BAB 2

LANDASAN TEORI

2.1.Kriptografi

2.1.1 Definisi Kriptografi

Kata kriptografi berasal dari bahasa Yunani, “kryptós” yang berarti tersembunyi dan “gráphein” yang berarti tulisan. Sehingga kata kriptografi dapat diartikan berupa frase “tulisan tersembunyi”. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Artinya, kriptografi dapat diartikan sebagai proses untuk melindungi data dalam arti yang luas. Tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi. Kriptografi dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan (Oppliger2005).

Sistem kriptografi adalah sistem yang dapat digunakan untuk mengubah pesan dengan tujuan membuat pesan tersebut tidak dimengerti oleh orang lain selain penerima. Proses ini disebut dengan Enkripsi (*encryption*) (Churchhouse, 2002).
Kriptografi memiliki 4 komponen utama yaitu:

a) *Plaintext*, yaitu pesan yang dapat dibaca yang akan disampaikan *sender* kepada *receiver*.
b) *Ciphertext*, yaitu pesan yang sudah dienkripsi yang tidak dapat dibaca.
c) Kunci, yaitu kunci untuk melakukan teknik kriptografi yang dikirimkan pengirim kepada penerima pesan untuk proses enkripsi dan dekripsi.
d) Algoritma, yaitu metode untuk melakukan enkrispi dan dekripsi.

Proses dari enkripsi dan dekripsi dapat dilihat pada Gambar 2.1.

Gambar 2.1 Diagram Proses Enkripsi dan Dekripsi

Dari gambar 2.1 menjelaskan diagram proses bagaimana terjadinya enkripsi dan dekripsi tersebut. Langkah pertama *plaintext* di enkripsi sehingga membentuk karakter ataupun kata yang sulit dibaca (*ciphertext*), kemudian *ciphertext* tersebut didekripsi agar dapat membentuk kata yang semula. Enkripsi adalah sebuah proses menjadikan pesan yang dapat dibaca (*plaintext*) menjadi pesan acak yang tidak dapat dibaca (*ciphertext*). Sedangkan dekripsi adalah proses kebalikan dari enkripsi dimana proses ini akan mengubah *ciphertext* menjadi *plaintext*. Kunci yang digunakan untuk melakukan enkripsi dan dekripsi terbagi menjadi 2 bagian, yaitu kunci pribadi (*private key*) dan kunci umum (*public key*) (Munir, 2006).

2.1.2. Tujuan Kriptorafi

Tujuan dari kriptografi yang juga merupakan aspek keamanan informasi adalah sebagai berikut (Menezes et al, 1996):

a) Kerahasiaan (confidentiality) adalah layanan yang digunakan untuk menyimpan isi informasi dari semua pihak kecuali pihak yang berwenang memiliki isi informasi tersebut. Kerahasiaan sangat identik dengan privasi. Ada banyak pendekatan untuk menjaga kerahasiaan, mulai dari pengamanan secara fisik hingga penggunaan algoritma matematika yang membuat data tidak dapat dipahami.
b) Integritas data adalah layanan yang menangani perubahan data dari pihak yang tidak berwenang. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berwenang. Manipulasi data mencakup hal-hal seperti penyisipan, penghapusan dan penggantian data.

c) Otentikasi adalah layanan yang berkaitan dengan identifikasi. Dua pihak yang saling berkomunikasi harus dapat mengidentifikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran harus diotentikasi seperti asal, tanggal, isi data, waktu dikirim dan lainnya.
d) Nirpenyangkalan (non-repudiation) adalah layanan yang mencegah suatu entitas yang berkomunikasi melakukan penyangkalan. Contohnya yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan. Prosedur yang melibatkan pihak ketiga yang terpercaya diperlukan untuk menyelesaikan masalah tersebut.

2.1.3. Kriptografi Simetris (Symmetric Cryptosystem)

Jenis kriptografi ini adalah yang paling umum digunakan. Kriptografi simetris atau disebut juga kriptografi konvensional adalah algoritma kriptografi yang menggunakan kunci yang sama pada proses enkripsi dan dekripsi. Keamanan pada kriptografi simetris ini terletak pada kuncinya, jadi pengirim dan penerima pesan harus memiliki kunci yang sama persis. Jika kunci dibocorkan maka siapapun yang mempunyai kunci tersebut termasuk pihak-pihak yang tidak diinginkan dapat mengetahui *ciphertext* (Basri, 2016). Penggunaan kriptografi simetris dapat dilihat pada gambar 2.2.

