Kriptografi dalam Kemanan Bertransaksi

Albhikautsar Dharma Kesuma (13511058)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511058@std.stei.itb.ac.id

***Abstract*—Kriptografi adalah salah satu cabang dari ilmu teori bilangan yang sudah banyak digunakan saat ini. Ilmu kriptografi sudah banyak dikembangkan dalam kehidupan sehari-hari. Salah satu applikasi kriptografi adalah kegunaannya dalam keamanan dalam melakukan pembayaran *online*. Sistem pembayaran *online* sudah banyak dilakukan dan tingkat kepercayaan penggunanya sudah tinggi akan sistem pembayaran ini. Pada makalah ini akan dijelaskan mengenai keamanan sistem pembayaran *online* dalam tingkat yang belum terlalu dalam dan masih dapat dipelajari.**

***Index Terms*—Teori Bilangan, Kriptografi, Internet Banking, Secure Hash Algorithm.**

I. PENDAHULUAN

Tanpa disadari, pengaplikasian ilmu teori bilangan sudah kita gunakan setiap hari. Ilmu teori bilangan sudah mendasari banyak ilmu-ilmu yang sangat berguna bagi kita sekarang. Kriptografi adalah salah satu ilmu yang didasari oleh ilmu teori bilangan. Kriptografi sudah ditemukan sekitar tahun 400 SM, dan sekarang ilmu ini sudah sangat berkembang sebagai contohnya adalah dalam keamanan sebuah jaringan.

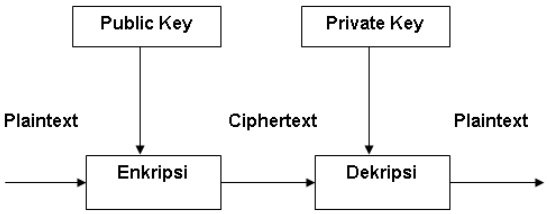
Kriptografi akan dijelaskan lebih mendalam pada makalah ini dalam pembahasannya mengenai keamanan dalam sistem perbankan yang sudah sering dilakukan dalam kehidupan sehari-hari ini.

II. TEORI BILANGAN

Teori bilangan adalah salah satu bidang matematika murni yang mempelajari bilangan bulat. Teori bilangan menjelaskan pula tentang bilangan prima dan faktorisasi dari bilangan prima itu sendiri. Teori bilangan juga biasa disebut dengan aritmatik tingkat tinggi.

A. Kriptografi

Kriptografi adalah salah satu ilmu yang digunakan untuk mengamankan sebuah pesan yang dilakukan oleh seseorang yang biasa disebut dengan kriptografer. Pesan yang sudah diamankan tersebut dapat diartikan atau dipahami kembali dengan suatu ilmu yang disebut dengan cryptanalisis, dan orang yang melakukan pendekodean ini biasa disebut dengan kriptanalisis.



Seperti yang sudah dijelaskan pada diagram diatas, ada dua buah pesan atau text yang digunakan dalam ilmu kriptografi dan kriptoanalisis, yaitu *plain text* dan *chipper text*. *Plain text* adalah sebuah pesan asli yang dapat dibaca oleh siapapun atau bisa disebut dengan pesan yang tidak dirahasiakan, atau pesan yang belum di enkripsi, atau belum di dekripsi. Sedangkan *chipper text* adalah pesan yang sudah di enkripsi dan belum di dekripsi, sehingga pesan tersebut tidak bisa dibaca atau sudah dirahasiakan. Proses enkripsi-dekripsi ini disebut dengan kriptosistem. Dalam proses penchipperan ini digunakan satu atau lebih kunci kriptografi berdasarkan apa jenis kriptografi apa yang digunakan.

