# Penerapan Aritmatika Modulo dan Transposisi Matriks untuk Algoritma Kriptografi *Columnar Transposition*

**Monterico Adrian - 13505036**
Program Studi Teknik Informatika ITB, Bandung 40132, email: monterico.adrian@gmail.com

***Abstrak*** *– Makalah ini membahas dua algoritma kriptografi columnar transposition yaitu Simple Columnar Transposition (SCTR) dan Keyed Columnar Transposition (KCTR) yang menerapkan konsep aritmatika modulo dan transposisi matriks pada proses enkripsi dan dekripsinya. SCTR dan KCTR adalah dua algoritma kriptografi klasik untuk pesan berbentuk teks yang proses enkripsi dan dekripsinya menggunakan perhitungan modulo panjang teks pesan dan perubahan posisi karakter-karakter teks pesan tersebut secara transposisi. Fungsi digunakannya dua konsep ini yaitu untuk pengaturan perubahan posisi karakter-karakter dalam teks pesan. Proses enkripsi dan dekripsi pada SCTR dan KCTR memakai sebuah kunci yang sama atau biasa disebut kunci simetri (kunci non-publik).*

***Kata Kunci****: kriptografi, aritmatika modulo, transposisi matriks, Simple Columnar Transposition, Keyed Columnar Transposition,.*

## 1. PENDAHULUAN

Ada beberapa algoritma kriptografi yang menerapkan konsep aritmatika modulo. Contohnya yang paling kuat dan terkenal hingga saat ini adalah algoritma *RSA* (*Rivest-Shamir-Adleman*). Lalu ada beberapa algoritma kriptografi juga yang menerapkan konsep transposisi matriks. Tetapi, ternyata hanya sedikit saja algoritma kriptografi yang dapat menggabungkan kedua konsep tersebut. Contohnya adalah algoritma *SCTR* dan *KCTR*, walaupun penerapan dua konsep tersebut hanya dalam skala kerumitan yang rendah dan tingkat penggunaan yang sedikit. Hasilnya, kekuatan algoritma *SCTR* dan *KCTR* pun menjadi tidak begitu kuat. Kekuatan algoritma *SCTR* dan *KCTR* juga tidak bergantung pada konsep aritmatika modulo dan transposisi matriks yang dilakukan, tetapi bergantung pada panjang teks pesan dan kunci simetri yang dipilih. Konsep aritmatika modulo dan transposisi matriks hanya membantu dalam proses enkripsi dan dekripsi, yaitu untuk mengatur perubahan posisi karakter-karakter yang terdapat pada teks pesan. Tetapi walaupun disebut sebagai algoritma kriptografi yang klasik dan relatif mudah dipecahkan, tetap saja dua algoritma kriptografi *columnar transposition* ini adalah algoritma yang melegenda dan telah diakui sebagai sumber inspirasi para perancang algoritma kriptografi pada masa setelahnya. Penjelasan lebih rinci tentang algoritma *SCTR* dan *KCTR* akan dibahas dalam bab metode.

