proses eksekusi, pemrosesan berkas dari plainteks menjadi ciphertext hanya berkisar 0,1 s.

Untuk berkas dengan ukuran sekitar 4bytes. Hampir tidak ada waktu tunggu dalam eksekusi proses.

Untuk file dengan ukuran 170kb. Waktu eksekusi menjadi sekitar 5 menit. Sebagai pembanding, untuk memproses file yang sama algoritma block cipher (tidak menggunakan one-time-pad), hanya membutuhkan waktu 3 s.

6. Kesimpulan

Setelah melakukan analisis terhadap beberapa sample penulis menyimpulkan:

- kombinasi dua konsep algoritma tersebut dapat menjadi algoritma yang cukup powerful dalam dunia kriptografi
- dari masalah waktu eksekusi, algoritma ini memiliki kekurangan, yaitu untuk file berukuran sedang(100kb ke atas), algoritma ini sangat lambat
- lambatnya algortima ini menyebabkan tidak mungkin diimplementasikan untuk mengenkripsi berkas-berkas dengan ukuran besar, namun untuk masalah data di dalam E-KTP algoritma ini menjadi sangat cocok mengingat ukuran data yang disimpan relatif kecil.

REFERENSI

[1] M. Liskov, R. Rivest, and D. Wagner, "Tweakable Block Ciphers", Crypto 2002 PDF
[2] http://hack87.blogspot.com/2010/01/biar-hemat-chip-e-ktp-cuma-4-kb.html
[3] http://www.unicode.org/charts/

[4] http://users.telenet.be/d.rijmenants/en/onetimepad.htm
[5] http://egadioniputri.wordpress.com/2010/02/19/apa-dan-mengapa-e-ktp/

LAMPIRAN

Fungsi untuk mengacak bitarray

```
public BitArray Scramble(BitArray input)
        {//mengubah susunan bitarray
            int MaxKolom = input.Length / 4;
            int destIndex = 0;
            int offset = 0;
            BitArray output = new BitArray(input.Length);
            for (int i = 0; i < MaxKolom; i++)
            {
                for (int j = offset; j >= 0; j--)
                {
                    output.Set(destIndex, input.Get(i + j * MaxKolom));
                    destIndex++;
                }
                for (int j = offset; j < 3; j++)
                {
                    output.Set(destIndex, input.Get(i + j * MaxKolom));
                    destIndex++;
                }
                if (offset < 3)
                {
                    offset++;
                }
                else
                {
                    offset = 0;
                }
            }
            return output;
        }
```

Fungsi untuk menambahkan dengan diri sendiri

```
public BitArray Mix(BitArray input)
        {//menambahkan kolom dengan kolom tetangga
            int MaxKolom = input.Length / 4;
            BitArray output = new BitArray(input.Length);
            for (int i = 0; i < MaxKolom; i++)
            {
                for (int j = 0; j < 4; j++)
                {
                    int temp = j * MaxKolom;
                    int temp2 = i + temp;
                    int target = 0;
                    if (j == 3)
                    {
                        if (i == (MaxKolom - 1))
                        {
                            target = 0;
                        }
```

```
                        else
                        {
                            target = i+1;
                        }
                    }
                    else
                    {
                        if (i == (MaxKolom - 1))
                        {
                            target = (j + 1) * MaxKolom;
                        }
                        else
                        {
                            target = (j + 1) * MaxKolom + i + 1;
                        }
                    }
                    output.Set(temp2, input.Get(temp2) ^ input.Get(target));
                }
            }
            return output;
        }
```

Fungsi untuk membangkitkan string kunci

```
public String genKeyString(String key,int length)
        {
            Random random = new Random(key.Length+length);
            String s = "";
            for (int i = 0; i < length; i++)
            {
                Char newchar = (Char)(((int)key[i % key.Length] +
random.Next(key.Length+length)) % 94+32);
                s += newchar;
            }
            if (length < key.Length)
            {
                char[] result2=s.ToCharArray();
                String s2 = "";
                for (int i = 0; i < key.Length; i++)
                {
                    result2[i % length] = (Char)((int)key[i % key.Length] +
random.Next(key.Length+length));
                }
                for (int i = 0; i < length; i++)
                {
                    Char newchar = (char)(((int)result2[i]+(int)s[i])%94 +32);
                    Console.WriteLine(newchar);
                    s2 += newchar;
                }
                s = s2;
            }
            Console.WriteLine("kunci: " + s+"--");
            return s;
        }
```