Ada dua macam kriptosistem, yaitu kriptosistem simetrik dan kriptosistem asimetrik. Kriptosistem simetrik adalah sebuah proses penchipperan yang menggunakan kunci yang identic dalam melakukan penchipperan. Kunci pada kriptosistem simetrik dapat diturunkan dari kunci yang lainnya. Contoh dari kriptosistem simetrik adalah DES (Data Encryption Standard), Blowfish, dan IDEA.

Kriptosistem asimetrik adalah proses penchipperan yang menggunakan dua buah kunci, yaitu kunci public (kunci yang dapat dipublikasikan) dan kunci privat (kunci yang harus dirahasiakan). Salah satu contoh penggunaan kriptosistem asimetrik adalah untuk suatu proses pengiriman yang dapat diketahui asal mula surat itu, dan hanya penerima yang dapat membaca pesan yang dikiriman oleh pengirim dengan menggunakan kunci publik dan kunci privat yang dimiliki masing-masing pengirim dan penerima. Contoh dari kriptosistem asimetrik adalah RSA Scheme dan Merkle-Hellman Scheme

1. Notasi Matematis

Jika dimisalkan C adalah chipper text dan dan P adalah plain text, maka fungsi enkrips E memetakan P ke C adalah

E(P) = C

Sedangkan pada proses dekripsi D memetakan C pada P adalah

D(C) = P

atau bisa diartikan bahwa fungsi D adalah invers dari fungsi E. Sebuah fungsi invers dapat terjadi jika dan hanya jika masing-masing elemen berkoresponden satu-satu. Dengan kata lain, sebuah chipper text hanya bisa diubah menjadi bentuk plain text semulanya tanpa ada perubahan didalamnya sedikitpun.

2. Metode Kriptografi

Teknik Dasar Kriptografi Kuno

A. Substitusi

Teknik substitusi biasa disebut dengan Caesar chipper. Pada proses enkripsi dan dekripsi dari pesan ini menggunakan tabel yang bisa dibuat sendiri, dengan tabel acak yang dibentuk, akan semakin aman dan tidak semua orang dapat mendekripsi pesan yang dikirimin. Berikut adalah contoh tabel yang digunakan untuk mendekripsikan sebuah pesan

a	b	c	d	e	f	g	h	i	j	k	l	n	n	o	p	q	r	s	t	u	v	w	x	y	z	0	1	2	3	4	5	6	7	8	9	.	,
q	r	b	l	k	x	d	a	t	y	c	s	m	7	u	v	e	6	f	w	h	j	g	n	.	i	5	2	o	4	3	,	p	8	z	0	9	1

Contoh pesan :

aqsu 7qmq fq.q qsrat9

yang artinya adalah

halo nama saya albhi.

B. Bloking

Bloking adalah sebuah teknik dalam kriptografi yang membagi sebuah pesan yang akan dikirim menjadi beberapa bagian blok. Penulisan pesan dapat dilakukan horizontal lalu dibaca secara vertical, atau dapat dilakukan dengan ditulis secara vertical dan dibaca secara horizontal. Berikut adalah contoh dari teknik Bloking.

h	m	
a	a	a
l		l
o	s	b
	a	h
n	y	i
a	a	.

Contoh Pesan :

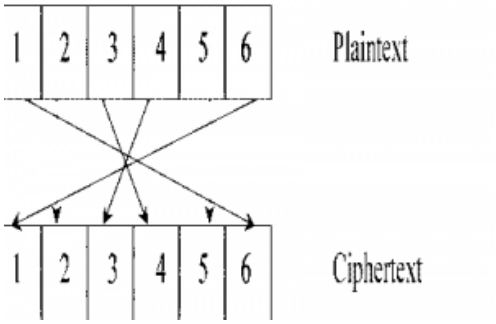
hm aaal l osb ah nyiaa.

Yang artinya adalah :

Halo nama saya albhi.

C. Permutasi

Teknik permutasi adalah salah satu teknik enkripsi yang penting. Teknik permutasi ini pada dasarnya merubah posisi dari tiap karakter sesuai dengan urutannya. Teknik ini berbeda dengan teknik pada saat penenkripsian sebuah pesan yang sudah diberikan tabelnya. Berikut adalah contoh dari teknik permutasi.