## 2. METODE

### 2.1. Enkripsi *Simple Columnar Transposition*

*SCTR* dibuat sekitar tahun 1940-an. Pada waktu itu *SCTR* digunakan oleh Sekutu untuk menandingi alat kriptografi buatan Jerman *enigma* untuk menjaga kerahasiaan pesan yang dikirimkan pada Perang Dunia II. *SCTR* mempunyai batasan yaitu teks pesan hanya dapat berisi huruf alfabet dan semuanya harus ber-*case* sama (huruf kapital semua atau huruf kecil semua). Metode enkripsi *SCTR* cukup sederhana. Pertama, hitung panjang teks pesan yang ingin didekripsi, misal bernilai $n$. Kemudian pilih sebuah bilangan bulat $p$, dengan nilai $1 < p < n$. Nilai $p$ inilah yang akan menjadi kunci simetri. Lalu hitung dua buah bilangan bulat $x$ dan $y$, dengan rumus perhitungan $x = n \text{ div } p$ dan $y = n \text{ mod } p$. Tiga buah bilangan $p$, $x$, dan $y$ inilah yang akan menentukan matriks yang akan dibentuk. Setelah itu buat sebuah matriks dengan jumlah kolom $p$ dan jumlah baris $(x+1)$, tetapi bila nilai $y = 0$ maka jumlah baris adalah $x$. Selanjutnya, letakkan karakter-karakter teks pesan ke dalam elemen-elemen matriks satu demi satu secara berurutan, dimulai dari elemen matriks paling kiri atas hingga elemen matriks kanan bawah dengan arah penempatan dari kiri ke kanan. Bila nilai $y > 0$, maka tentu akan ada elemen-elemen matriks yang bersisa atau kosong. Elemen-elemen matriks ini dinamakan *nulls*. Berikutnya ada dua pilihan metode disini. Metode pertama dinamakan *irregular case*, yaitu membiarkan elemen-elemen matriks tersebut tetap kosong. Metode kedua dinamakan *regular case*, yaitu menambahkan karakter-karakter *dummy* pada elemen-elemen matriks tersebut. Karakter-karakter *dummy* ini dinamakan *pad*. Karakter-karakter *dummy* ini juga mempunyai batasan, yaitu harus ber-*case* sama dengan teks pesan dan hanya dapat berupa huruf alfabet. Setelah semua karakter teks pesan telah terisi pada elemen-elemen matriks, maka matriks pun terbentuk. Selanjutnya, transposisikan matriks yang telah terbentuk. Setelah transposisi maka chiperteks pun otomatis terbuat. Chiperteks didapatkan dengan cara mengambil satu demi satu karakter-karakter dalam elemen-elemen matriks secara berurutan dari elemen matriks paling kiri atas hingga elemen matriks kanan bawah, dengan arah pengambilan dari kiri ke kanan. Akhirnya, selesailah proses enkripsi dengan algoritma *SCTR*. Contoh proses enkripsi dengan algoritma *SCTR* dapat dilihat pada bab hasil dan pembahasan.

### 2.2. Dekripsi *Simple Columnar Transposition*

Proses dekripsi pada algoritma *SCTR* mempunyai metode yang hampir sama dengan proses enkripsinya. Proses dekripsi *SCTR* ini tentu saja harus melibatkan kunci simetri yang telah dipilih sebelumnya pada waktu melakukan proses enkripsinya. Metode dekripsi *SCTR* dimulai dari menghitung panjang chiperteks, misal bernilai *n*. Diketahui kunci simetri adalah *p*. Kemudian hitung dua buah bilangan bulat *x* dan *y* yang didapatkan dari $x = n \; div \; p$ dan $y = n \; mod \; p$. Setelah itu buat sebuah matriks dengan jumlah kolom *x* dan jumlah baris *p*. Berikutnya berbeda dengan proses enkripsi *SCTR*. Sebelum memasukkan karakter-karakter chiperteks pada elemen-elemen matriks, dilakukan sebuah langkah untuk mengetahui metode enkripsi *SCTR* yang dipakai. Nilai *y* dibutuhkan kembali untuk menentukan apakah enkripsi *SCTR* menggunakan metode *regular case*, *irregular* case, atau tidak menggunakan keduanya (artinya teks pesan memang mempunyai panjang *n*). Jika $y = 0$, maka enkripsi *SCTR* tersebut mungkin menggunakan metode *regular case* atau tidak menggunakan keduanya, dan jika $y>0$ maka enkripsi *SCTR* tersebut pasti menggunakan *irregular case*. Metode yang digunakan ini akan menentukan pula cara dekripsi *SCTR* tersebut. Cara dekripsi *SCTR* ini tidak lain adalah cara penempatan karakter-karakter chiperteks pada elemen-elemen matriks agar proses dekripsi dapat dijalankan. Langkah selanjutnya adalah bergantung dari hasil metode yang didapatkan. Jika metode yang didapatkan adalah *regular case* atau tidak keduanya, maka langkah selanjutnya adalah menempatkan karakter-karakter chiperteks pada elemen-elemen matriks secara berurutan, dimulai dari kiri atas hingga kanan bawah, dengan arah penempatan dari kiri ke kanan. Tetapi jika metode yang didapatkan adalah *irregular case*, maka diperlukan penanganan tambahan lebih dulu. Penanganan tambahan ini juga melibatkan nilai *y*. Penanganan tambahan ini adalah membuat satu kolom baru setelah kolom terakhir matriks tetapi dengan banyak elemen-elemen matriks yang dapat diisi sejumlah *y* saja, dan biarkan elemen-elemen matriks sisanya kosong dan tandai agar tidak dapat diisi bila perlu. Selanjutnya barulah menempatkan karakter-karakter chiperteks pada elemen-elemen matriks secara berurutan, dimulai dari kiri atas hingga kanan bawah, dan dengan arah dari kiri ke kanan. Langkah terakhir adalah mentransposisikan matriks. Setelah itu teks pesan asli atau plainteks bisa didapatkan dengan cara mengambil satu demi satu karakter-karakter dalam elemen-elemen matriks dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah, dengan arah pengambilan dari kiri ke kanan, sama seperti ketika meletakkan karakter-karakter chiperteks pada elemen-elemen matriks. Akhirnya, proses dekripsi dengan algoritma *SCTR* pun selesai. Contoh proses dekripsi dengan algoritma *SCTR* dapat dilihat pada bab hasil dan pembahasan.

