BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada proses pengiriman data (pesan) terdapat beberapa hal yang harus diperhatikan, yaitu : kerahasiaan, integritas data, autentikasi dan non repudiasi. Oleh karenanya dibutuhkan suatu proses penyandian atau pengkodean data sebelum dilakukan proses pengiriman. Sehingga data yang dikirim terjaga kerahasiaannya dan tidak dapat dengan mudah diubah untuk menjaga integritas data tersebut.

Ilmu yang mempelajari tentang cara-cara pengamanan data dikenal dengan istilah Kriptografi, sedangkan langkah-langkah dalam kriptografi disebut algoritma kriptografi. Berdasarkan dari kunci yang digunakan algoritma kriptografi dapat dibagi menjadi dua, Algoritma Simetri dan Algoritma Assimetri. Dimana Algoritma Simetri menggunakan satu kunci untuk proses enkripsi dan dekripsinya. Algoritma kriptografi yang menggunakan kunci simetri adalah, DES, RC2, RC4, RC5, RC6, IDEA, AES, OTP, A5 dan lain sebagainya.

Sedangkan Algoritma Assimetri menggunakan dua kunci berbeda untuk proses enkripsi dan dekripsinya, yaitu kunci umum (public key) yang digunakan untuk proses enkripsi (perubahan data plain teks menjadi chipper text) yang sifatnya tidak rahasia, dan kunci pribadi (private key) yang digunakan untuk proses dekripsi (pengembalian data chipper text menjadi plain text) yang sifatnya rahasia dan masing-masing pihak memiliki kunci pribadi yang berbeda.

Penggunaan kunci pribadi dapat digunakan untuk autentikasi (pengenalan identitas pengirim) dan non repudiasi (pencegahan penyangkalan pengiriman data) karena dalam proses dekripsi dapat diketahui siapa pihak pengirim dengan melihat kunci pribadi yang dipakai.

Dalam proses penyandian, penyandian yang biasa dipakai adalah RSA Coding, dimana RSA Coding merupakan proses penyandian kunci asimetris (asymmetric key). Proses perumusan RSA Coding didasarkan pada Teorema Euler, sedemikian sehingga menghasilkan kunci umum dan kunci pribadi yang saling berkaitan. Sehingga meskipun proses enkripsi dan dekripsi menggunakan dua kunci yang berbeda hasilnya akan tetap benar. Kunci umum dan kunci pribadi yang digunakan adalah suatu bilangan prima, dan disarankan bilangan prima yang besar. Hal ini digunakan untuk pencegahan usaha pemecahan chipper text, karena semakin besar bilangan prima yang digunakan sebagai kunci maka semakin sulit mencari bilangan besar sebagai faktornya.

1.2 Perumusan Masalah

Berdasarkan apa yang telah diuraikan diatas, maka akan dibahas mengenai bagaimana penerapan Teorema Euler pada RSA Coding.

1.3 Pembatasan Masalah

Dalam proses penyandian teknik yang dapat digunakan cukup banyak, maka dibutuhkan pembatasan masalah. Berdasarkan dari kunci yang digunakan algoritma kriptografi dapat dibagi menjadi dua, Algoritma Simetri dan Algoritma

Assimetri. Teknik penyandian asimetris ada beberapa macam yaitu RSA, DSA dan El Gamal. Dalam hal ini hanya akan dibahas tentang RSA Coding. RSA Coding sendiri dalam prosesnya berdasarkan Teorema Euler.

1.4 Metode Pembahasan

Metode yang digunakan penulis dalam penyusunan tugas akhir ini adalah metode studi literatur. Terlebih dahulu penulis akan menjabarkan materi – materi dasar yang berkaitan dengan kriptografi dan aljabar khususnya tentang teori bilangan, seperti pengertian kriptografi, macam-macam kriptografi dan hal-hal yang berkaitan dengan keamanan kriptografi. Selanjutnya penulis juga akan menjelaskan mengenai teorema bilangan, seperti teorema-teorema yang berkaitan dengan bilangan prima dan aritmetika modulo.

Setelah itu penulis akan menjabarkan beberapa lema dan teorema yang berkaitan dengan penerapan aljabar dalam perumusan Algoritma RSA Coding.

1.5 Tujuan Penulisan

Tujuan penulisan Tugas Akhir ini adalah untuk mengetahui bagaimana penerapan Teorema Euler pada RSA Coding.

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini meliputi empat bab. Bab I merupakan bab pendahuluan yang mencakup latar belakang, perumusan masalah, pembatasan masalah, metode penulisan, tujuan penulisan, dan sistematika penulisan. Bab II

merupakan bab teori penunjang yang berisi penjelasan mengenai kriptografi, definisi – definisi dan teorema-teorema yang mendukung dan mendasari penulisan ini, yaitu mengenai teorema bilangan, aritemtika modulo, algoritma Euclid, system residu dan fungsi phi euler (φ). Sedangkan bab III merupakan bab pembahasan mengenai teorema-teorema yang digunakan untuk merumuskan algoritma RSA Coding dan Pembangkitan Pasangan Kunci. Dilanjutkan dengan penjelasan tentang keamanan RSA Coding, implemetasi RSA Coding dan perbandingan dengan algoritma kriptografi yang lain. Bab IV merupakan bab penutup yang berisi tentang kesimpulan dan saran dari hasil pembahasan sebelumnya.

BAB II

DASAR TEORI

2.1. Kriptografi

Kriptografi adalah ilmu untuk mengacak pesan sedemikian rupa sehingga tidak bisa dibaca oleh pihak ketiga atau yang tidak diberi otorisasi. Pesan sendiri dapat diartikan sebagai informasi yang berharga, sehingga dibutuhkan proses untuk menjaga kerahasiaan informasi tersebut. Tentu saja pesan yang diacak dalam bentuk kriptografis ini harus bisa dikembalikan ke bentuk semula oleh pihak yang berwenang. Dasar dari kriptografis ini sendiri adalah untuk menjaga kerahasiaan.

Di dalam kriptografi pesan yang ingin diacak bisaanya disebut Plain Text. Pesan diacak dengan menggunakan Kunci Enkripsi (Encryption Key) sementara proses pengacakannya disebut Enkripsi (Encryption). Plain Text yang telah diacak disebut Cipher Text. Setelah pesan dikirim, kemudian akan diproses untuk dikembalikan lagi menjadi plain teks. Proses untuk mengembalikan Cipher Text ke Plain Text ini disebut Dekripsi (Decryption). Kunci yang digunakan pada tahap Dekripsi disebut Kunci Dekripsi (Decryption Key).

2.1.1. Aspek-Aspek Keamanan Kriptografi

Dalam perkembangannya, teknik kriptografi tidak hanya digunakan semata-mata untuk menjaga kerahasiaan saja. Aspek yang kemudian

muncul seiring dengan semakin kompleksnya masalah yang dihadapi ketika dilakukan pengiriman pesan adalah:

1. Autentikasi (Authentication).

   Proses untuk menjamin keaslian suatu pesan, sehingga pihak yang menerima pesan dapat memastikan keaslian pesan tersebut datang dari orang yang dimintai informasi. Dengan kata lain informasi tersebut benar-benar datang dari orang yang dikehendaki.

2. Integritas (Integrity).

   Proses untuk menjaga agar sebuah pesan tidak diubah-ubah sewaktu dikirim atau disimpan. Perubahan pesan saat dilakukan pengiriman tentu membawa dampak yang tidak kecil, terutama ketika pesan tersebut nantinya digunakan sebagai bahan pengambil keputusan. Oleh karena itu, kriptografi juga harus membuat pesan asli tidak dapat diubah saat dikirim atau disimpan.

3. Penghindaran Penolakan (Non-repuditation).

   Proses untuk menjaga bukti-bukti bahwa suatu pesan berasal dari seseorang. Sehingga pihak yang mengirim tidak bisa menyangkal bahwa pesan tersebut berasal dari pihak tersebut.

4. Kerahasiaan (Confidentiality).

   Kerahasiaan adalah proses penyembunyian pesan dari orang-orang yang tidak punya otoritas sehingga pesan penting hanya akan dibaca oleh orang yang dituju.

2.1.2. Macam-Macam Algoritma Kriptografi

Berdasarkan dari kunci yang digunakan algoritma kriptografi dibagi menjadi dua, yaitu Algoritma Simetri dan Algoritma Assimetri.

1. Algoritma Simetri

Algoritma Simetri menggunakan satu kunci untuk proses enkripsi dan dekripsinya. Keamanan dari pesan yang menggunakan altoritma ini tergantung pada kunci, jika kunci tersebut diketahui orang lain maka orang tersebut dapat melakukan enkripsi dan dekripsi terhadap pesan tersebut.

Algoritma kriptografi yang menggunakan kunci simetri adalah, DES (Data Encryption Standard), RC2, RC4, RC5, RC6, IDEA (International Data Encryption Algorithm), AES (Advanced Encryption Standard), OTP (One Time Pad), A5 dan lain sebagainya.

2. Algoritma Assimetri

Algoritma Assimetri menggunakan dua kunci berbeda untuk proses enkripsi dan dekripsinya. Kunci umum (public key), kunci yang semua orang boleh tahu. Kunci umum digunakan untuk proses enkripsi. Kunci pribadi (private key), kunci yang dirahasiakan, hanya satu orang yang boleh tahu. Kunci pribadi digunakan untuk proses dekripsi. Kunci-kunci tersebut saling berhubungan satu dengan yang lainnya.

Algoritma assimetris lebih aman dari algoritma simetris. Algoritma kriptografi yang menggunakan algoritma assimetris adalah, DSA (Digital Signature Algorithm), RSA (Rivest-Shamir-Adleman), DH (Diffie-Hellman), ECC (Elliptic Curve Cryptography) dan lain sebagainya

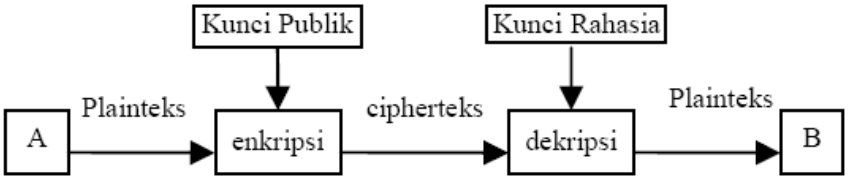
.