Gambar 2.2 Penggunaan Kriptografi Simetris (Sumber: Basri, 2016)

Algoritma kriptografi simetris dibagi menjadi 2 bagian yaitu algoritma blok (*block ciphers)* dan algoritma aliran (*stream ciphers).*

a. Algoritma Blok (*Block Ciphers)*

   Algoritma *block cipher* pada dasarnya adalah algoritma yang melakukan enkripsi sekelompok simbol *plaintext*dengan ukuran *m* dan membuat *ciphertext* dengan ukuran yang sama. Kunci yang sama digunakan untuk mengenkripsi ukuran blok. Contoh dari algoritma *block ciphers* adalah algoritma DES, algoritma AES, algoritma *blowfish* dan lain-lain (Arora & Gigras, 2014).

b. Algoritma Aliran (*Stream Ciphers)*

Algoritma *stream ciphers* adalah salah satu algoritma kunci simetris dimana *plaintext* digabungkan dengan *pseudorandom keystream*. Setiap digit plaintext akan dienkripsi satu per satu dengan digit *keystream* yang sesuai. Contoh dari algoritma *stream cipher* adalah algoritma *Spritz,* algoritma *RC4* dan algoritma $RC4^{+}$dan lainnya (Arora & Gigras, 2014).

2.1.4. Kriptografi Asimetris (Assymmetric Cryptosystem)

Kriptografi asimetris adalah pasangan kunci-kunci kriptografi yang salah satu kuncinya dipergunakan untuk proses enkripsi dan yang satu lainnya untuk proses dekripsi. Artinya, kriptograsi asimetris menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi. Pada gambar 2.3 dapat dilihat bahwa kunci publik dapat digunakan untuk mengenkripsi pesan oleh semua orang, sedangkan kunci pribadi untuk dekripsi pesan hanya dapat digunakan oleh orang yang memiliki kunci prbadi tersebut (Basri, 2016).

Gambar 2.3 Penggunaan Kriptografi Asimetris (Sumber: Basri, 2016)

2.2.Algoritma *Multi-Power RSA*

Salah satu cara untuk mempercepat proses dekripsi dari *RSA* adalah dengan menggunakan modulo dari rumus $N = p^{b-1}q$. Algoritma ini terdiri dari 3 buah proses, yaitu pembangkitan kunci, enkripsi dan dekripsi (Boneh & Shacham, 2002).

2.2.1. Landasan Teori Matematika Algoritma Multi-Power RSA

Dalam memahami dan mempelajari kriptografi, ada baiknya terlebih dahulu memahami dan mempelajari konsep-konsep dasar dalam perhitungan matematis yang digunakan dalam algoritma kriptogafi tersebut. Dalam algoritma *Multi-Power RSA* akan melibatkan konsep perhitungan matematis seperti bilangan prima, Faktor

Persekutuan Terbesar (*Greatest Common Divisor(GCD)),* relatif prima dan aritmatika modulo.

2.2.1.1.Bilangan Prima

Bilangan dikatakan prima apabila sebuah bilangan tersebut lebih besar dari 1 dan merupakan bilangan bulat. Bilangan prima hanya memiliki faktor 1 dan bilangan itu sendiri dan tidak memiliki bilangan pembagi lainnya. Bilangan yang bukan bilangan prima disebut bilangan komposit. Contoh bilangan prima adalah 2, 3, 11, 17, 19, dan lain sebagainya.

2.2.1.2. Faktor Persekutuan Terbesar (*Greatest Common Divisor(GCD)*)

Faktor persekutuan terbesar atau *Greatest Common Divisor* merupakan elemen terbesar pada himpunan semua faktor dua buah bilangan integer. Dua bilangan tersebut mungkin saja memiliki beberapa elemen faktor yang sama namun hanya satu yang terbesar. Misalnya 18 memiliki faktor{1, 2, 3, 6, 9, 18}, dan 48 memiliki faktor{1, 2, 3, 4, 6, 8, 12, 24, 48}. Maka himpunan faktorbersama yang dimiliki oleh dua bilangan tersebut adalah {1, 2, 3, 6} dan yang terbesar adalah 6. Sehingga dapat dinotasikan sebagai *GCD*(18,48) = 6 (Sadikin, 2012).

2.2.1.3. Inversi Modulo

Inversi modulo merupakan bilangan *e* yang mempunyai modulo *n* jika dan hanya jika *GCD*(*e, n*) = 1. Inversi modulo dari *e* dalam modulo n dinotasikan dengan e^{-1}. Persamaan inversi *e* modulo *n* dapat ditulis dengan $e^{-1}.e = 1 \pmod{n}$.