### 2.3. Enkripsi *Keyed Columnar Transposition*

*KCTR* dibuat pada tahun 1950-an, sebagai pengganti dan penyempurnaan dari algoritma *SCTR* yang pada waktu tersebut telah dapat dengan mudah dipecahkan oleh kriptanalis karena metodenya yang cukup sederhana. Dengan cara mencoba satu demi satu kunci simetri yang mungkin (kunci simetri terbatas, hanya bilangan diantara satu sampai panjang chiperteks), kriptanalis hanya memerlukan selembar kertas dan pensil untuk menulis semua hasil percobaannya itu. Ditambah lagi bila kriptanalisnya lebih dari satu orang, percobaan memecahkan chiperteks dapat dibagi-bagi berdasarkan bilangan kunci simetri. Oleh karena itu, dibuatlah algoritma *SCTR* baru yang kunci simetrinya tidak menggunakan sebuah bilangan bulat, tetapi menggunakan sebuah kata kunci (*keyword*) atau kata terang. Fungsi kata terang ini jelas untuk lebih memperkuat proses enkripsi dan mempersulit proses kriptanalisis, yaitu dengan cara mengubah urutan transposisi matriks berdasarkan urutan angka yang dimiliki oleh kata terang. Batasan teks pesan dalam *KCTR* sama dengan *SCTR*, yaitu hanya dapat berisi huruf alfabet dan semua karakternya harus ber-*case* sama (huruf kapital semua atau huruf kecil semua). Batasan ini juga berlaku untuk kata terang yang dipilih. Langkah pertama proses enkripsi *KCTR* adalah menghitung panjang teks pesan yang ingin didekripsi, misal bernilai *n*. Kemudian pilih sebuah kata terang dengan jumlah karakter tertentu, misal bernilai *p*, dengan $1<p<n$. Lalu beri nomor urut pada setiap karakter dalam kata terang tersebut. Nomor urut ini adalah angka dari satu hingga *p* tentunya. Penomoran ini didapatkan dari urutan tiap karakter dalam abjad alfabet (biasanya yang digunakan adalah urut abjad yang menaik). Jika terdapat karakter-karakter yang sama pada kata terang, maka boleh dipilih yang manapun dari karakter-karakter yang sama tersebut untuk diberi nomor lebih dulu, asal nomor urut tetap bertambah untuk karakter-karakter yang sama lainnya dan untuk karakter-karakter selanjutnya. Kemudian hitung dua buah bilangan bulat *x* dan *y* dari perhitungan $x = n \; div \; p$ dan $y = n \; mod \; p$. Selanjutnya buat sebuah matriks dengan jumlah kolom *p* dan jumlah baris $(x+1)$, tetapi bila nilai $y = 0$ maka jumlah baris adalah *x*. Berikutnya letakkan secara berurutan karakter-karakter teks pesan satu demi satu ke dalam elemen-elemen matriks, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah penempatan dari kiri ke kanan. Lalu seperti pada proses enkripsi *SCTR*, bila $y>0$ maka harus memilih antara menggunakan metode *irregular case* atau *regular case*. Setelah itu buat baris baru sebelum baris paling pertama matriks. Baris ini hanya baris semu yang berfungsi sebagai indeks tempat nomor urut milik kata terang berada. Lalu isi elemen-elemen matriks dalam baris semu tersebut dengan nomor urut yang telah dibuat untuk kata terang sesuai urutan karakter dalam kata terang. Kemudian transposisikan matriks, tetapi dengan aturan kolom

yang mendapatkan nomor urut (angka pada kolomnya dalam baris semu) terkecil (asumsi menggunakan urut abjad menaik), mendapat giliran transposisi paling awal, lalu dilanjutkan dengan kolom yang mendapat nomor urut setelah itu, begitu seterusnya hingga semua kolom ditransposisikan. Setelah matriks transposisi dibuat, maka chiperteks bisa didapatkan dengan mengambil karakter-karakter dalam elemen-elemen matriks secara berurutan satu demi satu, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah pengambilan dari kiri ke kanan. Akhirnya, selesailah proses enkripsi dengan algoritma *KCTR*. Contoh proses enkripsi dengan algoritma *KCTR* dapat dilihat pada bab hasil dan pembahasan.