2.2. RSA Coding

Sandi RSA merupakan algoritma kriptografi kunci public (asimetris). Ditemukan pertama kali pada tahun 1977 oleh Ron Rivest, Adi Shamir, dan Len Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut.

Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. RSA mendasarkan proses enkripsi dan dekripsinya pada konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun dekripsi keduanya merupakan bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diberikan kepada umum (sehingga disebut dengan kunci publik), namun kunci untuk dekripsi bersifat rahasia (kunci privat).

Untuk menemukan kunci dekripsi, dilakukan dengan memfaktorkan suatu bilangan bulat menjadi faktor-faktor primanya. Kenyataannya, memfaktorkan bilangan bulat menjadi faktor primanya bukanlah pekerjaan yang mudah. karena belum ditemukan algoritma yang efisien untuk melakukan pemfaktoran. Cara yang bisaa digunakan dalam pemfaktoran adalah dengan menggunakan pohon faktor. Jika semakin besar bilangan yang akan difaktorkan, maka semakin lama waktu yang dibutuhkan. Jadi semakin besar bilangan yang difaktorkan, semakin sulit pemfaktorannya, semakin kuat pula algoritma RSA.



Gambar 2.1. Skema Algoritma Assimetri

2.3. ASCII System

Plain teks yang akan dienkripsi dengan RSA Coding merupakan angka-angka, sedangkan pesan yang dikirimkan bisaanya berbentuk teks atau tulisan. Sehingga dibutuhkan suatu kode yang sifatnya universal untuk mengubah pesan teks menjadi plain teks dalam bentuk bilangan.

ASCII (American Standard Code for Information Interchange) atau Kode Standar Amerika untuk pertukaran informasi merupakan suatu standar internasional dalam kode huruf dan symbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". ASCII selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks.

Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 8 bit. Dimulai dari 0000 0000 hingga 1111 1111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan Desimal (didapat dari nilai decimal bilangan biner 11111111).

Kode ASCII dari karakter yang umum digunakan dalam penulisan data (A, B, C, …., Z dan 0, 1, 2, 3, …., 9) adalah 32 sampai dengan 126, seperti diberikan pada tabel berikut :

Tabel 2.1. Tabel ASCII

Karakter	Kode ASCII
Spasi	32
!	33
"	34
#	35
$	36
%	37
&	38

Karakter	Kode ASCII
@	64
A	65
B	66
C	67
D	68
E	69
F	70

Karakter	Kode ASCII
`	96
a	97
b	98
c	99
d	100
e	101
f	102

'	39	G	71	g	103
(	40	H	72	h	104
)	41	I	73	i	105
*	42	J	74	j	106
+	43	K	75	k	107
,	44	L	76	l	108
-	45	M	77	m	109
.	46	N	78	n	110
/	47	O	79	o	111
0	48	P	80	p	112
1	49	Q	81	q	113
2	50	R	82	r	114
3	51	S	83	s	115
3	51	S	83	s	115
4	52	T	84	t	116
5	53	U	85	u	117
6	54	V	86	v	118
7	55	W	87	w	119
8	56	X	88	x	120
9	57	Y	89	y	121
:	58	Z	90	z	122
;	59	[	91	{	123
<	60	\	92	|	124
=	61	]	93	}	125
>	62	^	94	~	126
?	63	_	95		

2.4. Pembagian Bilangan Bulat

Jika sebuah bilangan bulat dibagi dengan bilangan bulat lain yang tidak nol, maka hasil baginya tidak selalu bilangan bulat. Contohnya 28 / 7=4 adalah bilangan bulat, tetapi 33 / 4=8,25 bukan bilangan bulat.

Definisi 2.1. [5]

Jika *a* dan *b* bilangan bulat dengan $a \neq 0$, maka dikatakan bahwa *a* membagi *b* jika terdapat bilangan bulat c sehingga $b = ac$. Jika *a* membagi *b*, dapat dikatakan juga bahwa *a* adalah *pembagi* atau *faktor* dari *b*.

Jika *a* membagi *b* dapat dituliskan dengan $a|b$, dan jika *a* tidak membagi *b* dapat dituliskan dengan $a \nmid b$.

Contoh 2.1.

Misal diberikan dua bilangan bulat $a = 13$ dan $b = 182$, maka dapat dikatakan $a|b$ karena terdapat bilangan bulat $c = 14$ dimana $b = ac$ atau $182 = 14 . 13$. Karena $13|182$ maka dapat dikatakan bahwa 13 merupakan pembagi atau faktor dari 182. Sedangkan 17 bukan pembagi dari 182, karena tidak terdapat bilangan bulat *c* sedemikian sehingga $182 = 17c$, sehingga dapat juga dituliskan $17 \nmid 182$.

Teorema 2.1. [5]

Jika *a*, *b*, dan *c* adalah bilangan bulat dengan $a|b$ dan $b|c$, maka $a|c$.

Bukti :

Misalkan $a|b$ dan $b|c$, maka terdapat bilangan bulat *e* dan *f* sehingga $ae = b$ dan $bf = c$. Oleh karena itu, $c = bf = (ae)f = a(ef)$. Karena *e* dan *f* bilangan bulat, dan mengingat perkalian dua bilangan bulat menghasilkan bilangan bulat,

maka ef juga bilangan bulat. Dengan demikian keran $c = a(ef)$, dengan ef bilangan bulat, maka $a|c$. ■

Contoh 2.2.

Diambil $a = 6$, $b = 48$, dan $c = 288$.

Dapat dilihat $6|48$ *dan* $48|288$, maka akan terdapat bilangan bulat $e = 8$ dan $f = 6$ sehingga $ae = b$ dan $bf = c$.

Oleh karena itu $288 = bf = 48 . 6 = (ae)f = (6 . 8)\ 6 = a\ (ef) = 6\ (8 . 6)$

Maka, $288 = 6 . 48$. Jadi dapat dikatakan $6|288$.

Teorema 2.2. [5]

Misalkan *a*, *b*, *c*, *m* dan *n* adalah bilangan bulat dengan $c \neq 0$. Jika $c|a$ dan $c|b$, maka $c|(ma + nb)$.

Bukti :

Misalkan $c|a$ dan $c|b$, maka terdapat bilangan bulat *e* dan *f* sehingga $a = ce$ dan $b = cf$. Oleh karena itu, $ma + nb = mce + ncf = c(me + nf)$. Dengan argumen serupa seperti pada pembuktian teorema sebelumnya, yaitu perkalian 2 bilangan bulat akan menghasilkan bilangan bulat lagi dan mengingat hasil penjumlahan 2 bilangan bulat juga merupakan bilangan bulat akibatnya $c|(ma + nb)$. ■

Contoh 2.3.

Diambil $a = 28, b = 56, c = 7, m = 5$ dan $n = 9$.

Karena $7|28$ dan $7|56$, maka dapat didapat $7|(5.28 + 9.56)$ atau $7|644$

Teorema 2.3

Jika *a*, dan *b* adalah bilangan bulat positif, maka terdapat bilangan bulat x dan y sedemikian sehingga $ax + by = gcd(a, b)$.

Bukti :

Misalkan terdapat himpunan $K = \{am + bn \,|a, b \in Z\}$, maka terdapat k bilangan bulat terkecil dari elemen K. Karena k merupakan elemen dari himpunan K maka dapat ditulis $k = ax + by$. Berdasarkan algoritma pembagian maka terdapat bilangan bulat q dan r dengan $0 \leq r < k$ sedemikian sehingga $a = qk + r$. Maka $r = a - qk = a - q(ax + by) = a(1 - qx) + b(-qy)$. Jika r positif $(0 < r < k)$ maka akan bertentangan dengan k merupakan elemen terkecil dari K. Maka dari itu $r = 0$ sehingga $a = qk$, maka $k|a$.

Dengan cara yang sama akan didapatkan $k|b$. Karena $k|a$ dan $k|b$ maka k merupakan gcd dari a dan b. ■

Lemma 2.1. [5]

Jika *a*, *b*, dan *c* adalah bilangan bulat positif dimana $gcd(a, b) = 1$ dan $a|bc$, maka $a|c$.

Bukti :

Diambil *a* dan *b* bilangan bulat yang relatif prima atau $gcd(a,b) = 1$, berdasarkan Identitas Bezout dimana $gcd(a,b) = ax + by$, maka $ax + by = 1$.

Kedua ruas dikalikan dengan *c*, maka didapat $c = acx + bcy$.

$c = acx + (bc)y = acx + (ak)y = a(xk + ky)$.

Dengan argumen serupa seperti pada pembuktian teorema sebelumnya, yaitu perkalian 2 bilangan bulat akan menghasilkan bilangan bulat lagi dan mengingat hasil penjumlahan 2 bilangan bulat juga merupakan bilangan bulat akibatnya $a|c$

■

Contoh 2.4.

Diambil $a = 121$, $b = 38$ dan $c = 242$ dimana $gcd(a,b) = 1$ dan $a|bc$.

Dengan algoritma euclid dapat diketahui gcd dari *a* dan *b*

$121 = 38 . 3 + 7$

$38 = 7 . 5 + 3$

$7 = 3 . 2 + 1$

Maka diketahui bahwa $gcd(121,38) = 1$.

Karena $gcd(121,38) = 1$ dan $121|38.242$ maka $121|242$

2.5. Aritmetika Modulo

Dalam penerapan Teorema Euler pada perumusan algoritma RSA Coding sangat dibutuhkan pemahaman tentang modulo.

Modulo sendiri berarti sisa hasil bagi. Misalkan *a* adalah bilangan bulat dan *m* adalah bilangan bulat dimana *a* dan *m* lebih besar dari 0. Maka operasi *a mod m* (dibaca "*a* modulo *m*") memberikan sisa jika *a* dibagi dengan *m*.

Bilangan *m* disebut modulus atau modulo, dan hasil modulo *m* terletak di dalam himpunan {0, 1, 2, …, *m*–1}

Contoh 2.5.