2.2.1.4. Relatif Prima

Dua buah bilangan dikatakan relatif prima jika dan hanya jika *GCD* dari dua buah bilangan tersebut adalah 1. *GCD*(*a, b*) = 1, maka dapat diartikan *a* relatif prima dengan *b*.

2.2.1.5. Aritmatika Modulo

Aritmatika modulo digunakan agar operasi aritmatika selalu menghasilkan integer pada lingkup yang sama. Operasi modulo memerlukan 2 buah masukan yaitu sebuah bilangan bulat *a* dan sebuah bilangan positif yang disebut modulus *b*. Operasi modulo mengembalikan nilai *r* yang merupakan sisa bagi atas operasi *a* dibagi *b*. Notasi operasi modulus dituliskan sebagai *a* mod *b* = *r* (Sadikin, 2012). Tabel 2.1 menunjukkan beberapa persamaaan aritmatika dengan modulo 11.

Tabel 2.1 Aritmatika Modulo 11

Ekspresi	Hasil
9	9
18 – 10	8
7 + 5	1
5 x 6	8
-9	2

2.2.1.6. Pengujian Bilangan Prima Algoritma Agrawal Kayal Saxena (AKS)

Pada tahun 2002 tiga peneliti berkebangsaan India bernama Manindra Agrawal, Neeraj Kayal, dan Nitin Saxena telah membangun sebuah algoritma pengujian bilangan prima yang mempunyai sifat deterministik dalam waktu polinomial yang diberi nama *Algoritma AKS*. Diberikan sebuah bilangan bulat *p* yang ingin diuji keprimaannya. Selanjutnya pilih lagi sebulah bilangan bulat positif *z*, yang mana $2 \leq z < p - 2$ dan GCD(p, z) = 1 (*Agrawal et al*, 2002). Maka *p* merupakan bilangan prima jika dan hanya jika:

$$(1 + z)^p \equiv 1 + z^p (mod\ p)$$

Contoh: Menguji keprimaan dari 23. Maka perhitungan menggunakan algoritma *Agrawal Kayal Saxena* dapat dilakukan dengan cara berikut:

1. Bilangan yang ingin diuji keprimaannya adalah 11, maka $p = 11$.
2. Menentukan *z* dengan cara $2 \leq z < p - 2$, maka $2 \leq z < 9$ oleh karena itu *z* yang dapat dibentuk adalah {2, 3, 4, 5, 6, 7, 8}. Pilih satu nilai *z* yang akan digunakan misalnya 5.

3. Kemudian membandingkan nilai yang ada di ruas kanan dengan nilai yang ada di ruas kiri apakah sama jika dimodulo dengan *p*.

   $(1 + 5)^{11} \equiv 1 + 5^{11}$ (*mod* 11)

   $6^{11} \equiv 1 + 5^{11}$ (*mod* 11)

   $362797056 \equiv 1 + 48828125$ (*mod* 11)

   $362797056 \equiv 48828126$ (*mod* 11)

   $6 \equiv 6$ (*mod* 11)

Karena nilai yang ada di ruas kiri sama dengan nilai yang ada di ruas kanan dalam modulo *p*, maka *p* merupakan bilangan prima. Namun, jika nilai yang ada di ruas kiri dan nilai yang ada di ruas kanan tidak sama, maka *p* bukan merupakan bilangan prima (Bilangan Komposit).

1. Pembangkitan kunci

   Algoritma pembangkitan kunci mengambil sebuah masukan sebagai parameter *n* dan sebuah parameter tambahan *b*. Algoritma ini membangkitkan pasangan *public key* dan *private key* RSA dengan ketentuan sebagai berikut:

   a) Bangkitkan 2 buah bilangan prima dengan panjang *n-bit*, yaitu *p* dan *q* kemudian hitunglah $N = p^{b-1}q$.

   b) Selanjutnya gunakan pangkat publik *e* yang biasanya digunakan pada algoritma kunci publik RSA *standard,* yaitu *e* = 65537. Kemudian hitunglah $d = e^{-1} \bmod (p - 1)(q - 1)$.

   c) Lalu hitung lagi $r_1 = d \bmod p - 1$ dan $r_2 = d \bmod q - 1$.

   Maka *public key*-nya adalah {*N,e}* dan *private key*-nya adalah {p, q, r_1, r_2}.