### 2.4. Dekripsi *Keyed Columnar Transposition*

Proses dekripsi pada algoritma *KCTR* juga memiliki metode yang hampir sama dengan proses enkripsinya. Kata terang yang dipilih sebagai kunci simetri jelas mempengaruhi kekuatan enkripsi *KCTR*. Semakin banyak jumlah karakter dan karakter yang sama pada kata terang maka semakin sulit pula untuk kriptanalis dapat memecahkan enkripsi *KCTR* tersebut. Tetapi sayangnya, pemilihan kata terang ini juga dapat membuat proses dekripsi menjadi sulit pula. Ini disebabkan oleh proses dekripsi *KCTR* yang tidak selanjar seperti proses dekripsi yang dimiliki *SCTR*. Proses dekripsi *KCTR* dimulai dengan menghitung panjang chiperteks, misal bernilai *n*. Lalu hitung pula panjang kunci simetri atau kata terang, misal bernilai *p*. Kemudian hitung dua buah bilangan bulat *x* dan *y* dari perhitungan $x = n \; div \; p$ dan $y = n \; mod \; p$ dan buat matriks dengan jumlah kolom *x* dan jumlah baris *p*. Berikutnya sama seperti proses dekripsi *KCTR*, tentukan apakah enkripsi *KCTR* menggunakan metode *irregular case*, *regular case*, atau tidak keduanya. Bila metode yang didapatkan adalah *regular case* atau tidak keduanya maka selanjutnya juga sama yaitu letakkan secara berurutan karakter-karakter chiperteks satu demi satu ke dalam elemen-elemen matriks, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah penempatan dari kiri ke kanan. Lalu tambahkan satu kolom semu sebelum kolom pertama matriks untuk indeks nomor urutan karakter-karakter dalam kata terang. Lalu beri nomor urut pada setiap karakter dalam kata terang (caranya seperti pada proses enkripsi *KCTR*) dan masukkan ke dalam elemen-elemen matriks pada kolom semu, terurut per karakter kata terang ke dalam elemen matriks paling atas hingga elemen matriks paling bawah. Catatan, bila ada karakter-karakter dalam kata terang yang sama, tentu akan mempunyai dua atau lebih kombinasi urutan angka. Normalnya dekriptor hanya mengetahui kata terang sebagai kunci simetrinya, tetapi tidak dalam hal penomoran dan pengurutan karakter-karakter dalam kata terang. Hal inilah yang akan membuat proses dekripsi menjadi lebih sulit. Jumlah kombinasi ini akan menentukan berapa kali proses dekripsi yang dapat dijalankan maksimal. Kasus terbaik adalah satu kali proses, yaitu dalam sekali percobaan dengan suatu urutan angka ternyata urutan angka tersebut adalah urutan angka yang benar untuk karakter-karakter dalam kata terang tersebut. Dan kasus terburuk tentu saja adalah jumlah kombinasi maksimal yang mungkin terjadi tadi. Pada setiap kombinasi urutan angka untuk karakter-karakter dalam kata terang yang dicoba, langkah yang harus dilakukan adalah mengatur kembali matriks yaitu melakukan pertukaran antar baris agar mendapatkan urutan baris yang benar. Urutan baris ini didasarkan pada urutan angka pada indeks dalam kolom semu. Cara pengaturannya adalah angka pada elemen matriks kolom semu baris pertama menunjukkan nomor baris matriks saat itu yang akan menjadi baris pertama matriks baru, angka pada elemen matriks kolom semu baris kedua menunjukkan nomor baris matriks saat itu yang akan menjadi baris kedua matriks baru, begitu seterusnya hingga semua baris telah diatur sedemikian rupa berdasarkan angka pada kolom semu. Kemudian barulah matriks ditransposisikan seperti biasa. Lalu teks pesan asli atau plainteks bisa didapatkan dengan mengambil karakter-karakter dalam elemen-elemen matriks secara berurutan satu demi satu, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah pengambilan dari kiri ke kanan. Bila ternyata plainteks masih berbentuk cipherteks, berarti urutan angka untuk karakter-karakter dalam kata terang salah, dan dekriptor harus mencoba lagi dengan kombinasi urutan angka yang lain, mengulangi pengaturan baris matriks berdasarkan urutan angka tersebut, mentransposisikan matriks lagi, begitu seterusnya hingga didapatkan plainteks yang sebenarnya. Lain halnya dengan *irregular case*, langkah awal adalah membuat kolom baru setelah kolom terakhir matriks dengan jumlah elemen matriks *y* saja. Kemudian dekriptor juga harus memberikan nomor urut pada karakter-karakter dalam kata terang, yang bila ada karakter-karakter yang sama tentu urutan angka menjadi lebih dari satu kombinasi. Tiap urutan angka yang dicoba berfungsi untuk pengaturan baris pada matriks dan jumlah kolom pada tiap baris tersebut. Langkah awal adalah memasukkan urutan angka karakter-karakter dalam kata terang ke dalam elemen matriks kolom semu. Tetapi setelah itu ada satu masalah disini. Karena jumlah kolom pada tiap baris tidak semuanya sama, hal ini menyebabkan dekriptor tidak bisa begitu saja langsung memasukkan semua karakter chiperteks ke dalam elemen-elemen matriks, karena dekriptor tidak tahu baris mana saja yang mempunyai jumlah kolom lebih banyak atau lebih sedikit. Hal ini dapat diketahui dengan suatu cara. Caranya adalah cari angka indeks satu ada pada baris mana dalam kolom semu, misal angka indeks satu ada pada baris *q* kolom semu. Kemudian bandingkan *q* dengan *y*, bila $q > y$ maka baris pertama matriks mempunyai *x* kolom, dan bila $q \leq y$ maka baris pertama mempunyai $(x+1)$ kolom, lalu cari