Teknik Dasar Kriptografi Modern

1. Data Encryption Standard (DES)

Data Encryption Standard adalah suatu cara enkripsi yang menggunakan kunci privat. Sama seperti pada proses penchipperan, baik pengirim maupun penerima harus mengetahui kunci untuk membaca pesan tersebut. Kunci yang didapat, diperoleh dari pengambilan angka secara random dalam jumlah kunci yang sangat banyak.

2. RSA Encryption

Rivert-Shamir-Adleman adalah peneliti yang berhasil membuat proses enkripsi yang didasari oleh konsep bilangan prima dan modulo. Kunci dari enkripsi dan deksripsi menggunakan bilangan bulat. Kunci enkripsi yang digunakan adalah kunci public, namun kunci dekripsi menggunakan kunci privat. Kunci dekripsi didapatkan oleh penerima dengan memfaktorkan bilangan non prima menjadi bilangan prima. Semakin sulit pemfaktoran ini, semakin aman pula mengamanan dalam surat ini

3. Secure Hash Algorithm (SHA)

Secure Hash Algorithm adalah salah satu pengamanan surat yang tidak dapat merubah chippertext menjadi plain text kembali. Penjelasan mengenai SHA akan dijelaskan lebih mendalam pada makalah ini.

4. Security Token

Security Token adalah pengamanan modern yang sudah biasa digunakan dalam bertransaksi dalam

dunia perbankan. Penjelasan mengenai token akan dijelaskan lebih lanjut pad makalah ini.

3. *Fungsi Hash*

Fungsi Hash berguna untuk menempatkan suatu record yang mempunyai nilai kunci k, hal ini berfunsi untuk penyimpanan data dalam suatu memori komputer yang ditempatkan sedemikian rupa sehingga pencarian dapat dilakukan dengan cepar. Setiap data memiliki *field* atau tempat yang unik sehingga dapat dibedakan antara tiap penyimpanan dari data tersebut. Fungsi Hash yang paling umum berbentuk :

$$h(k) = k \textbf{ mod } m$$

4. *Keamanan Jaringan*

Keamanan jaringan adalah salah satu permasalahan yang penting belakangan ini. Berkembanganya ilmu komputer telah memberikan kita banyak manfaat, namun tidak dapat dipungkiri lagi bahwa perkembangan ini dari ilmu komputer ini pula dapat memberikan kita kemampuan untuk meretas atau mengganggu suatu jaringan komputer. Karena itu pula keamanan jaringan sudah sepatutnya turut meningkat dengan berkembangnya ilmu komputer.

Belakangan ini, informasi sudah menjadi hal yang penting dalam kehidupan berorganisasi, setiap anggotanya ingin dapat mengakses informasi tersebut. Pada makalah ini akan dibahas mengenai keamanan sebuah jaringan yang diimplementasikan pada dunia perbankan, lebih spesifiknya pada internet/online banking.

Sebuah bank harus dapat menjaga validitas dan integritas informasi dari setiap nasabahnya, selain itu pula bank harus mampu mencegah pengaksesan data dari pihak yang tidak berwenang.

III. PERMASALAHAN DALAM KRIPTOSISTEM DAN KEAMANAN JARINGAN

A. Jenis Penyerangan pada Protokol

1. Chippertext-only attack
   Jenis penyerangan ini adalah seorang kriptonalisis memiliki chippertext dari sejumlah pesan yang sudah di enkripsi dengan menggunakan alogoritma yang sama.

2. Known-plaintext attack
   Pada penyerangan ini, kriptonalisis bukan hanya memiliki chippertext dari sejumlah pesan, namun memiliki plaintextnya juga.