Diambil $a = 20$ dan $m = 6$. Karena 20 dibagi 6 adalah 3 bersisa 2, maka diperoleh $a \bmod m \equiv 20 \bmod 6 = 2$.

Didalam aritmetika modulo dikenal isitilah kongruen, dimana dua buah bilangan bulat yang lebih besar dari 0 dikatakan kongruen modulo *m* jika dan hanya jika sisanya pada pembagian oleh *m* adalah sama.

Definisi 2.2. [5]

Misalkan *m* bilangan bulat positif. Jika *a* dan *b* bilangan bulat, maka dapat dikatakan *a* kongruen *b* modulo *m* jika $m|(a-b)$.

Jika *a* kongruen *b* modulo *m*, dapat ditulis $a \equiv b \pmod m$. Jika $m \nmid (a-b)$, dapat ditulis $a \not\equiv b \pmod m$ atau *a* tidak kongruen *b* modulo *m*.

Contoh 2.6.

Diketahui $a = 32$, $b = 14$ dan $m = 6$.

Maka, $32 \equiv 14 \pmod 6$, karena $6|(32-14)$ atau $6|18$.

Teorema 2.4. [5]

Jika *a* dan *b* bilangan bulat, maka $a \equiv b \pmod{m}$ jika dan hanya jika terdapat bilangan bulat *k* sedemikian sehingga $a = b + km$.

Bukti :

Misalkan $a \equiv b \pmod{m}$, maka $m|(a-b)$. Hal ini berarti terdapat bilangan bulat *k* dengan $km = a-b$, jadi $a = b + km$.

Sebaliknya, jika terdapat bilangan bulat *k* dengan $a = b + km$, maka $km = a - b$. Oleh karena itu $m|(a-b)$, sehingga $a \equiv b \pmod{m}$. ■

Contoh 2.7.

Diambil $a = 17$, $b = 2$ dan $m = 3$.

Karena terdapat bilangan $k = 5$ sedemikian sehingga $17 = 2 + 3.5$, maka $17 \equiv 2 \pmod{3}$

Teorema 2.5. [5]

Misalkan *m* bilangan bulat positif. Kekongruenan modulo *m* memenuhi beberapa sifat berikut :

(i) Sifat Refleksi, yaitu jika *a* bilangan bulat, maka $a \equiv a \pmod{m}$

(ii) Sifat Simetris, yaitu jika *a* dan *b* bilangan bulat dimana $a \equiv b \pmod{m}$, maka $b \equiv a \pmod{m}$.

(iii) Sifat Transitif, yaitu jika *a*, *b*, dan *c* bilangan bulat dengan $a \equiv b \pmod{m}$ dan $b \equiv c \pmod{m}$, maka $a \equiv c \pmod{m}$.

Bukti :

(i) Dapat dilihat bahwa $a \equiv a \pmod{m}$, karena terdapat bilangan bulat $k = 0$ sedemikian sehingga $a = a + 0 \,.m$.

(ii) Jika $a \equiv b \pmod{m}$, maka $m|(a-b)$. Oleh karena itu terdapat bilangan bulat *k* dengan $km = a-b$. Hal ini menunjukkan bahwa $(-k)m = -(a-b)$, jadi $m|(b-a)$. Sebagai konsekuensinya, $b \equiv a \pmod{m}$.

(iii) Jika $a \equiv b \pmod{m}$ dan $b \equiv c \pmod{m}$, maka $m|(a-b)$ dan $m|(b-c)$. Oleh karena itu, terdapat bilangan bulat *k* dan *l* dengan $km = a-b$ dan $lm = b-c$.

Oleh karena itu, $a-c = (a-b) + (b-c) = km + lm = (k+l)m$.

$a-c = (k+l)m$, jadi $m|(a-c)$ sehingga $a \equiv c \pmod{m}$. ■

Teorema 2.6. [5]

Jika *a, b, c* dan *m* adalah bilangan bulat dengan $m \geq 0$ dan $a \equiv b \pmod{m}$, maka :

(i) $a + c \equiv b + c \pmod{m}$,

(ii) $a - c \equiv b - c \pmod{m}$,

(iii) $ac \equiv bc \pmod{m}$

Bukti :

Dengan $a \equiv b \pmod{m}$, dapat diketahui bahwa $m|(a-b)$.

(i) Karena $(a+c)-(b+c) = a-b$, maka $m|((a+c)-(b+c))$, jadi $a + c \equiv b + c \pmod{m}$.

(ii) Demikian juga persamaan (ii). Karena $(a - c)-(b - c) = a-b$, maka $m|((a + c)-(b + c))$, jadi $a - c \equiv b - c \pmod{m}$

(iii) Untuk menunjukkan bahwa persamaan (iii) benar, dilihat bahwa $ac-bc = c(a-b)$, Karena $m|(a-b)$, maka $m|c(a-b)$.

Jadi $ac \equiv bc \pmod{m}$. ■

Contoh 2.8.

Diambil bilangan $a = 60$, $b = 4$, $c = 3$ dan $m = 8$ sehingga $60 \equiv 4 \pmod{8}$.

(i) $a + c \equiv b + c \pmod{m}$

$60 + 3 \equiv 4 + 3 \pmod{8}$

$63 \equiv 7 \pmod{8}$

(ii) $a - c \equiv b - c \pmod{m}$

$60-3 \equiv 4 - 3 \pmod{8}$

$57 \equiv 1 \pmod{8}$

(iii) $ac \equiv bc \pmod{m}$

$60 .3 \equiv 4 .3 \pmod{8}$

$180 \equiv 12 \pmod{8}$

Teorema 2.7. [5]

Jika *a*, *b*, *c* dan *m* adalah bilangan bulat dimana $m > 0$, $d = gcd(c, m)$ dan $ac \equiv bc \pmod{m}$, maka $a \equiv b \pmod{m/d}$

Bukti :

Misalkan $ac \equiv bc \pmod{m}$, dapat diketahui bahwa $m|(ac-bc) \equiv m|c\,(a-b)$. Oleh karena itu, terdapat bilangan bulat k dengan $c\,(a-b) = km$. Dengan membagi kedua sisi dengan d, didapatkan $\frac{c}{d}\,(a-b) = \frac{m}{d}\,k$. Berdasarkan Lemma 2.1 jika $gcd\left(\frac{m}{d}, \frac{c}{d}\right) = 1$, maka $m/d\,|(a-b)$. Sehingga, $a \equiv b \pmod{m/d}$. ■

Contoh 2.9.

Diambil $a = 5$, $b = 2$, $c = 10$, $m = 15$.

Selama $50 \equiv 20 \pmod{15}$ dan $d = gcd(c,m) = gcd(10,15) = 5$. Dapat dilihat bahwa $50/10 \equiv 20/10 \pmod{15/5}$ atau $5 \equiv 2 \pmod{3}$.

Akibat 2.1.

Jika a, b, c dan m adalah bilangan bulat, $m > 0$, $gcd(c,m) = 1$, dan $ac \equiv bc \pmod{m}$, maka $a \equiv b \pmod{m}$

Contoh 2.10.

Diambil $a = 6$, $b = 1$, $c = 7$, $m = 5$.

Jika $42 \equiv 7 \pmod{5}$ dan $d = gcd(c,m) = gcd(7,5) = 1$. Dapat dilihat bahwa $42/7 \equiv 7/7 \pmod{5/1}$ atau $6 \equiv 1 \pmod{5}$.

Teorema 2.8. [5]

Jika *a, b, c, d* dan *m* adalah bilangan bulat dengan $m > 0$, $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka :

(i) $a + c \equiv b + d \pmod{m}$,

(ii) $a - c \equiv b - d \pmod{m}$,

(iii) $ac \equiv bd \pmod{m}$

Bukti :

Dengan $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka diketahui bahwa $m|(a - b)$ dan $m|(c - d)$. Oleh karena itu terdapat bilangan bulat *k* dan *l* dengan $km = a - b$ dan $lm = c - d$.

(i) Untuk membuktikan persamaan (i), dicatat bahwa

$(a + c) - (b + d) = (a - b) + (c - d) = km + lm = (k + l)m.$

$(a + c) - (b + d) = (k + l)m.$

Oleh karena itu, $m|[(a + c) - (b + d)]$, maka $a + c \equiv b + d \pmod{m}$.

(ii) Untuk membuktikan persamaan (ii), dicatat bahwa

$(a - c) - (b - d) = (a - b) - (c - d) = km - lm = (k - l)m.$

$(a - c) - (b - d) = (k - l)m.$

Oleh karena itu, $m|[(a - c) - (b - d)]$, maka $a - c \equiv b - d \pmod{m}$.

(iii) Untuk membuktikan persamaan (iii), dicatat bahwa

$ac - bd = ac - bc + bc - bd = c(a - b) + b(c - d)$

$ac - bd = ckm + clm = m(ck + bl)$

$ac - bd = m(ck + bl)$

Oleh karena itu, $m|(ac - bd)$, maka $ac \equiv bd \ (mod\ m)$. ■

Contoh 2.11.

Diambil $a = 17$, $b = 5$, $c = 27$, $d = 7$ dan $m = 4$.

Diketahui $17 \equiv 5 (\mathrm{mod}\ 4)$ dan $27 \equiv 7 (\mathrm{mod}\ 4)$.

Maka :

(i) $(17 + 27) \equiv (5 + 7)(mod\ 4)$

$44 \equiv 12 (mod\ 4)$

(ii) $(17 - 27) \equiv (5 - 7)(mod\ 4)$

$(-10) \equiv (-2)(mod\ 4)$

(iii) $17.27 \equiv 5.7 (mod\ 4)$

$459 \equiv 35 (mod\ 4)$

Teorema 2.9. [5]

Jika *a*, *b*, *k* dan *m* adalah bilangan bulat dengan $k \geq 0$, $m \geq 0$ dan $a \equiv b \ (mod\ m)$, maka $a^k \equiv b^k \ (mod\ m)$.

Bukti :

Karena $a \equiv b \ (mod\ m)$, didapatkan $m|(a-b)$. Selanjutnya karena, $a^k - b^k = (a - b)(a^{(k-1)} + \cdots + ab^{(k-2)} + b^{(k-1)})$, maka $(a-b)|(a^k-b^k)$.