2. Enkripsi

   Proses enkripsi pada algoritma *multi-power RSA* ini sama dengan proses enkripsi yang ada pada algoritma RSA *standard*. Pada proses ini *public key* {*N,e}* akan digunakan. Untuk mengenkripsi *plaintext* menjadi *ciphertext* dapat dilakukan dengan cara berikut:

   $$C = m^e \bmod N$$

3. Dekripsi

   Untuk mendekripsi *ciphertext* C digunakan *private key* {p, q, r_1, r_2}, dengan cara (Vuillaume, 2002):

   a) Hitunglah, $p^2 = p.p$, $C_1 = C \bmod p^2$, dan $C_2 = C \bmod q$.

   b) Kemudian, dengan menggunakan *Hensel Lifting,* hitunglah nilai dari:

      i. $M_1 = C_1^{r1-1} \bmod p$

      ii. $K_0 = M_1.C_1 \bmod p$

      iii. $A = -K_0^{e} \bmod p^2$

      iv. $A = A + C \bmod p^2$

      v. $M_1 = M_1.A \bmod p^2$

      vi. $M_1 = M_1.(e^{-1} \bmod p) \bmod p^2$

      vii. $M_1 = M_1 + K_0 \bmod p^2$

   c) Selanjutnya, pada proses akhir dari dekripsi pesan, hitunglah nilai dari:

      i. $M_2 = C_2^{r2} \bmod q$

      ii. $V = M_2 - M_1 \bmod q$

      iii. $V = V.(p^{2-1} \bmod q) \bmod q$

      iv. $M = V.p^2 \bmod N$

      v. $M = M + M_1 \bmod N$, maka pesan berhasil terdekripsi.

Contoh pembangkitan kunci, enkripsi, dan dekripsi menggunakan algoritma *Multi-Power RSA,* sebagai berikut:

Tentukan proses enkripsi dan dekripsi pada kata berikut : “Mahadi”

a) Ubah karakter masing-masing plaintext menjadi kode karakterM = 77, a = 97, h = 104, a = 97, d = 100, i = 105.

b) Selanjutnya akan dibangkitkan pasangan *public key* dan *private key* dengan menentukan *n* dan *b*. Misalnya *n = 4* dan *b = 3*.

c) Kemudian akan ditentukan 2 buah bilangan prima yang mempunyai panjang *n-bit*. Misalnya 11 dan 13, maka bilangan prima tersebut kita konversikan menjadi bilangan yang mempunyai panjang *n = 4bit*, sehingga:

   11 = 1011 dan 13 = 1101

Selanjutnya menghitung nilai *N* dengan rumus $N = p^{b-1}q$, maka nilai *N* adalah

$$N = 11^{3-1} \text{ x } 13$$

$$N = 1573$$

d) Selanjutnya menghitung nilai *d*, dengan rumus $d = e^{-1} \bmod (p-1)(q-1)$. Nilai *e* yang biasa digunakan adalah nilai *e* yang digunakan pada algoritma kunci publik RSA *standard*, yaitu *e* = 65537, namun untuk mempermudah perhitungan ini, pilihalah nilai *e* yang memenuhi syarat $2 < e < (p\text{-}1)(q\text{-}1)$.Sehingga *e* yang digunakan adalah 7. Sehingga nilai *d* adalah:

$d = e^{-1} \bmod (p-1)(q-1)$

$d = 103 \bmod (11-1)(13-1)$ *invers dari e = 7 adalah 103*

$d = 103 \bmod 120$

$d = 103$

e) Lalu menghitung nilai $r_1$dan r_2 dengan cara seperti berikut:

$r_1 = d \bmod (p-1)$	$r_2 = d \bmod (q-1)$
$r_1 = 103 \bmod (11-1)$	$r_2 = 103 \bmod (13-1)$
$r_1 = 103 \bmod 10$	$r_2 = 103 \bmod 12$
$r_1 = 3$	$r_2 = 7$

Maka *public key*-nya {*N, e}* adalah {1573, 7} dan *private key*-nya {p, q, r_1, r_2} adalah {11, 13, 3, 7}.

f) Selanjutnya akan dienkripsi kata “adi” dengan menggunakan rumus:

$C = m^e \bmod N$ sehingga:

a $= 97^7 \bmod 1573 = 59$

d $= 100^7 \bmod 1573 = 815$

i$= 105^7 \bmod 1573 = 261$

Maka *ciphertext* dari kata “adi” adalah {363, 59, 520, 59, 815, 261}

g) Dekripsi Pesan

- Dekripsi karakter “a”:

1) $p^2 = p.p$

$= 11.11$

$= 121$

2) $C_1 = C \bmod p^2$

$= 59 \bmod 121$

$= 59$

3) $C_2 = C \bmod q$

$= 59 \bmod 13$

$= 7$

4) HENSEL LIFTING

a. $M_1 = C_1^{r1-1} \bmod p$

$= 59^{3-1} \bmod 11$

$= 5$

b. $K_0 = M_1.C_1 \bmod p$

$= 5.59 \bmod 11$

$= 9$

c. $A = -K_0^{e} \bmod p^2$

$= -(9)^7 \bmod 121$

$= 40$

d. $A = A + C \bmod p^2$

$= 40 + 59 \bmod 121$

$= 99$

e. $M_1 = M_1.A \bmod p^2$

$= 5.99 \bmod 121$

$= 11$

f. $M_1 = M_1.(e^{-1} \bmod p) \bmod p^2$

$= 11.(e^{-1} \bmod 11) \bmod 121$

$= 11.8 \bmod 121$

$= 88$

g. $M_1 = M_1 + K_0 \bmod p^2$

$= 88 + 9 \bmod 121$

$= 97$

5) $M_2 = C_2^{r2} \bmod q$

$= 7^7 \bmod 13$

$= 6$

6) $V = M_2 – M_1 \bmod q$

$= 6 – 97 \bmod 13$

$= 0$

7) V = $V.((p^2)^{-1} \bmod q) \bmod q)$
= $0.((121^{-1} \bmod 13) \bmod 13)$
= $0.10 \bmod 13$
= 0

8) M = $V.p^2 \bmod N$
= $0.121 \bmod N$
= $0 \bmod 1573$
= 0

9) M = $M + M_1 \bmod N$
= $0 + 97 \bmod 1573$
= 97 (karakter a berhasil di dekripsi)

- Dekripsi karakter “d”:

1) p^2 = p.p
= 11.11
= 121

2) C_1 = $C \bmod p^2$
= $815 \bmod 121$
= 89

3) C_2 = $C \bmod q$
= $815 \bmod 13$
= 9

4) HENSEL LIFTING

a. M_1 = $C_1^{r1-1} \bmod p$
= $89^{3-1} \bmod 11$
= 1

b. K_0 = $M_1.C_1 \bmod p$
= $1.89 \bmod 11$
= 1

c. A = $-K_0^e \bmod p^2$
= $-(1)^7 \bmod 121$
= 120

d. A = $A + C \bmod p^2$

$= 120 + 89 \bmod 121$

$= 209 \bmod 121$

$= 88$

e. $M_1 = M_1.A \bmod p^2$

$= 1.88 \bmod 121$

$= 88$

f. $M_1 = M_1.(e^{-1} \bmod p) \bmod p^2$

$= 88.(e^{-1} \bmod 11) \bmod 121$

$= 88.8 \bmod 121$

$= 704 \bmod 121$

$= 99$

g. $M_1 = M_1 + K_0 \bmod p^2$

$= 99 + 1 \bmod 121$

$= 100$

5) $M_2 = C_2^{r2} \bmod q$

$= 9^7 \bmod 13$

$= 9$

6) $V = M_2 - M_1 \bmod q$

$= 9 - 100 \bmod 13$

$= 0$

7) $V = V.((p^2)^{-1} \bmod q) \bmod q)$

$= 0.((121^{-1} \bmod 13) \bmod 13)$

$= 0.10 \bmod 13$

$= 0$

8) $M = V.p^2 \bmod N$

$= 0.121 \bmod N$

$= 0 \bmod 1573$

$= 0$

9) $M = M + M_1 \bmod N$

$= 0 + 100 \bmod 1573$

$= 100$ (karakter d berhasil di dekripsi)

- Dekripsi karakter “i”:

1) p^2 = p.p
   = 11.11
   = 121
2) C_1 = C mod p^2
   = 261 mod 121
   = 19
3) C_2 = C mod q
   = 261 mod 13
   = 1
4) HENSEL LIFTING
   a. M_1 = C_1^{r1-1} mod p
      = 19^{3-1} mod 11
      = 9
   b. K_0 = $M_1.C_1$ mod p
      = 9.19 mod 11
      = 6
   c. A = $-K_0^{e}$ mod p^2
      = $-(6)^7$ mod 121
      = -(279936) mod 121
      = 58
   d. A = A + C mod p^2
      = 58 + 261 mod 121
      = 319 mod 121
      = 77
   e. M_1 = M_1.A mod p^2
      = 9.77 mod 121
      = 693 mod 121
      = 88
   f. M_1 = $M_1.(e^{-1}$ mod p) mod p^2
      = 88.(e^{-1} mod 11) mod 121
      = 88.8 mod 121
      = 99

g. $M_1 = M_1 + K_0 \bmod p^2$

$= 99 + 6 \bmod 121$

$= 105 \bmod 121$

$= 105$

5) $M_2 = C_2^{r2} \bmod q$

$= 1^7 \bmod 13$

$= 1$

6) $V = M_2 - M_1 \bmod q$

$= 1 - 105 \bmod 13$

$= -104 \bmod 13$

$= 0$

7) $V = V.((p^2)^{-1} \bmod q) \bmod q)$

$= 0.((121^{-1} \bmod 13) \bmod 13)$

$= 0.10 \bmod 13$

$= 0$

8) $M = V.p^2 \bmod N$

$= 0.121 \bmod N$

$= 0 \bmod 1573$

$= 0$

9) $M = M + M_1 \bmod N$

$= 0 + 105 \bmod 1573$

$= 105$ (karakter i berhasil di dekripsi)

2.3.Algoritma *Blowfish*

Blowfish mempunyai nama lain *OpenPGP.Cipher.4* yang merupakan enkripsi golongan *Symmetric Cryptosystem*, metode enkripsinya mirip dengan DES (DES-like Cipher) diciptakan oleh seorang *Cryptanalyst* bernama Bruce Schneier. *Blowfish* termasuk dalam enkripsi block Cipher 64-bit dangan panjang kunci yang bervariasi antara 32-bit sampai 448-bit(Mollin, 2005). Algoritma *Blowfish* terdiri atas dua bagian, yaitu:

1. Key-Expansion

   Berfungsi untuk merubah kunci (Minimum 32-bit, Maksimum 448-bit) menjadi beberapa array subkunci (subkey) dengan total 4168 byte.

2. Enkripsi Data

   Terdiri dari iterasi fungsi sederhana (*Feistel Network*) sebanyak 16 kali putaran. Setiap putaran terdiri dari permutasi kunci-*dependent* dan substitusi kunci- dan data-*dependent*.

Langkah kerja algoritma *Blowfish* terdiri atas dua bagian, yaitu (Kurnia, 2016):

a) Proses Ekspansi Kunci (Key Expansion)

1. Inisialisasi P-array yang pertama dan juga empat S-box, berututan, dengan string yang telah pasti.String tersebut terdiri dari digit-digit heksadesimal dari phi, tidak termasuk angka tiga di awal.

   Contoh :

   P_1= 0x243f6a88

   P_2= 0x85a308d3

   P_3= 0x13198a2e

   P_4= 0x03707344

   dan seterusnya sampai dengan S-box yang terakhir atau P_{18}.

2. Kemudian P_1di XOR dengan 32-bit awal kunci, P_2di XOR dengan 32-bit berikutnya dari kunci, dan seterusnya untuk semua bit kunci. Jika panjang kunci ternyata kurang dari jumlah Pbox, maka siklus perhitungan akan diulangi hingga semua P ter-XOR-kan.
3. Enkripsikan string yang seluruhnya nol (all-zero string) dengan algoritma *Blowfish*, menggunakan subkunci yang telah didekripsikan pada langkah 1 dan 2.
4. Gantikan P_1 dan P_2 dengan keluaran dari langkah 3.
5. Enkripsikan keluaran langkah 3 menggunakan algoritma *Blowfish* dengan subkunci yang telah dimodifikasi.
6. Gantikan P_3 dan P_4 dengan keluaran dari langkah 5.
7. Lanjutkan langkah-langkah di atas, gantikan seluruh elemen P-array dan juga gantikan ke-empat S-box secara berurutan, dengan hasil keluaran algoritma *Blowfish* yang terus-menerus berubah.

b) Proses Enkripsi Data (Faldy, 2016)

Proses enkripsi sebelumnya telah dibahas dan digunakan pada proses perluasan kunci. Kali ini akan diterapkan pada *plaintext* yang berbeda dengan anggapan bahwa proses perluasan kunci telah diselesaikan dengan baik sehingga nilai semua blok P dan S telah diperbaharui. Hal ini penting dikarenakan proses enkripsi dapat dilakukan jika proses perluasan kunci telah selesai.