3. Chosen-plaintext attack
   Sama hal-nya pada *known-plaintext attack,* seoarng kriptonalisis memiliki sejumlah chippertext dan plaintext, namun ia bisa juga memilih pesan mana yang akan di enkripsi

4. Adaptive-chosen-plaintext attack
   Penyerangan tipe ini adalah suatu kasuss khusus dari *chosen-plaintext attack*. Kriptanalis dapat memiliki kemampuan untuk merubah pilihan berdasarkan hasil enkripsi sebelumnya. Pada *adaptive-chosen-plaintext attack* kriptonalis dapat memilih blok dari plain text lalu memilih blok lain dengan menggunakan referensi yang sama, maka kriptonalis mampu memiliki keseluruan data.

5. Chosen-chippertext attack
   Tipe penyerangan ini memberikan kriptonalis memilih chippertext mana yang akan didekripsi dan memiliki akses penuh terhadap plaintext.

6. Chosen-key attack
   Pada penyerangan ini, kriptonalis memiliki kemampuan tentang hubungan dari kunci-kunci yang berbeda.

7. Rubber-hose cryptanalysis
   Pada penyerangan ini, kriptonalis bertindak yang tidak sepantasnya, seperti mengancam dan atau memaksa seseorang hingga mereka memberikan kunci kepada kriptonalis tersebut.

B. Jenis Penyerangan pada Jaringan

1. Sniffing
   Sniffing dalam bahasa Indonesia adalah mengendus, hal ini bukan lagi hal yang jarang terjadi, sang pengendus dapat mengganggu suati jarinagan komunikasi dan mengambil pesan baik yang belum ataupun sudah di enkripsi. Sang pengundus pula dapat merekam semua komunikasi.

2. Replay attack
   Replay attack adalah suatu penyerangan yang merekam sebuah komunikasi dan memutar ulangnya untuk mengelabui pihak lain.

3. Spoofing
   Penyerangan tipe ini adalah sebuah pengelabuan satu atau berbagai pihak untuk memercaya bahwa A itu adalah B. Dengan berikut, pihak lain tidak akan mengetahu bahwa B bukanlah B yang sesungguhnya. Hal ini bisa terjadi pada saat konfirmasi pada saat penggunaan telepon genggam dan menguhubungi *customer service.*

4. Man in the Middle
   Pada dasarnya penyerangan ini sama dengan *Spoofing,* biasanya *Spoofing* hanya mengelabui satu pihak, namun pada *Man in the Middle*, seseorang dapat mengelabui berbagai pihak

dengan mengaku sebagai berbagai orang yang berbeda.

IV. PENCEGAHAN PERETASAN KEAMANAN JARINGAN

Dalam dunia perbankan, sudah banyak alternative yang digunakan untuk keamanan bertransaksi, Penipuan melalui ATM sudah banyak menelan korban. Pada kenyataannya, *Spoofing,* atau penggunaan kartu ATM yang bisa digunakan oleh pihak lain apabila seseorang itu memiliki kartu ATM dan memiliki PIN untuk memverifikasi sehigga dapat melakukan transaksi. Untuk meningkatkan keamanan dan kenyamanan dalam bertransaksi, beberapa bank di Indonesia sudah memberikan opsi *internet/online banking.*

Untuk memverifikasi transaksi pada *internet/online banking,* bank biasanya menggunakan sebuah token yang akan memberikan digit yang diberikan sesuai dengan algoritma yang digunakan.

Ada bermacam-macam jenis Token yang digunakan untuk bertransaksi.