Oleh karena itu, berdasarkan Teorema 2.1 karena $m|(a-b)$ dan $(a-b)|(a^k-b^k)$ maka $m|(a^k-b^k)$. Karenanya, $a^k \equiv b^k \ (mod\ m)$. ■

Contoh 2.12.

Diambil $a = 14$, $b = 2$, $k = 4$, dan $m = 3$.

Diketahui $14 \equiv 2 (\mathrm{mod}\ 3)$, maka $14^4 = 38416 \equiv 2^4 = 16 (\mathrm{mod}\ 3)$

2.6. Algoritma Euclid

Dalam RSA Coding, kunci publik *e* merupakan bilangan relatif prima terhadap $\varphi(n)$ yang didapat dari $n = pq$, dengan *p* dan *q* merupakan bilangan prima besar yang dipilih secara acak. Kemudian *e* dikatakan relatif prima terhadap $\varphi(n)$ jika $gcd(e, \varphi(n)) = 1$.

Nilai $gcd(a, b)$ dengan $a > b > 0$ dapat ditentukan dengan Algoritma Euclid, sebagaimana diberikan oleh teorema berikut.

Teorema 2.10. [5]

Misalkan $r_0 = a$ dan $r_1 = b$ adalah bilangan bulat dimana $a \geq b > 0$. Dengan menerapkan algoritma pembagian untuk mendapatkan $r_j = r_{j+1} q_{j+1} + r_{j+2}$ dengan $0 < r_{j+2} < r_{j+1}$ untuk $j = 0, 1, 2, 3, \ldots, n-2$ dan $r_{n+1} = 0$, maka $gcd(a, b) = r_n$ sisa terakhir yang tidak 0. ■

Contoh 2.13.

Tentukan $(1248, 76)$:

$1248 = 16 . 76 + 32$

$76 = 2.32 + 12$

$32 = 2.12 + 8$

$12 = 1.8 + 4$

$8 = 2.4 + 0$

Karena 4 merupakan sisa terakhir yang tidak 0, maka $gcd(1248,76) = 4$.

2.7. Sistem Residu

Perumusan RSA Coding didasarkan pada teorema euler, sedangkan pada pembuktian teorema euler terdapat penggunaan sistem residu. Oleh karena itu sebelum membahas tentang teorema euler akan dibahas terlebih dahulu tentang sistem residu.

1. Sistem Residu Lengkap Modulo n

Definisi 2.3. [5]

Sistem residu lengkap modulo *n* adalah himpunan bilangan bulat yang setiap bilangan dalam himpunan tersebut kongruen modulo n terhadap tepat satu bilangan dalam himpunan bilangan tersebut.

Contoh 2.14.

Misalkan diberikan $n = 5$, maka himpunan yang merupakan sistem residu lengkap modulo 5 adalah himpunan bilangan 0, 1, 2, 3, 4. Karena setiap bilangan dalam himpunan tersebut kongruen modulo 5 terhadap tepat satu bilangan dalam himpunan itu sendiri.

Teorema 2.11. [5]

Jika $r_1, r_2, ..., r_m$ merupakan sistem residu lengkap modulo *m*, dan *a* adalah bilangan bulat positif dengan $gcd(a, m) = 1$, maka $ar_1 + b, ar_2 + b, \ldots, ar_m + b$ merupakan sistem residu lengkap modulo *m* untuk setiap bilangan bulat *b*.

Bukti :

Misal diberikan $ar_j + b \equiv ar_k + b \pmod{m}$, maka berdasarkan teorema 2.6 maka $ar_j \equiv$. Berdasarkan akibat 2.1 jika $ar_j \equiv$, maka $r_j \equiv$. Oleh karena $r_j \not\equiv r_k \pmod{m}$ jika $r_j \neq r_k$, maka dapat disimpulkan $r_j = r_k$. Hal ini merupakan kontradiksi dari himpunan bahwa r_j tidak boleh sama dengan r_k.

Dari uraian diatas dapat dilihat tidak ada dua bilangan dari himpunan $ar_1 + b, ar_2 + b, \ldots, ar_m + b$ yang kongruen modulo *m*. ■

2. Sistem Residu Tereduksi Modulo n

Definisi 2.3. [5]

Sistem residu tereduksi modulo *n* adalah himpunan bilangan bulat dari $\varphi(n)$ dimana setiap bilangan tersebut relatif prima terhadap *n* dan tidak ada dua bilangan bulat yang mempunyai kelas sisa yang sama.

Contoh 2.15.

Misalkan diberikan $n = 9$, maka himpunan yang merupakan sistem residu tereduksi modulo 9 adalah himpunan bilangan 1, 2, 4, 5, 7, 8. Karena setiap bilangan dalam himpunan tersebut relatif prima terhadap 9 dan tidak ada dua bilangan bulat yang mempunyai kelas sisa yang sama.

Teorema 2.12. [5]

Jika $r_1, r_2, \ldots, r_{\varphi(m)}$ merupakan sistem residu tereduksi modulo *m*, dan jika *a* adalah bilangan bulat positif dengan $gcd(a, m) = 1$, maka $ar_1, ar_2, \ldots, ar_{\varphi(m)}$ juga merupakan sistem residu tereduksi modulo *m* untuk setiap bilangan bulat.

Bukti :

Untuk membuktikan teorema 2.12 akan ditunjukkan bahwa setiap bilangan bulat ar_j relatif prima terhadap *m* dan tidak terdapat dua ar_j kongruen modulo *m.*

Untuk menunjukkan bahwa setiap bilangan bulat ar_j relatif prima terhadap *m* diasumsikan bahwa $gcd(ar_j, m) > 1$. Kemudian didapatkan pembagi prima *p* dari nilai $gcd(ar_j, m)$. Karena itu juga, didapatkan $p|a$ atau $p|r_j$. Seperti itu juga didapatkan $p|a$ dan $p|m$ atau $p|r_j$ dan $p|m$. Karena r_j adalah anggota dari sistem residu tereduksi modulo *m*

maka bagaimanapun tidak bisa didapatkan $p|r_j$ dan $p|m$. Karena $gcd(a,m) = 1$ maka juga tidak bisa didapatkan $p|a$ dan $p|m$. Oleh karena itu dapat disimpulkan bahwa ar_j dan *m* relatif prima untuk $j = 1, 2, \ldots, \varphi(m)$.

Kemudian untuk menunjukkan bahwa tidak terdapat dua ar_j yang kongruen modulo *m* diasumsikan bahwa $ar_j \equiv ar_k \pmod m$, dimana *j* dan *k* bilangan bulat positif yang berbeda dengan $1 \leq j \leq \varphi(m)$ dan $1 \leq k \leq \varphi(m)$. Karena $gcd(a,m) = 1$, berdasarkan akibat 2.1. dapat dilihat bahwa $r_j \equiv r_k \pmod m$. Hal ini merupakan kontradiksi, karena r_j dan r_k berasal dari satu sistem residu tereduksi modulo *m*, jadi $r_j \not\equiv r_k \pmod m$. ■

Contoh 2.16.

Misalkan diberikan $n = 9$ dan $a = 4$ dengan $gcd(9,4) = 1$, dapat diketahui himpunan bilangan 1, 2, 4, 5, 7, 8 merupakan sistem residu tereduksi modulo 9. Oleh karena itu himpunan bilangan 4, 8, 16, 20, 28, 32 juga merupakan sistem residu tereduksi modulo 9. Karena setiap bilangan dalam himpunan tersebut relatif prima terhadap 9 dan tidak ada dua bilangan bulat yang mempunyai kelas sisa yang sama.

2.8. Fungsi Phi Euler (φ)

Dalam RSA Coding, setelah dipilih dua bilangan prima besar *p* dan *q* kemudian akan dihitung nilai $n = pq$. Untuk mendapatkan nilai totient dari *n* digunakan fungsi phi euler (φ).

Definisi 2.4. [5]

Fungsi phi euler (φ) didefinisikan sebagai fungsi yang menyatakan banyaknya bilangan bulat positif yang lebih kecil dari sebuah bilangan bulat dan relatif prima terhadap bilangan bulat tersebut.

Jadi jika terdapat bilangan bulat positif *n*, maka nilai $\varphi(n)$ adalah banyaknya bilangan bulat positif yang lebih kecil dari *n* dan relatif prima terhadap *n*.

Contoh 2.17.

Misalkan diberikan $n = 10$, maka $\varphi(n) = 4$, karena bilangan yang lebih kecil dari 10 dan relatif prima terhadap 10 hanya 1, 3, 7 dan 9.

Teorema 2.13. [5]

Jika *n* prima, maka $\varphi(n) = n-1$. Sebaliknya, jika $\varphi(n) = n-1$ maka *n* prima.

Bukti :

Jika *p* adalah bilangan prima berdasarkan sifat bilangan prima yang hanya habis dibagi bilangan 1 atau bilangan *p* sendiri, maka jika dibagi dengan bilangan yang lebih kecil dari *p* sisa terakhir yang tidak nol adalah 1 oleh karena itu setiap bilangan bulat positif yang lebih kecil dari *p* adalah relatif prima terhadap *p*. Dapat dikatakan juga bahwa $\varphi(p) = p-1$.

Sebaliknya jika *p* adalah bilangan komposit dan *p* memiliki sebuah biangan pembagi *d* dengan $1 < d < p$ dan *d* tidak relatif prima terhadap *p*. Jika didapat salah satu bilangan yang lebih kecil dari p adalah d yang tidak telatif prima terhadap p maka $\varphi(p) \leq p-2$. Oleh karena itu, jika $\varphi(p) = p-1$, maka *p* harus merupakan bilangan prima ■

Contoh 2.18.

Misal diberikan bilangan prima $p = 5$, tentukan $\varphi(5)$.

Karena 5 adalah bilangan prima, maka $\varphi(5) = 5 - 1 = 4$

Teorema 2.14. [5]

Jika *m* dan *n* adalah bilangan bulat positif yang relatif prima, maka $\varphi(mn) = \varphi(m)\ \varphi(n)$.