angka indeks dua ada pada baris mana dalam kolom semu, kemudian bandingkan kembali dengan *y*, begitu seterusnya hingga semua baris telah mendapatkan jumlah kolom yang tepat. Berikutnya barulah memasukkan karakter-karakter chiperteks ke dalam elemen-elemen matriks secara berurutan satu demi satu, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah penempatan dari kiri ke kanan. Selanjutnya tukar baris matriks seperti cara yang dilakukan pada metode *regular case* atau non-metode, yaitu angka pada elemen matriks kolom semu baris pertama menunjukkan nomor baris matriks saat itu yang akan menjadi baris pertama matriks baru, angka pada elemen matriks kolom semu baris kedua menunjukkan nomor baris matriks saat itu yang akan menjadi baris kedua matriks baru, begitu seterusnya hingga semua baris telah diatur sedemikian rupa berdasarkan angka pada kolom semu. Kemudian matriks ditransposisikan seperti biasa. Lalu teks pesan asli atau plainteks bisa didapatkan dengan mengambil karakter-karakter dalam elemen-elemen matriks secara berurutan satu demi satu, dimulai dari elemen matriks paling kiri atas hingga elemen matriks paling kanan bawah dengan arah pengambilan dari kiri ke kanan. Bila ternyata plainteks masih berbentuk cipherteks, berarti urutan angka untuk karakter-karakter dalam kata terang salah, dan dekriptor harus mencoba lagi dengan kombinasi urutan angka yang lain, menentukan kembali jumlah kolom yang benar untuk tiap baris, mengulangi pengaturan baris matriks berdasarkan urutan angka tersebut, mentransposisikan matriks lagi, begitu seterusnya hingga didapatkan plainteks yang sebenarnya. Akhirnya, selesailah proses dekripsi dengan algoritma *KCTR*. Cukup rumit memang bila dibandingkan dengan enkripsinya atau dekripsi *SCTR*. Contoh proses dekripsi dengan algoritma *KCTR* dapat dilihat pada bab hasil dan pembahasan.

## 3. HASIL DAN PEMBAHASAN

### 3.1. Contoh Enkripsi dan Dekripsi *SCTR*

Diketahui plainteks adalah:
TUGASMAKALAHSTRUKDIS
maka enkripsinya adalah hitung *n*, *n = 20*, lalu pilih sebuah bilangan bulat *p*, dengan *1<p<20*. Bila tidak ingin menggunakan metode *irregular case* maupun *regular case*, tentu harus memilih bilangan yang memenuhi persamaan (*20 mod p = 0*) atau habis membagi *20* , yaitu *2*, *4*, *5*, atau *10*.