- Terputus Token
  Terputus token tidak memiliki koneksi fisik maupun logis ke komputer klien dan tidak menggunakan perangkat khusus, Terputus token menggunakan built-in layar untuk membaca masukan dari pengguna dan memverifikasinya. Token jenis inilah yang umum digunakan dalam dua faktor otentikasi untuk identifikasin online.
- Connected Token
  Token jenis ini membutuhkan koneksi fisik langsung ke komputer. Info akan secara otomatis di identifikasi dengan server.
- Contactless Token
  Contactless token memberikan kita kemudahan yang tidak membutuhkan koneksi fisik ke komputer namun dapat menggunakan koneksi secara logis dengan komputer sehingga komputer dapat mengartikannya. Token jenis ini sangat popular dalam kehidupan sehari-hari ini. Hal ini memudahkan kita untuk melakukan pembayaran online dengan cepat dan aman.
- Telepon Selular GSM
  Token tipe ini menggunakan aplikasi java yang di install pada ponsel, metode yang digunakan bisa berupa dalam bentuk pesan SMS maupun menggunakan protokol internet standar seperti HTTP atau HTTPS.

SHA (Secure Hash Algorithm)

Secure Hash Algorithma atau SHA adalah satu cabang ilmu pengamanan pesan yang tidak dapat di dekripsi kembali. Fungsi Hash juga akan men-*generate* berapa besarpun text menjadi sebuah ukuran yang pasti. Dengan begitu, penyimpanan data akan sangat aman karena tidak akan ada pihak lain yang akan mendapatkan pesan atau informasi asli dari setiap pelanggannya. Hal ini akan sangat cocok untuk digunkan dalah dunia perbankan.

SHA 1

SHA 1 adalah bentuk dari Secure Hash Algorithm yang paling aman. SHA 1 digunakan dalam SSL (Secure Sockets Level), PGP (Pretty Good Privacy), dan XML Signatures. SHA 1 juga sudah didefiniskan pada NIST atau National Institute of Standard and Technology sebagai standard 'FIPS 180-2'. NIST juga memberikan *test vector* untuk memverifikasi perbaikan dari implementasi.