Bukti :

Untuk membuktikan teorema 2.14 akan ditampilkan bilangan bulat positif kurang dari sama dengan *mn* dalam bentuk berikut :

$$\begin{array}{ccccc} 1 & m+1 & 2m+1 & \cdots & (n-1)m+1 \\ 2 & m+2 & 2m+2 & \cdots & (n-1)m+2 \\ 3 & m+3 & 2m+3 & \cdots & (n-1)m+3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r & m+r & 2m+r & \cdots & (n-1)m+r \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m & 2m & 3m & \cdots & mn \end{array}$$

Kemudian asumsikan *r* adalah bilangan bulat positif kurang dari sama dengan *m*. Asumsikan juga bahwa $gcd(m,r) = d > 1$. Maka tidak ada bilangan pada baris ke-*r* yang relatif prima terhadap *mn*, karena setiap elemen pada baris ini dari bentuk $km + r$, dimana *k* bilangan bulat dengan $1 \leq k \leq n-1$ dan $d|(km+r)$, karena $d|m$ dan $d|r$.

Sebagai konsekuensi, untuk mencari bilangan dari bentuk diatas yang relatif prima terhadap mn, jika $gcd(m,r) = 1$ maka hanya perlu melihat baris ke-*r* saja. Jika $gcd(m,r) = 1$ dan $1 \leq r \leq m$, harus ditentukan berapa banyak bilangan dalam

baris tersebut yang relatif prima terhadap mn. Elemen dari baris tersebut adalah $r, m + r, 2m + r, \ldots, (n-1)\ m + r$. Jika $gcd(r,m) = 1$, maka setiap bilangan dalam baris tersebut relatif prima terhadap *mn*. Berdasarkan teorema 2.12, *n* bilangan dari baris ke-*r* adalah bentuk sistem residu lengkap modulo *n*. Oleh karena itu, sebenarnya bilangan $\varphi(n)$ relatif prima terhadap *n*. Jika bilangan $\varphi(n)$ juga relatif prima terhadap *m*, maka juga relatif prima terhadap mn.

Jika terdapat $\varphi(m)$ baris, dimana setiap baris mengandung $\varphi(n)$ bilangan yang relatif prima terhadap mn. Dapat disimpulkan $\varphi(mn) = \varphi(m)\ \varphi(n)$. ■

BAB III

PENERAPAN TEOREMA EULER PADA KRIPTOGRAFI RSA

3.1. Perumusan Algoritma RSA Coding

RSA Coding merupakan salah satu algoritma Assimetris Kriptografi, algoritma lainnya yaitu : DSA, DH, ECC dan lain sebagainya. RSA Coding dikembangkan oleh *Ron Rivest*, *Adi Shamir*, dan *Len Adleman*.

Perumusan algoritma RSA Coding didasarkan pada penggunaan Teorema Euler, sehingga dapat menghasilkan rumus enkripsi dan dekripsi yang saling berkaitan.

Besaran-besaran yang digunakan pada RSA Coding antara lain :

1. p dan q bilangan prima (rahasia)
2. $n = pq$ (tidak rahasia)
3. $\varphi(n) = (p-1)(q-1)$ (rahasia)
4. e (kunci enkripsi) (tidak rahasia)
5. d (kunci dekripsi) (rahasia)
6. m (plainteks) (rahasia)
7. c (cipherteks) (tidak rahasia)

Sebelum pembahasan tentang algoritma RSA coding akan dijelaskan tentang teorema euler terlebih dahulu.

Teorema 3.1. [5]

Jika m adalah bilangan bulat positif dan a adalah bilangan bulat dengan $gcd(a,m) = 1$, maka $a^{\varphi(m)} \equiv 1(mod\ m)$.

Bukti :

Misalkan $r_1, r_2, \dots, r_{\varphi(m)}$ merupakan sistem residu tereduksi yang terbentuk dari bilangan bulat positif yang tidak melebihi *m* relatif prima terhatap *m*. Berdasarkan teorema 2.12, selama $gcd(a,m) = 1$, maka barisan $ar_1, ar_2, \dots, ar_{\varphi(m)}$ juga merupakan sistem residu tereduksi modulo *m*. Oleh karena itu, sisa bagi positif yang paling kecil dari $ar_1, ar_2, \dots, ar_{\varphi(m)}$ pasti bilangan bulat $r_1, r_2, \dots, r_{\varphi(m)}$ dalam masalah yang sama. Konsekuensinya, jika dikalikan bersamaan semua bilangan dalam tiap-tiap sistem residu tereduksi tersebut, didapatkan

$$ar_1 ar_2 \dots ar_{\varphi(m)} \equiv r_1 r_2 \dots r_{\varphi(m)} \ (mod\ m)$$

Dengan begitu,

$$a^{\varphi(m)} r_1 r_2 \dots r_{\varphi(m)} \equiv r_1 r_2 \dots r_{\varphi(m)} \ (mod\ m)$$

Selama $gcd(r_1 r_2 \dots r_{\varphi(m)}, m) = 1$, berdasarkan akibat 2.1, dapat disimpulkan bahwa $a^{\varphi(m)} \equiv 1 \ (mod\ m)$ ■

Contoh 3.1.

Misalkan $m = 20$ dan $a = 3$ dengan $gcd(20,3) = 1$.

Dengan $m = 20$ didapat $\varphi(m) = 8$, maka berdasarkan teorema euler

$$a^{\varphi(m)} \equiv 1 \ (mod\ m)$$

$$3^8 \equiv 1 \ (mod\ 20)$$

$$6561 \equiv 1 \ (mod\ 20)$$

$$6561 \ mod \ 20 = 1$$

Setelah dipahami teorema euler maka akan dijabarkan tentang penerapan teorema euler dalam pembentukan rumus enkripsi dan dekripsi dari algoritma RSA Coding.

Dari teorema euler didapatkan $a^{\varphi(n)} \equiv 1 \ (mod \ n)$. Dimana *a* harus relatif prima terhadap *n*. Digunakan notasi *m* untuk menggantikan *a*, dimana *m* adalah plain teks.

$$m^{\varphi(n)} \equiv 1 \ (mod \ n) \qquad (3.1)$$

Berdasarkan Teorema 2.9 dimana $a^k \equiv b^k \ (mod \ n)$ untuk *k* bilangan bulat > 0, maka persamaan (3.1) menjadi :

$$m^{k\varphi(n)} \equiv 1^k \ (mod \ n)$$

atau,

$$m^{k\varphi(n)} \equiv 1 \ (mod \ n) \qquad (3.2)$$

Berdasarkan Teorema 2.6 dimana $ac \equiv bc \ (mod \ n)$, maka jika persamaan (3.2) dikali dengan *m* menjadi :

$$m^{k\varphi(n)+1} \equiv m \ (mod \ n) \qquad (3.3)$$

Misalkan *e* dan *d* (nantinya sebagai kunci enkripsi dan dekripsi) yang telah dipilih sedemikian sehingga :

$$e.d \equiv 1 \ (mod \ \varphi(n)) \text{ atau } e.d \equiv k\varphi(n) + 1 \qquad (3.4)$$

Persamaan (3.4) ini nantinya digunakan dalam pembangkitan pasangan kunci.

Persamaan (3.4) disubtitusikan ke dalam persamaan (3.3), menjadi :

$$m^{e.d} \equiv m \ (mod\ n) \tag{3.5}$$

Persamaan (3.5) dapat ditulis kembali menjadi :

$$(m^e)^d \equiv m \ (mod\ n) \tag{3.6}$$

Yang artinya, perpangkatan *m* dengan *e* diikuti dengan perpangkatan dengan *d* menghasilkan kembali *m* semula.

Berdasarkan persamaan (3.6), maka enkripsi dan dekripsi diberikan oleh :

enkripsi (*m*) = $c \equiv m^e \ mod\ n$

dekripsi (*c*) = $m \equiv c^d \ mod\ n$

3.2. Pembangkitan Pasangan Kunci

Sebagai algoritma Asimetris Kriptografi, RSA Coding membutuhkan dua kunci yang berbeda untuk enkripsi dan dekripsi. Bilangan yang dipilih sebagai kunci adalah bilangan prima yang besar, dengan alasan pemfaktoran sebuah bilangan hasil perkalian dari dua bilangan prima yang besar menjadi dua bilangan prima yang sesuai akan sangat sulit. Sehingga keamanan dari RSA Coding dapat terjamin.

Dalam kaitannya dengan pemanfaatan bilangan prima akan dijelaskan terlebih dahulu tentang Teorema Fermat.

Teorema 3.2. [5]

Jika *p* adalah bilangan prima dan *a* adalah bilangan bulat dimana *p* tidak membagi *a*, maka $a^{p-1} \equiv 1 \ (mod\ p)$.

Bukti :

Misalkan *p* adalah bilangan prima dan *a* adalah bilangan bulat positif sedemikian sehingga $p \nmid a$.

Asumsikan terdapat barisan berikut $a, 2a, 3a, 4a, \ldots, (p-1)a$ didapat dari ka, dengan $k = 1, 2, \ldots, p-1$.

Perhatikan bahwa tidak ada bilangan dari barisan di atas yang habis dibagi *p*. Andaikan $p|ja$, berdasarkan Lemma 2.1 maka $p|j$ selama $p \nmid a$. Maka hal ini tidak mungkin karena $1 \leq j \leq p-1$. Kemudian, dari barisan itu tidak ada dua bilangan yang kongruen modulo *p*. Untuk melihat hal tersebut andaikan bahwa $ja \equiv ka \pmod{p}$ dimana $1 \leq j \leq k \leq p-1$. Berdasarkan Akibat 2.1 dengan $gcd(a, p) = 1$ didapatkan $j \equiv k \pmod{p}$. Hal ini tidak mungkin karena *j* dan *k* adalah bilangan bulat positif lebih kecil dari $p-1$ sehingga *j* dan *k* tidak mungkin memiliki kelas sisa yang sama dalam modulo *p*.