1. *Plaintext* yang digunakan adalah “al p 4a ?”, dimana *plaintext* dalam nilai angka = 263752415256265289. Nilai tersebut dipecah menjadi nilai xL dan xR yang masing-masing berukuran 32 bit, dimana xL merupakan 32 bit pertama dari *plaintext* dan xR 32 bit kedua dari *plaintext*. Didapatkan nilai xL= 26375241 dan xR= 52562652.
2. Perulangan dilakukan sebanyak 16 kali dimana disetiap perulangan dilakukan operasi xor xL dengan Pi dimana i menunjukkan jumlah perulangan yang sudah dilakukan. Lalu, nilai xR merupakan hasil dari operasi xor antara (xL) dan xR sendiri. Setelah didapatkan nilai xLdan xR, maka kedua nilai tersebut dipertukarkan satu dengan yang lain. Didapatkan hasil sebagaimana terlihat pada tabel 2.2.

Tabel 2.2 Proses Enkripsi Algoritma *Blowfish*

Indeks Proses	P Indeks Proses	XL	F(XL)	XR
1	8D717830	2373003349	8064871505	6775194193
2	884446F9	4757456040	7702866319	5474697690
3	AE21C62E	8194656244	9891654133	14328245597
4	BF256408	16796272981	3141382032	5692532836
5	FFA3C4B2	7196306646	2190551807	14691853226
6	25BE839C	14194401334	4396058258	2867358276
7	3BE7ED1B	2433714015	8032108102	11019312752
8	C8A843D3	10072956323	5225731923	7087706124
9	988940B3	5351706815	9358013627	1973610776
10	DEC445AC	2875632820	4330329214	1021772993
11	C36BE21B	4287424218	9554018848	11040485012
12	21DBF4AE	11606399546	4282496971	13455633
13	F0F8B6EB	4030064634	11368883188	375978446

14	ECB21FE2	4208649772	10029928532	11372951470
15	2215D28D	10870891811	3622815553	758456685
16	8034E687	2902575082	11963342236	1323743423

3. Nilai xL dan xR dipertukarkan sehingga,

   xL= 2902575082

   xR= 1323743423

4. Dilakukan operasi *xor* terhadap xR dan kunci $P17$

   xR=$xR \oplus P17$

   xR= 1258505066

5. Dilakukan operasi *xor* terhadap xL dan kunci $P18$

   xL= $xL \oplus P18$

   xL= 3371315541

Masing-masing nilai xR dan xL dilakukan operasi *and* terhadap bilangan heksadesimal FFFF FFFF agar ukuran nya tepat 32 *bit*.

xR= 1258505066 and 4294967295

xR= 1258505066

xL=(3371315541 and 4294967295)$\ll$32

xL=3371315541$\ll$32

xL= 14479689993091547136

Hasil dari proses ini adalah $xL \oplus xR$ = 11759496057487293479.

Cipher text blowfish : L]˙¦F[>di:

c) Proses Dekripsi

Proses dekripsi pada algoritma *Blowfish* hampir sama dengan proses enkripsinya. Perbedaan terletak pada urutan penggunaan kunci, yaitu kunci dimulai dari indeks paling tinggi menuju indeks 1. Disini akan dilakukan proses dekripsi terhadap *ciphertext* ” L]˙¦F[>di: ”dengan menggunakan kunci yang sudah didekripsi oleh algoritma RSA yaitu “9na=36W7_” yang dihasilkan proses enkripsi sebelumnya. Proses Dekripsi dapat dilihat pada tabel 2.3.

1. Nilai angka dari *ciphertext* = 11759496057487293479
2. Dilakukan proses perulangan sebanyak 16 kali terhadap nilai xL dan xR.

Tabel 2.3 Proses Dekripsi Algoritma *Blowfish*

Indeks Proses	P indeks Proses	XL	F(XL)	XR
1	65F3F6BF	2902575082	11963342236	10772111094
2	5E5FBD5	10870891811	3622815553	2062418091
3	8034E687	4208649772	10029928532	3525326199
4	2215D28D	4030064634	11368883188	10191734232
5	ECB21FE2	11606399546	4282496971	259280945
6	F0F8B6EB	4287424218	9554018848	2327664666
7	21DBF4AE	2875632820	4330329214	8549528228
8	C36BE21B	5351706815	9358013627	10848655375
9	DEC445AC	10072956323	5225731923	159827948
10	988940B3	2433714015	8032108102	15143871461
11	C8A843D3	14194401334	4396058258	6828917197
12	3BE7ED1B	7196306646	2190551807	16317696713
13	25BE839C	16796272981	3141382032	4694688582
14	FFA3C4B2	8194656244	9891654133	7058251936
15	BF256408	4757456040	7702866319	592506491
16	AE21C62E	2373003349	8064871505	4213516537