Berikut adalah salah satu implementasi SHA-1 dalam bahasa java yang disadur dari http://www.movable-type.co.uk/scripts/sha1.html :

```
/* - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - -
-  */
/*  SHA-1 implementation in JavaScript | (c)
Chris Veness 2002-2010 | www.movable-type.co.uk
*/
/*   - see
http://csrc.nist.gov/groups/ST/toolkit/secure_h
ashing.html                                   */
/*
http://csrc.nist.gov/groups/ST/toolkit/examples
.html                                         */
/* - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - -
-  */

var Sha1 = {};  // Sha1 namespace

/**
 * Generates SHA-1 hash of string
 *
 * @param {String} msg                  String to
be hashed
 * @param {Boolean} [utf8encode=true] Encode
msg as UTF-8 before generating hash
 * @returns {String}                    Hash of
msg as hex character string
 */
Sha1.hash = function(msg, utf8encode) {
  utf8encode =  (typeof utf8encode ==
'undefined') ? true : utf8encode;

  // convert string to UTF-8, as SHA only deals
with byte-streams
  if (utf8encode) msg = Utf8.encode(msg);

  // constants [§4.2.1]
  var K = [0x5a827999, 0x6ed9eba1, 0x8f1bbcdc,
0xca62c1d6];

  // PREPROCESSING

  msg += String.fromCharCode(0x80);  // add
trailing '1' bit (+ 0's padding) to string
[§5.1.1]

  // convert string msg into 512-bit/16-integer
blocks arrays of ints [§5.2.1]
  var l = msg.length/4 + 2;  // length (in 32-
bit integers) of msg + '1' + appended length
  var N = Math.ceil(l/16);   // number of 16-
integer-blocks required to hold 'l' ints
  var M = new Array(N);

  for (var i=0; i<N; i++) {
    M[i] = new Array(16);
    for (var j=0; j<16; j++) {  // encode 4
chars per integer, big-endian encoding
      M[i][j] = (msg.charCodeAt(i*64+j*4)<<24)
| (msg.charCodeAt(i*64+j*4+1)<<16) |
        (msg.charCodeAt(i*64+j*4+2)<<8) |
(msg.charCodeAt(i*64+j*4+3));
    } // note running off the end of msg is ok
'cos bitwise ops on NaN return 0
```

```
  }
  // add length (in bits) into final pair of
32-bit integers (big-endian) [§5.1.1]
  // note: most significant word would be (len-
1)*8 >>> 32, but since JS converts
  // bitwise-op args to 32 bits, we need to
simulate this by arithmetic operators
  M[N-1][14] = ((msg.length-1)*8) / Math.pow(2,
32); M[N-1][14] = Math.floor(M[N-1][14])
  M[N-1][15] = ((msg.length-1)*8) & 0xffffffff;

  // set initial hash value [§5.3.1]
  var H0 = 0x67452301;
  var H1 = 0xefcdab89;
  var H2 = 0x98badcfe;
  var H3 = 0x10325476;
  var H4 = 0xc3d2e1f0;

  // HASH COMPUTATION [§6.1.2]

  var W = new Array(80); var a, b, c, d, e;
  for (var i=0; i<N; i++) {

    // 1 - prepare message schedule 'W'
    for (var t=0;  t<16; t++) W[t] = M[i][t];
    for (var t=16; t<80; t++) W[t] =
Sha1.ROTL(W[t-3] ^ W[t-8] ^ W[t-14] ^ W[t-16],
1);

    // 2 - initialise five working variables a,
b, c, d, e with previous hash value
    a = H0; b = H1; c = H2; d = H3; e = H4;

    // 3 - main loop
    for (var t=0; t<80; t++) {
      var s = Math.floor(t/20); // seq for
blocks of 'f' functions and 'K' constants
      var T = (Sha1.ROTL(a,5) + Sha1.f(s,b,c,d)
+ e + K[s] + W[t]) & 0xffffffff;
      e = d;
      d = c;
      c = Sha1.ROTL(b, 30);
      b = a;
      a = T;
    }

    // 4 - compute the new intermediate hash
value
    H0 = (H0+a) & 0xffffffff;  // note
'addition modulo 2^32'
    H1 = (H1+b) & 0xffffffff;
    H2 = (H2+c) & 0xffffffff;
    H3 = (H3+d) & 0xffffffff;
    H4 = (H4+e) & 0xffffffff;
  }

  return Sha1.toHexStr(H0) + Sha1.toHexStr(H1)
+
    Sha1.toHexStr(H2) + Sha1.toHexStr(H3) +
Sha1.toHexStr(H4);
}

//
// function 'f' [§4.1.1]
//
Sha1.f = function(s, x, y, z)  {
  switch (s) {
  case 0: return (x & y) ^ (~x & z);
// Ch()
  case 1: return x ^ y ^ z;
// Parity()
  case 2: return (x & y) ^ (x & z) ^ (y & z);
// Maj()
  case 3: return x ^ y ^ z;
// Parity()
  }
}

//
// rotate left (circular left shift) value x by
n positions [§3.2.5]
//
Sha1.ROTL = function(x, n) {
  return (x<<n) | (x>>>(32-n));
}

//
// hexadecimal representation of a number
//   (note toString(16) is implementation-
dependant, and
//   in IE returns signed numbers when used on
full words)
//
Sha1.toHexStr = function(n) {
  var s="", v;
  for (var i=7; i>=0; i--) { v = (n>>>(i*4)) &
0xf; s += v.toString(16); }
  return s;
}


/* - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - -
-  */
/*  Utf8 class: encode / decode between multi-
byte Unicode characters and UTF-8 multiple
*/
/*              single-byte character encoding
(c) Chris Veness 2002-2010
*/
/* - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - -
-  */

var Utf8 = {};  // Utf8 namespace

/**
 * Encode multi-byte Unicode string into utf-8
multiple single-byte characters
 * (BMP / basic multilingual plane only)
 *
 * Chars in range U+0080 - U+07FF are encoded
in 2 chars, U+0800 - U+FFFF in 3 chars
 *
 * @param {String} strUni Unicode string to be
encoded as UTF-8
 * @returns {String} encoded string
 */
Utf8.encode = function(strUni) {
  // use regular expressions & String.replace
callback function for better efficiency
  // than procedural approaches
  var strUtf = strUni.replace(
      /[\u0080-\u07ff]/g,  // U+0080 - U+07FF
=> 2 bytes 110yyyyy, 10zzzzzz
      function(c) {
        var cc = c.charCodeAt(0);
        return String.fromCharCode(0xc0 |
cc>>6, 0x80 | cc&0x3f); }
    );
  strUtf = strUtf.replace(
      /[\u0800-\uffff]/g,  // U+0800 - U+FFFF
=> 3 bytes 1110xxxx, 10yyyyyy, 10zzzzzz
      function(c) {
        var cc = c.charCodeAt(0);
        return String.fromCharCode(0xe0 |
cc>>12, 0x80 | cc>>6&0x3F, 0x80 | cc&0x3f); }
    );
  return strUtf;
}

/**
 * Decode utf-8 encoded string back into multi-
byte Unicode characters
 *
 * @param {String} strUtf UTF-8 string to be
decoded back to Unicode
 * @returns {String} decoded string
 */
Utf8.decode = function(strUtf) {
  // note: decode 3-byte chars first as decoded
2-byte strings could appear to be 3-byte char!
  var strUni = strUtf.replace(
      /[\u00e0-\u00ef][\u0080-
\u00bf][\u0080-\u00bf]/g,  // 3-byte chars
      function(c) {  // (note parentheses for
precence)
        var cc = ((c.charCodeAt(0)&0x0f)<<12) |
((c.charCodeAt(1)&0x3f)<<6) | (
c.charCodeAt(2)&0x3f);
        return String.fromCharCode(cc); }
    );
  strUni = strUni.replace(
      /[\u00c0-\u00df][\u0080-\u00bf]/g,
// 2-byte chars
      function(c) {  // (note parentheses for
```

```
precence)
        var cc = (c.charCodeAt(0)&0x1f)<<6 |
c.charCodeAt(1)&0x3f;
        return String.fromCharCode(cc); }
    );
  return strUni;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  */
```