Bila tidak menggunakan metode, misal *p = 5*, maka buat matriks dengan (*20 div 5*) baris dan *p* kolom, lalu isi matriks dengan plainteks,

| | | | | |
|---|---|---|---|---|
| T | U | G | A | S |
| M | A | K | A | L |
| A | H | S | T | R |
| U | K | D | I | S |

maka chiperteks yang didapatkan setelah transposisi:
TMAUU AHKGK SDAAT ISLRS

Bila menggunakan *irregular case* dengan *p = 6*, berarti ada (*20 mod 6*) elemen matriks yang terisi pada baris terakhir,

| | | | | | |
|---|---|---|---|---|---|
| T | U | G | A | S | M |
| A | K | A | L | A | H |
| S | T | R | U | K | D |
| I | S | | | | |

maka chiperteks yang didapatkan setelah transposisi:
TASIU KTSGA RALUS AKMHD

Bila menggunakan *regular case* dengan *p = 6*, berarti ada (*6* - (*20 mod 6*)) karakter *dummy* dan misal karakter *dummy*/*pad* adalah XOXO,

| | | | | | |
|---|---|---|---|---|---|
| T | U | G | A | S | M |
| A | K | A | L | A | H |
| S | T | R | U | K | D |
| I | S | X | O | X | O |

maka chiperteks yang didapatkan setelah transposisi :
TASIU KTSGA RXALU OSAKX MHDO

Dekripsinya adalah bila panjang chiperteks *n*, buat matriks *p* (*p* adalah kunci simetri) X (*n div p*).

Bila chiperteks adalah:
TASIU KTSGA RXALU OSAKX MHDO (*n = 24*)
dan kunci simetri *6*, berarti tidak menggunakan metode atau *regular case* karena (*24 mod 6 = 0*)

| | | | |
|---|---|---|---|
| T | A | S | I |
| U | K | T | S |
| G | A | R | X |
| A | L | U | O |
| S | A | K | X |
| M | H | D | O |

maka plainteks didapatkan setelah transposisi:
TUGASMAKALAHSTRUKDISXOXO
(sebaiknya karakter *pad/dummy* adalah karakter yang tidak membentuk kata yang mempunyai arti agar dekriptor tidak salah mengartikan plainteks)

Bila chiperteks adalah:
TASIU KTSGA RALUS AKMHD (*n = 20*)
dan kunci simetri 6, maka menggunakan *irregular case* karena (*20 mod 6 > 0*) dan ada tambahan kolom baru dengan jumlah baris (*20 mod 6*).

| | | | |
|---|---|---|---|
| T | A | S | I |
| U | K | T | S |
| G | A | R | |
| A | L | U | |
| S | A | K | |
| M | H | D | |

maka plainteks yang didapatkan setelah transposisi:
TUGASMAKALAHSTRUKDIS

### 3.2. Contoh Enkripsi dan Dekripsi *KCTR*

Diketahui plainteks adalah:
ARITMATIKAMODULODANTRANSPOSISI
maka enkripsinya hitung $n$, $n = 30$, lalu pilih sebuah kata terang yang mempunyai panjang $p$, dengan $1<p<30$. Bila tidak ingin menggunakan metode *irregular case* maupun *regular case*, tentu harus memilih kata terang yang mempunyai panjang yang habis membagi *30*. Cara penggunaan metode sama, jadi di contoh ini akan lebih ditekankan ke arah urutan angka dalam karakter-karakter kata terang, dengan misal kata terang adalah DISKRIT, maka $p = 7$ dan menggunakan metode *irregular case*. Urutan angka kata terang yang terjadi:

| | | | | | | |
|---|---|---|---|---|---|---|
| D | I | S | K | R | I | T |
| 1 | 2 | 6 | 4 | 5 | 3 | 7 |

atau

| | | | | | | |
|---|---|---|---|---|---|---|
| D | I | S | K | R | I | T |
| 1 | 3 | 6 | 4 | 5 | 2 | 7 |

lalu buat matriks dengan jumlah baris ((*30 div 7*)+*1*) dan jumlah kolom $p$, dan baris semu yang berisi urutan angka yang didapatkan,