3. Nilai xL dan xR dipertukarkan sehingga

   xL= 2373003349

   xR= 4213516537

4. Dilakukan operasi *xor* terhadap xR dan kunci $P17$

   $xR=xR \oplus P17$

   xR= 52562652

5. Dilakukan operasi *xor* terhadap xL dan kunci $P18$

   $xL= xL \oplus P18$

   xL= 20581

Masing-masing nilai xR dan xL dilakukan operasi *and* terhadap bilangan heksadesimal FFFF FFFF agar ukuran nya tepat 32 *bit*.

xR= 52562652 and 4294967295

xR= 52562652

xL=(20581 and 4294967295)$\ll$32

xL=20581$\ll$32 xL= 88394721918976

Hasil dari proses ini adalah $xL \oplus xR$ = 263752415256265289.

Plaintext blowfish : al p 4a ?

2.4.*Hybrid Cryptosystem*

Hybrid cryptosystem merupakan gabungan antara *Cryptosystem* yang memakai algoritma simetris dan *Cryptosystem* yang memakai algoritma asimetris. Protokol kriptografi modern pada saat ini banyak yang menggabungkan algoritma simetris dengan algoritma asimetris untuk memperoleh keunggulan-keunggulan pada masing-masing algoritma (Schneier, 1996).

Teknik enkripsi menggunakan algoritma asimetris lebih lambat dibandingkan enkripsi dengan menggunakan algoritma simetris, karena pada algoritma asimetris menggunakan perhitungan matematika yang sangat rumit, langkahnya *plaintext* dienkripsi dengan algoritma simetris kemudian kunci *private* algoritma simetris dienkripsi dengan algoritma asimetris (Bridgeca, 1999). Proses dan cara kerja *hybrid cryptosystem* dapat dilihat pada gambar 2.4.

Gambar 2.4 Proses dan Cara Kerja *Hybrid Cryptosystem*

2.5.Penelitian yang Terdahulu

Beberapa penelitian terdahulu yang relevan dengan penelitian yang akan dilakukan oleh penulis antara lain adalah sebagai berikut:

- Dua algoritma enkripsi yaitu ECC dan *Blowfish* terkenal dengan penyediaan level keamanan dan kecepatan enkripsi yang lebih cepat. ECC sudah berhasil di-*attack* tetapi tidak dengan *Blowfish*. Dua algoritma enkripsi ini lebih aman dan bekerja dengan cepat (Bhanot & Hans, 2015).
- Ketika menggunakan memori yang sama, algoritma *Multi-power*RSAmodulo $N = P^2Q$ lebih cepat sekitar tiga kali lipat dibandingkan dengan algoritma *RSA* klasik dengan menggunakan *Chinese remainder theorem* (dan lebih cepat duabelas kali lipat dibandingkan *RSA* tanpa menggunakan *Chinese remainder theorem*) (Vuillaume, 2002).
- (Boneh & Shacham, 2002) meneliti 4 variasi dari desain RSA untuk mempercepat dekripsi RSA dan menjadikannya kompatibel dengan standar RSA. Dua teknik *Multi-factor RSA* menjanjikan bahwa dua teknik tersebut sepenuhnya kompatibel. *Multi-factor RSA* dan *Rebalanced RSA* dapat digunakan bersama-sama untuk memberikan kecepatan tambahan.
- (Fauziah, 2008) menyimpulkan bahwa keamanan data tergantung pada sisi kunci, semakin besar atau panjang kunci semakin lama waktu yang dibutuhkan untuk memecahkannya. Kunci yang besar atau yang panjang tersebut tidak memungkinkan *user* untuk menghafalnya, maka pada HBR*Crypto* versi 1.0 kunci dibangkitkan oleh suatu *password* yang memungkinkan *password* tersebut dapat disadap oleh user lain yang tidak berkepentingan.
- Dengan menggunakan konsep kriptografi asimetris kerahasiaan kunci lebih terjaga, karena memiliki *private key* dan *public key* yang memiliki fungsi yang berbeda serta didukung oleh panjang *private key* yang relatif lebih panjang yaitu 1024 bit (Munawar, 2013).