Karena barisan $a, 2a, 3a, 4a, \ldots, (p-1)a$ dimana dari barisan itu tidak ada dua bilangan yang kongruen modulo *p* diketahui jika bilangan-bilangan tersebut dibagi dengan *p*, maka sisa pembagiannya akan selalu berbeda satu sama lain. Sehingga,

$$a.\,2a.3a \ldots .(p-1)a \equiv 1.2.3 \ldots .(p-1) \pmod{p}$$

$$a^{p-1}\,(p-1)! \equiv (p-1)! \pmod{p}$$

Karena $gcd((p-1)!, p) = 1$, dengan akibat 2.1 maka :

$$a^{p-1} \equiv \pmod{p}$$ ■

Contoh 3.2.

Misalkan $p = 13$ dan $a = 4$, maka : $4^{13-1} \equiv 1 \pmod{13}$

$$4^{12} \equiv 1 \pmod{13}$$

atau $16777216 \equiv 1 \pmod{13}$

Pasangan kunci merupakan suatu komponen penting dalam kriptografi RSA Coding. Untuk membangkitkan kedua kunci dipilih dua bilangan prima acak yang besar. Sehingga terjadi pemfaktoran bilangan yang sangat besar, karena alasan tersebut RSA Coding dianggap aman.

Berikut ini langkah-langkah proses pembangkitan pasangan kunci :

1. Dipilih dua buah bilangan prima sembarang, *p* dan *q*
2. Dihitung nilai $n = pq$
3. Nilai totient dari n yang disimbolkan $\varphi(n)$ dapat dihitung menggunakan Fungsi φ Euler.
4. Dipilih bilangan *e*, dimana e relatif prima terhadap $\varphi(n)$. Bilangan yang relatif prima adalah bilangan yang memiliki gcd sama dengan 1. Hal ini dapat ditentukan dengan menggunakan Algoritma Euclide.
5. Ditentukan bilangan *d* dengan persamaan (3.4).

   Perhatikan $e.d \equiv 1 + k\varphi(n)$, sehingga *d* dapat dihitung dengan

$$d = \frac{1 + k\varphi(n)}{e}$$

dimana $k = 1, 2, 3, \ldots$, sehingga diperoleh nilai *d* yang bulat

dengan demikian kunci umum adalah pasangan (*e*, *n*)

kunci pribadi adalah pasangan (*d*, *n*)

3.3. Proses Enkripsi

Langkah-langkah pada proses enkripsi adalah sebagai berikut :

1. Plain teks diubah ke dalam bentuk bilangan.

   Untuk mengubah plain teks yang berupa huruf menjadi bilangan dapat digunakan kode ASCII dalam sistem bilangan decimal.

2. Plainteks m dinyatakan menjadi blok-blok $m_1, m_2, m_3, \ldots,$ sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang $[0, n-1]$, sehingga transformasinya menjadi satu ke satu.

3. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus

   $c_1 = m_1^{\,e} \bmod n$

3.4. Proses Dekripsi

Langkah-langkah pada proses dekripsi adalah sebagai berikut :

1. Setiap blok chipper teks c_i didekripsi kembali menjadi blok m_i dengan rumus

$m_i = c_i^{\,d} \bmod n$

2. Kemudian blok-blok $m_1, m_2, m_3, \ldots,$ diubah kembali ke bentuk huruf dengan melihat kode ASCII hasil dekripsi.

3.5. Keamanan RSA Coding

Membawa suatu pesan yang sangat penting selain ada pihak yang ingin menjaga agar pesan tetap aman, ada juga ternyata pihak-pihak yang ingin mengetahui pesan rahasia tersebut secara tidak sah. Bahkan ada pihak-pihak yang

ingin agar dapat mengubah isi pesan tersebut. Ilmu untuk mendapatkan pesan yang asli dari pesan yang telah disandikan tanpa memiliki kunci untuk membuka pesan rahasia tersebut disebut kriptoanalisis. Sedangkan usaha untuk membongkar suatu pesan sandi tanpa mendapatkan kunci dengan cara yang sah dikenal dengan istilah serangan (attack).

Ada beberapa interpretasi yang mungkin dari memecahkan RSA Coding. Yang paling merusak adalah seorang penyerang yang menemukan kunci privat yang berkorespondensi dengan sebuah kunci publik yang disebarluaskan. Hal ini akan membuat penyerang bisa membaca semua pesan yang terenksripsi dengan kunci publik dan memalsukan pesan.

Cara yang paling jelas untuk melakukan serangan jenis ini adalah dengan memfaktorkan modulus *n*, ke dalam dua buah faktor prima, *p* dan *q*. Sedemikian sehingga $n = pq$.

Jika *n* berhasil difaktorkan menjadi *p* dan *q*, maka $\varphi(n) = (p-1)(q-1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *e* diumumkan (tidak rahasia), maka kunci dekripsi *d* dapat dihitung dari persamaan $e \,.\, d \equiv 1 \ (mod\ \varphi(n))$.

Dengan kemajuan teknologi, saat ini banyak terdapat software yang dapat mencari faktor prima dari suatu bilangan dengan cepat. Salah satunya adalah software matlab.

Panjang Kunci Aman Untuk RSA Coding

Penemu algoritma RSA menyarankan nilai *p* dan *q* panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = pq$ akan berukuran lebih dari 200 digit.

Menurut Rivest dan kawan-kawan, usaha untuk mencari faktor prima dari bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun, sedangkan untuk bilangan 500 digit membutuhkan waktu 1025 tahun. (Dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

Namun seiring perkembangan teknologi dan peningkatan kecepatan komputer, memunculkan anjuran pemakaian kunci yang lebih panjang. Pada tahun 1992, RSA Coding yang menggunakan kunci 512 digit dapat dipecahkan dalam waktu 7 bulan.

Lebih dianjurkan untuk menggunakan ukuran kunci 1024 bit untuk hal yang berhubungan dengan hukum dan 2048 bit untuk kunci-kunci yang ekstrem, seperti kunci yang digunakan oleh alat untuk verfikasi wewenang. Beberapa standar pada saat ini lebih menganjurkan 1024 bit untuk penggunaan dalam bidang hukum. Informasi yang lebih tidak terlalu penting sudah cukup dienkripsi dengan kunci 768 bit.

3.6. Kekurangan RSA Coding

Meskipun disisi keamanan, RSA Coding sangat baik. Namun RSA Coding juga mempunyai kekurangan, yakni proses enkripsi dan dekripsi yang lama dan juga membutuhkan biaya yang besar. Jadi penggunaan RSA Coding juga harus melihat seberapa nilai data yang dirahasiakan dan berapa lama data tersebut harus dirahasiakan.

Selain itu, keamanan dari RSA Coding sangat bergantung pada panjang kunci. Semakin panjang kunci yang digunakan maka akan semakin aman.

3.7. Implementasi RSA Coding

Selanjutnya akan diberikan beberapa contoh implementasi dari RSA Coding. Plain teks yang sama akan diproses dengan beberapa pasang kunci yang berbeda.

Contoh 3.3.

1. Diberikan plain teks :

   Bersiap di perbatasan, penyerangan dimulai pukul 05:00. Siapkan 3000 pasukan infanteri dan 1000 kendaraan perang. Bantuan udara akan berangkat pukul 10.00, usahakan bertahan sampai bantuan udara datang.

2. Dipilih bilangan prima $p = 523$ dan $q = 857$.

3. Kemudian dihitung nilai *n* hasil perkalian bilangan *p* dan *q*,

   $n = p \,.\, q = 523 \,.\, 857 = 448211$

4. Dicari nilai totient dari *n*

$\varphi(n) = (p-1)(q-1)$

$\varphi(n) = (523-1)(857-1) = 522 \,.\, 856 = 446832$

5. Akan ditentukan sembarang bilangan bulat *e* yang relatif prima terhadap $\varphi(n)$. Misalkan pada contoh ini dipilih $e = 719$, karena $gcd(446832, 719) = 1$. Hal ini dapat ditunjukkan sebagai berikut :

   $446832 = 621 \,.\, 719 + 333$

$719 = 2 . 333 + 53$

$333 = 6 . 53 + 15$

$53 = 3 . 15 + 8$

$15 = 1 . 8 + 7$

$8 = 1 . 7 + 1$

$7 = 7 . 1 + 0$

Karena sisa terakhi yang tidak 0 adalah 1, maka $gcd(446832, 719) = 1$

6. Setelah didapatkan nilai *e* akan ditentukan bilangan bulat *d* dengan rumus $d = \frac{1+k\varphi(n)}{e}$ dimana dalam contoh ini $k = 95$.

$$d = \frac{1 + k\varphi(n)}{e} = \frac{1 + 95 . 446832}{719} = 59039$$

Dengan demikian kunci umum adalah pasangan (719, 448211)

kunci pribadi adalah pasangan (59039, 448211)

7. Plain teks diubah ke dalam bentuk bilangan.

Berdasarkan kode ASCII plain teks diubah menjadi :

66101114115105971123210010532112101114989711697115971104432112

10111012110111497110103971103210010510911710897105321121171071

17108324853584848463283105971121079711032514848483211297115117

10797110321051101029711011610111410532100971103249484848321071

01110100971149797110321121011149711010346326697110116117971103

21171009711497329710797110329810111497110103107971163211211710

71171083249484648484432117115971049710797110329810111411697104

971103211597109112971053298971101161179711032117100971149732100971697110103

8. Plain teks m diubah menjadi blok-blok $m_1, m_2, m_3, \ldots$, sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang $[0, n-1]$

   Plain teks=

   66101 11411 51059 71123 21001 05321 12101 11498 97116 97115 97110 44321 12101 11012 11011 14971 10103 97110 32100 10510 91171 08971 05321 12117 10711 71083 24853 58484 84632 83105 97112 10797 11032 51484 84832 11297 11511 71079 71103 21051 10102 97110 11610 11141 05321 00971 10324 94848 48321 07101 11010 09711 49797 11032 11210 11149 71101 03463 26697 11011 61179 71103 21171 00971 14973 29710 79711 03298 10111 49711 01031 07971 16321 12117 10711 71083 24948 46484 84432 11711 59710 49710 79711 03298 10111 41169 71049 71103 21159 71091 12971 05329 89711 01161 17971 10321 17100 97114 97321 00971 16971 10103

9. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus

   $c_i = m_i^e \bmod n$

   $c_1 = 66101^{719} \bmod 448211 = 278684$

   $c_2 = 11411^{719} \bmod 448211 = 445233$

   $c_3 = 51059^{719} \bmod 448211 = 165190$

   $c_4 = 71123^{719} \bmod 448211 = 100318$

   $c_5 = 21001^{719} \bmod 448211 = 52057$

⋮

$c_{99} = 97321^{719} \ mod \ 448211 = 366234$

$c_{100} = 00971^{719} \ mod \ 448211 = 359830$

$c_{101} = 16971^{719} \ mod \ 448211 = 173489$

$c_{102} = 10103^{719} \ mod \ 448211 = 419361$

Chipper teks hasil enkripsi =

278684 445233 165190 100318 52057 99277 202907 76030 123315 289122 257663 189353 202907 182883 158240 104136 419361 257663 336375 125055 351663 251565 99277 139983 215220 56454 392506 376940 363268 97983 95271 350717 426878 340916 156921 51761 40892 319443 335053 70612 347691 257663 270919 20568 99277 359830 225945 256928 78280 304654 896 323115 182569 426878 329305 42801 411228 119856 353389 158240 437298 335053 397747 359830 11557 108073 102565 171952 312960 241138 278663 199125 340601 139983 215220 56454 408193 312647 418448 70958 372718 61381 102565 171952 312960 362897 275674 335053 122951 105714 349359 179386 132522 417128 144218 163557 440801 393281 366234 359830 173489 419361

10. Pada sisi penerima blok chipper teks c_i didekripsi kembali menjadi blok m_i dengan rumus

$m_i = {c_i}^d \ mod \ n$

$m_1 = 278684^{89039} \ mod \ 448211 = 66101$

$m_2 = 445233^{89039} \ mod \ 448211 = 11411$

$m_3 = 165190^{89039} \ mod \ 448211 = 51059$

$m_4 = 100318^{89039} \ mod \ 448211 = 71123$

$m_5 = 52057^{89039} \ mod \ 448211 = 21001$

$\vdots$

$m_{99} = 366234^{89039} \ mod \ 448211 = 97321$

$m_{100} = 359830^{89039} \ mod \ 448211 = 00971$

$m_{101} = 173489^{89039} \ mod \ 448211 = 16971$

$m_{102} = 419361^{89039} \ mod \ 448211 = 10103$

Plain teks hasil dekripsi =

66101 11411 51059 71123 21001 05321 12101 11498 97116 97115 97110 44321 12101 11012 11011 14971 10103 97110 32100 10510 91171 08971 05321 12117 10711 71083 24853 58484 84632 83105 97112 10797 11032 51484 84832 11297 11511 71079 71103 21051 10102 97110 11610 11141 05321 00971 10324 94848 48321 07101 11010 09711 49797 11032 11210 11149 71101 03463 26697 11011 61179 71103 21171 00971 14973 29710 79711 03298 10111 49711 01031 07971 16321 12117 10711 71083 24948 46484 84432 11711 59710 49710 79711 03298 10111 41169 71049 71103 21159 71091 12971 05329 89711 01161 17971 10321 17100 97114 97321 00971 16971 10103

11. Kemudian blok-blok $m_1, m_2, m_3, \ldots$ diubah kembali ke bentuk huruf dengan melihat kode ASCII hasil dekripsi.

Percobaan pemfaktoran nilai *n* dari contoh 3.3 dengan menggunakan matlab 6.5. Faktor 448211=523 dan 857, waktu proses=0,25 detik.

Contoh 3.4.

1. Dengan plain teks yang sama akan dicoba dilakukan proses penyandian menggunakan pasangan kunci yang berbeda
2. Dipilih $p = 2903$ dan $q = 5879$.
3. $n = pq = 2903 .5879 = 17066737$
4. $\varphi(n) = (p-1)(q-1)$

$\varphi(n) = (2903-1)(5879-1) = 2902.5878 = 17057956$

5. Kunci umum, $e = 4423$ dimana *e* relatif prima terhadap $\varphi(n)$.

   $gcd(17057956,4423)$:

12. Akan ditentukan sembarang bilangan bulat *e* yang relatif prima terhadap $\varphi(n)$. Misalkan pada contoh ini dipilih $e = 4423$, karena $gcd(17057956,4423) = 1$. Hal ini dapat ditunjukkan sebagai berikut :

    17057956=3856 . 4423+2868

    4423=1 . 2868+1555

    2868=1 . 1555+1313

    1555=1 . 1313+242

    1313=5 . 242+103

    242=2 . 103+36

103=2 . 36+31

36=1 . 31+5

31=6 . 5+1

5=5 . 1+0

Karena sisa terakhir yang tidak 0 adalah 1 maka $gcd(17057956, 4423) = 1$

6. Kunci pribadi $d = 30803051$

   d dihitung dengan rumus berikut dengan $k = 7987$

$$d = \frac{1 + k\varphi(n)}{e} = \frac{1 + 7987.17057956}{4423} = 30803051$$

Dengan demikian kunci umum adalah pasangan (4423, 17066737)

kunci pribadi adalah pasangan (30803051, 17066737)

7. Plain teks diubah ke dalam bentuk bilangan.

   Berdasarkan kode ASCII plain teks diubah menjadi :

   66101114115105971123210010532112101114989711697115971104432112

   10111012110111497110103971103210010510911710897105321121171071

   17108324853584848463283105971121079711032514848483211297115117

   10797110321051101029711011610111410532100971103249484848321071

   01110100971149797110321121011149711010346326697110116117971103

   21171009711497329710797110329810111497110103107971163211211710

   71171083249484648484432117115971049710797110329810111411697104

   97110321159710911297105329897110116117971103211710097114973210

   09711697110103

8. Plain teks m diubah menjadi blok-blok $m_1, m_2, m_3, \ldots,$ sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang $[0, n-1]$

   Plain teks=

   6610111 4115105 9711232 1001053 2112101 1149897 1169711 5971104 4321121 0111012 1101114 9711010 3971103 2100105 1091171 0897105 3211211 7107117 1083248 5358484 8463283 1059711 2107971 1032514 8484832 1129711 5117107 9711032 1051101 0297110 1161011 1410532 1009711 0324948 4848321 0710111 0100971 1497971 1032112 1011149 7110103 4632669 7110116 1179711 0321171 0097114 9732971 0797110 3298101 1149711 0103107 9711632 1121171 0711710 8324948 4648484 4321171 1597104 9710797 1103298 1011141 1697104 9711032 1159710 9112971 0532989 7110116 1179711 0321171 0097114 9732100 9711697 110103

9. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus

$c_i = m_i^e \bmod n$

$c_1 = 6610111^{4423} \bmod 17066737 = 14808817$

$c_2 = 4115105^{4423} \bmod 17066737 = 5606162$

$c_3 = 9711232^{4423} \bmod 17066737 = 1142976$

$c_4 = 1001053^{4423} \bmod 17066737 = 11165065$

$c_5 = 2112101^{4423} \bmod 17066737 = 16815130$

$\vdots$

$c_{70} = 0097114^{4423} \bmod 17066737 = 12830877$

$c_{71} = 9732100^{4423} \ mod \ 17066737 = 16915487$

$c_{72} = 9711697^{4423} \ mod \ 17066737 = 15342794$

$c_{73} = 110103^{4423} \ mod \ 17066737 = 1349445$

Chipper teks hasil enkripsi =

14808817 5606162 1142976 11165065 16815130 8138417 14407714 15384018 8663608 9733628 10161263 5289413 4058311 14869325 5511654 6157115 10061060 9061592 1205219 5078190 5296924 9177581 7496605 6841632 7695725 4742165 11385053 14478472 8176181 1412251 2164670 4322368 3341477 1730825 11024490 3334445 12261815 10589812 12217855 14035672 15605431 6847699 5243023 7461320 3006522 12830877 221357 13238256 15470132 16888658 13975649 9373187 2222450 3684054 4969681 1812580 178125 283651 16153315 11169479 5865061 10462148 14478472 6249089 3044404 6735632 5243023 7461320 3006522 12830877 16915487 15342794 1349445

10. Pada sisi penerima blok chipper teks c_i didekripsi kembali menjadi blok m_i dengan rumus

$m_i = c_i^{\ d} \ mod \ n$

$m_1 = 14808817^{30803051} \ mod \ 17066737 = 6610111$

$m_2 = 5606162^{30803051} \ mod \ 17066737 = 4115105$

$m_3 = 1142976^{30803051} \ mod \ 17066737 = 9711232$

$m_4 = 11165065^{30803051} \ mod \ 17066737 = 1001053$

$m_5 = 16815130^{30803051} \ mod \ 17066737 = 2112101$

⋮

$m_{70} = 12830877^{30803051} \ mod \ 17066737 = 0097114$

$m_{71} = 16915487^{30803051} \ mod \ 17066737 = 9732100$

$m_{72} = 15342794^{30803051} \ mod \ 17066737 = 9711697$

$m_{73} = 1349445^{30803051} \ mod \ 17066737 = 110103$

Plain teks hasil dekripsi =

6610111 4115105 9711232 1001053 2112101 1149897 1169711 5971104 4321121 0111012 1101114 9711010 3971103 2100105 1091171 0897105 3211211 7107117 1083248 5358484 8463283 1059711 2107971 1032514 8484832 1129711 5117107 9711032 1051101 0297110 1161011 1410532 1009711 0324948 4848321 0710111 0100971 1497971 1032112 1011149 7110103 4632669 7110116 1179711 0321171 0097114 9732971 0797110 3298101 1149711 0103107 9711632 1121171 0711710 8324948 4648484 4321171 1597104 9710797 1103298 1011141 1697104 9711032 1159710 9112971 0532989 7110116 1179711 0321171 0097114 9732100 9711697 110103

11. Kemudian blok-blok $m_1, m_2, m_3, \ldots$ diubah kembali ke bentuk huruf dengan melihat kode ASCII hasil dekripsi.

Percobaan pemfaktoran nilai n dari contoh 3.4 dengan menggunakan matlab 6.5.