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 6 | 4 | 5 | 3 | 7 |
| A | R | I | T | M | A | T |
| I | K | A | M | O | D | U |
| L | O | D | A | N | T | R |
| A | N | S | P | O | S | I |
| S | I | | | | | |

maka chiperteks yang didapatkan setelah transposisi berdasarkan baris semu dengan urutan 1264537:
AILAS RKONI ADTST MAPMO NOIAD STURI

sedangkan bila berdasarkan baris semu 1364527 maka chiperteks yang didapatkan setelah transposisi:
AILAS ADTSR KONIT MAPMO NOIAD STURI

Dekripsinya, misal chiperteks adalah:
AILAS ADTSR KONIT MAPMO NOIAD STURI
dan kunci simetri adalah DISKRIT, setelah hitung panjang chiperteks ($n = 30$), maka buat matriks dengan panjang kunci simetri atau kata terang ($p = 7$) menjadi jumlah baris dan (*30 div 7*) menjadi jumlah kolom awal. Lalu buat urutan-urutan angka yang mungkin dari kata terang DISKRIT dan jadikan kolom semu (didapatkan urutan 1264537 dan 1364527).

Bila menggunakan urutan 1264537, cari jumlah kolom yang tepat pada tiap baris dengan cara:

- angka 1 berada pada baris 1 kolom semu, maka jumlah kolom baris 1 adalah ((*30 div 7*) + *1*) karena memenuhi persamaan ($2 \leq 30 \bmod 7$)
- angka 2 berada pada baris 2 kolom semu, maka jumlah kolom baris 2 adalah ((*30 div 7*) + *1*) karena memenuhi persamaan ($1 \leq 30 \bmod 7$)
- angka 3 berada pada baris 6 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan ($6 > 30 \bmod 7$)
- angka 4 berada pada baris 4 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan ($4 > 30 \bmod 7$)
- angka 5 berada pada baris 5 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan ($5 > 30 \bmod 7$)
- angka 6 berada pada baris 3 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan ($3 > 30 \bmod 7$)
- angka 7 berada pada baris 7 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan ($7 > 30 \bmod 7$)

lalu isi dengan chiperteks, maka matriks adalah:

| | | | | | |
|---|---|---|---|---|---|
| 1 | A | I | L | A | S |
| 2 | A | D | T | S | R |
| 6 | K | O | N | I | |
| 4 | T | M | A | P | |
| 5 | M | O | N | O | |
| 3 | I | A | D | S | |
| 7 | T | U | R | I | |

lalu atur baris sebenarnya dengan cara:

- baris 1 kolom semu menunjukkan angka 1, artinya baris 1 sekarang akan menjadi baris 1 matriks baru
- baris 2 kolom semu menunjukkan angka 2, artinya baris 2 sekarang akan menjadi baris 2 matriks baru
- baris 3 kolom semu menunjukkan angka 6, artinya baris 6 sekarang akan menjadi baris 3 matriks baru
- baris 4 kolom semu menunjukkan angka 4, artinya baris 4 sekarang akan menjadi baris 4 matriks baru
- baris 5 kolom semu menunjukkan angka 5, artinya baris 5 sekarang akan menjadi baris 5 matriks baru
- baris 6 kolom semu menunjukkan angka 3, artinya baris 3 sekarang akan menjadi baris 6 matriks baru
- baris 7 kolom semu menunjukkan angka 7, artinya baris 7 sekarang akan menjadi baris 7 matriks baru

maka matriks baru yang didapatkan:

| | | | | |
|---|---|---|---|---|
| A | I | L | A | S |
| A | D | T | S | R |
| I | A | D | S | |
| T | M | A | P | |
| M | O | N | O | |
| K | O | N | I | |
| T | U | R | I | |

maka plainteks yang didapatkan setelah transposisi:
AAITMKTIDAMOOULTDANNRASSPOIISR

Ternyata plainteks masih berbentuk chiperteks (karena teks tidak jelas dan tidak dapat dimengerti), maka disimpulkan urutan angka salah. Lalu lanjutkan menggunakan kombinasi urutan angka yang lain, yaitu 1364527, maka dengan memakai cara yang sama untuk menentukan jumlah kolom yang benar untuk tiap baris pada matriks:

- angka 1 berada pada baris 1 kolom semu, maka jumlah kolom baris 1 adalah ((*30 div 7*) + *1*) karena memenuhi persamaan (*2* ≤ *30 mod 7*)
- angka 2 berada pada baris 6 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan (*6 > 30 mod 7*)
- angka 3 berada pada baris 2 kolom semu, maka jumlah kolom baris 3 adalah ((*30 div 7*) + *1*) karena memenuhi persamaan (*1* ≤ *30 mod 7*)
- angka 4 berada pada baris 4 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan (*4 > 30 mod 7*)
- angka 5 berada pada baris 5 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan (*5 > 30 mod 7*)
- angka 6 berada pada baris 3 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan (*3 > 30 mod 7*)
- angka 7 berada pada baris 7 kolom semu, maka jumlah kolom baris 2 adalah (*30 div 7*) karena memenuhi persamaan (*7 > 30 mod 7*)

lalu isi dengan chiperteks sehingga matriks adalah:

| | | | | | |
|---|---|---|---|---|---|
| 1 | A | I | L | A | S |
| 3 | A | D | T | S | |
| 6 | R | K | O | N | I |
| 4 | T | M | A | P | |
| 5 | M | O | N | O | |
| 2 | I | A | D | S | |
| 7 | T | U | R | I | |

lalu dengan cara yang sama pula untuk menentukan posisi baris yang benar pada matriks:

- baris 1 kolom semu menunjukkan angka 1, artinya baris 1 sekarang akan menjadi baris 1 matriks baru
- baris 2 kolom semu menunjukkan angka 3, artinya baris 3 sekarang akan menjadi baris 2 matriks baru
- baris 3 kolom semu menunjukkan angka 6, artinya baris 6 sekarang akan menjadi baris 3 matriks baru
- baris 4 kolom semu menunjukkan angka 4, artinya baris 4 sekarang akan menjadi baris 4 matriks baru
- baris 5 kolom semu menunjukkan angka 5, artinya baris 5 sekarang akan menjadi baris 5 matriks baru
- baris 6 kolom semu menunjukkan angka 2, artinya baris 2 sekarang akan menjadi baris 6 matriks baru
- baris 7 kolom semu menunjukkan angka 7, artinya baris 7 sekarang akan menjadi baris 7 matriks baru

maka matriks baru yang didapatkan:

| | | | | |
|---|---|---|---|---|
| A | I | L | A | S |
| R | K | O | N | I |
| I | A | D | S | |
| T | M | A | P | |
| M | O | N | O | |
| A | D | T | S | |
| T | U | R | I | |

maka plainteks yang didapatkan setelah transposisi:
ARITMATIKAMODULODANTRANSPOSISI

Plainteks benar karena teks jelas dan dapat dimengerti. Bila semua kombinasi urutan angka telah dicoba dan tidak menghasilkan plainteks yang benar, maka kemungkinan terdapat kesalahan ketika menentukan jumlah kolom yang tepat pada tiap baris atau ketika mengatur baris yang benar.

## 4. KESIMPULAN

1. Konsep aritmatika modulo dan transposisi matriks ternyata dapat diterapkan untuk algoritma kriptografi
2. Algoritma kriptografi *columnar transposition* adalah algoritma yang menerapkan konsep aritmatika modulo dan transposisi matriks
3. Kekuatan algoritma kriptografi *columnar transposition* terletak pada panjang plainteks dan ketepatan kunci simetri yang dipilih
4. Ada dua jenis algoritma kriptografi *columnar transposition* yaitu *SCTR* (*Simple Columnar Transposition*) dan *KCTR* (*Keyed Columnar Transposition*)
5. Perbedaan antara *SCTR* dan *KCTR* terletak pada bentuk kunci simetri yang digunakan dan urutan transposisi matriksnya.
6. Algoritma *SCTR* menggunakan bilangan bulat sebagai kunci simetrinya dan cara transposisi yang lanjar, sedangkan *KCTR* menggunakan kata terang sebagai kunci simetrinya dan cara transposisi yang didasarkan dari urutan karakter-karakter dalam kata terang tersebut

## DAFTAR REFERENSI


[1] Munir, Rinaldi, *Diktat Kuliah IF2151 Matematika Diskrit Edisi Keempat*, Departemen Teknik Informatika Institut Teknologi Bandung, 2004

[2] http://www.contestcen.com/columnar.htm, tanggal akses: 25 Desember 2008 pukul 05.30

[3] http://rif13crypt.wordpress.com/2007/12/08/kriptografi-clasik, tanggal akses: 25 Desember 2008 pukul 05.40

[4] http://en.wikipedia.org/wiki/Transposition_cipher.htm, tanggal akses: 25 Desember 2008 pukul 05.50

[5] http://crypto.dsplabs.com.au/classical/columnar-transposition-cipher-cryptanalysis.php, tanggal akses: 25 Desember 2008 pukul 06.00