V. KESIMPULAN

Dalam perkembangan dunia teknologi, kemanan jaringan sudah menjadi hal yang sangat penting, contohnya adalah dalam dunia perbankan dan dalam *internet banking*. Ada beberapa alternative dalam menjaga keamanan bertransaksi, yaitu menggunakan token. Algotritma yang aman digunakan dalam dunia perbankan adalah SHA 1, karena memiliki tingkat kemanan yang paling tinggi.

REFERENCES

Munir, Rinaldi. *Matermatika Diskrit Revisi Kelima.* Bandung : Palasari 2012.

Number Theory, http://mathworld.wolfram.com/NumberTheory.html diakses pada tanggal 14 Desember 2012 pukul 19.02 WIB

Security Token, http://visilubai.wordpress.com/2010/05/07/security-token/ diakses pada tanggal 15 Desmber 2012, pukul 13.25 WIB

Kriptografi Dalam Konteks Keamanan Komputer dan Jaringan, http://panjinovantara.staffsite.uniku.ac.id/2012/05/09/kriptografi-dalam-konteks-keamanan-komputer-dan-jaringan/ diakses pada tanggal 15 Desember 2012, pukul 14.00 WIB

SHA-1 Cryptographic Hash Algorith http://www.movable-type.co.uk/scripts/sha1.html diakses pada tanggal 16 Desember 2012, pukul 20.08 WIB

Keamanan Jaringan, http://lecturer.ukdw.ac.id/cnuq/wp-content/uploads/keamananjaringan.pdf, diakses pada tangal 16 Desember 2012, pukul 20.45 WIB

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, 16 Desember 2012

Albhikautsar Dharma Kesuma (13511058)