Faktor 17066737=2903 dan 5879, waktu proses=0,711 detik

3.8. Perbandingan RSA Coding Dengan Algoritma Kriptografi Yang Lain

Pada dasarnya kriptografi kunci simetris dan kunci assimetris memiliki kelebihan dan kekurangan masing-masing, yakni :

Kriptografi kunci simetris :

Kelebihan :

1. Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetrik.
2. Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time*

Kekurangan :

1. Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
2. Jika jumlah pengguna banyak, maka jumlah kunci yang harus yang diorganisir akan sangat banyak, yaitu sebanyak $\frac{n(n-1)}{2}$.
3. Tidak memenuhi aspek autentifikasi, sehingga dapat terjadi penyangkalan pesan dari pihak pengirim.

Kriptografi kunci assimetris :

Kelebihan :

1. Jumlah kunci lebih sedikit dari kriptografi kunci simetris.
2. Memenuhi aspek autentifikasi, karena chipper teks hanya dapat didekripsi dengan kunci pribadi pengirim. Jadi pengirim tidak dapat menyangkal bahwa pesan tersebut dikirim oleh dirinya.

Kekurangan :

1. Membutuhkan waktu proses yang lebih lama.

Selanjutnya akan dipaparkan beberapa algoritma kriptografi selain RSA Coding sebagai bahan perbandingan :

1. Data Encryption Standard (DES)

Algoritma DES dikembangkan di IBM di bawah kepemimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma LUCIFER yang dibuat oleh Horst Feistel.

DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis cipher blok artinya DES bekerja dalam plaintext dengan ukuran yang telah diberikan dan mengembalikan ciphertext dengan ukuran yang sama pula. DES beroperasi pada ukuran blok 64 bit (angka biner 0 dan 1). DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks. Untuk melakukan enkripsi, DES membutuhkan kunci yang juga mempunyai ukuran 64 bit, namun dalam prakteknya bit ke 8 dari setiap kelompok 8 bit diabaikan, sehingga ukuran kunci menjadi 56 bit. Skema global dari algoritma DES adalah sebagai berikut :

1. Blok plainteks dipermutasi (pertukaran) dengan tabel permutasi awal (*initial permutation* atau IP).

   Proses permutasi awal mengacu pada tabel IP sebagai berikut :

   Tabel 3.1. Tabel Permutasi Awal (IP)

Input Bit	1	2	3	4	5	6	7	8
Output Bit	58	50	42	34	26	18	10	2
Input Bit	9	10	11	12	13	14	15	16
Output Bit	60	52	44	36	28	20	12	4
Input Bit	17	18	19	20	21	22	23	24

Output Bit	62	54	46	38	30	22	14	6

Input Bit	25	26	27	28	29	30	31	32
Output Bit	64	56	48	40	32	24	16	8
Input Bit	33	34	35	36	37	38	39	40
Output Bit	57	49	41	33	25	17	9	1
Input Bit	41	42	43	44	45	46	47	48
Output Bit	59	51	43	35	27	19	11	3
Input Bit	49	50	51	52	53	54	55	56
Output Bit	61	53	45	37	29	21	13	5
Input Bit	57	58	59	60	61	62	63	64
Output Bit	63	55	47	39	31	23	15	7

Tabel tersebut dibaca bit 1 menjadi bit 40, bit 58 menjadi bit 1 dan seterusnya.

Contoh : Misalkan plain teks SEMARANG

Plain teks	Decimal	Hexa Decimal	Biner
S	83	53	01010011
E	69	45	01000101
M	77	4D	01001101
A	65	41	01000001
R	82	52	01010010
A	65	41	01000001
N	78	4E	01001110
G	71	47	01000111

Maka plain teks biner, X = 01010011 01000101 01001101 01000001 01010010 01000001 01001110 01000111

Kemudian plain teks dipermutasi awal menjadi X_0, dengan

$X_0 = IP(X) = L_0R_0$

X_0 = 11111111 00010001 11000110 10101111 00000000 00000000 01000100 11010001

Maka, L_0 = 11111111 00010001 11000110 10101111

R_0 = 00000000 00000000 01000100 11010001

2. Hasil permutasi awal kemudian di*enciphering* (proses penyandian) sebanyak 16 kali/putaran. Setiap putaran menggunakan kunci internal yang berbeda. Pada proses putaran ke-16 terjadi pertukaran pada sisi kiri (L) dan sisi kanan (R). Proses ini menghasilkan pre-output.

Pencarian Kunci 16 putaran

Kunci internal didapat dari kunci awal (64 bit) yang dipermutasi dengan PC-1 (permutasi choice one) berdasarkan tabel PC-1 menghasilkan Kunci Internal 56 bit.

Tabel 3.2. Tabel Permutasi Choice One (PC-1)

Input Bit	1	2	3	4	5	6	7	8
Output Bit	57	49	41	33	25	17	9	1
Input Bit	9	10	11	12	13	14	15	16
Output Bit	58	50	42	34	26	18	10	2
Input Bit	17	18	19	20	21	22	23	24
Output Bit	59	51	43	35	27	19	11	3
Input Bit	25	26	27	28	29	30	31	32
Output Bit	60	52	44	36	63	55	47	39
Input Bit	33	34	35	36	37	38	39	40
Output Bit	31	23	15	7	62	54	46	38
Input Bit	41	42	43	44	45	46	47	48
Output Bit	30	22	14	6	61	53	45	37
Input Bit	49	50	51	52	53	54	55	56
Output Bit	29	21	13	5	28	20	12	4

Misalkan Kunci Awal Hexa=14 6D 25 C5 9B AD 37 1E

K biner = 00010100 01101101 00100101 11000101 10011011 10101101 00110111 00011110

Selanjutnya *K* dipermutasi dengan PC-1 menghasilkan K_0

K_0 = 00111000 00001010 01100110 11011101 00001110 11111011 00100001

Selanjutnya K_0 dibagi menjadi 2 bagian C_0 dan D_0 dimana masing-masing terdiri dari 28 bit.

C_0 = 0011100000001010011001101101

D_0 = 1101000011101111101100100001

Setelah C_0 dan D_0 didefinisikan, selanjutnya dibuat 16 blok C_n dan D_n, $1 \leq n \leq 16$. Setiap pasang dari blok C_n dan D_n dibentuk dari pasangan C_{n-1} dan D_{n-1} secara berulang untuk $n = 1, 2, \ldots, 16$ menggunakan aturan "left shift" dari blok sebelumnya. Aturan "left shift" adalah memindahkan setiap bit ke kiri kecuali beberapa bit pertama yang akan berpindah ke bit terakhir.

Banyaknya pergeseran untuk masing-masing *n* adalah :

Tabel 3.3. Tabel Pergeseran Bit (Left Shift)

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

Kunci 16 putaran didapat dari pasangan C_nD_n yang dipermutasi dengan PC-2 (permutasi choice two). Tabel PC-2 adalah sebagai berikut :

Tabel 3.4. Tabel Permutasi Choice Two (PC–2)

Input Bit	1	2	3	4	5	6	7	8
Output Bit	14	17	11	24	1	5	3	28

Input Bit	9	10	11	12	13	14	15	16
Output Bit	15	6	21	10	23	19	12	4
Input Bit	17	18	19	20	21	22	23	24
Output Bit	26	8	16	7	27	20	13	2
Input Bit	25	26	27	28	29	30	31	32
Output Bit	41	52	31	37	47	55	30	40
Input Bit	33	34	35	36	37	38	39	40
Output Bit	51	45	33	48	44	49	39	56
Input Bit	41	42	43	44	45	46	47	48
Output Bit	34	53	46	42	50	36	29	32

C_0 = 00111000000010100110011011 01

D_0 = 11010000111011111011001000 01

C_1 = 01110000000101001100110110 10

D_1 = 10100001110111110110010000 11

C_1D_1 = 01110000 00010100 11001101 10101010 00011101 11110110 01000011

$K_1 = PC - 2(C_1D_1)$

K_1 = 110100100010001100001001101111010000100100111110

Proses ini dilakukan sampai dihasilkan K_{16}.

Setelah semua kunci internal untuk 16 putaran, selanjutnya dilakukan enchipering plain teks 16 putaran. Dengan menggunakan rumus :

$L_n = R_{n-1}$

$R_n = L_{n-1} \oplus f(R_{n-1}, K_n),$

Dimana fungsi *f* terdiri dari tiga tahapan.

Pertama, proses ekspansi nilai R_{n-1} dari 32 bit menjadi 48 bit dengan tabel ekspansi.

Tabel 3.5. Tabel Fungsi Ekspansi

Input Bit	1	2	3	4	5	6	7	8
Output Bit	32	1	2	3	4	5	4	5
Input Bit	9	10	11	12	13	14	15	16
Output Bit	6	7	8	9	8	9	10	11
Input Bit	17	18	19	20	21	22	23	24
Output Bit	12	13	12	13	14	15	16	17
Input Bit	25	26	27	28	29	30	31	32
Output Bit	16	17	18	19	20	21	20	21
Input Bit	33	34	35	36	37	38	39	40
Output Bit	22	23	24	25	24	25	26	27
Input Bit	41	42	43	44	45	46	47	48
Output Bit	28	29	28	29	30	31	32	1

L_0 = 11111111 00010001 11000110 10101111

R_0 = 00000000 00000000 01000100 11010001

K_1 = 110100100010001100001001101111010000100100111110

$L_1 = R_0$ = 00000000 00000000 01000100 11010001

$R_1 = L_0 \oplus f(R_1, K_0)$

$f(R_0, K_1) = E(R_0) \oplus K_1$

$E(R_0)$ = 10000000 00000000 00000000 00100000 10010110 10100010

Kemudian nilai $E(R_{n-1})$ dikenakan operasi XOR dengan K_n.

$f(R_0, K_1) =$

10000000 00000000 00000000 00100000 10010110 10100010

11010010 00100011 00001001 10111101 00001001 00111110 ⊕

01010010 00100011 00001001 10010101 10011111 10011100

$f(R_0, K_1)$ = 010100100010001100001001100111011001111110011100

Hasil proses ini akan dikenai sebuah operasi pada setiap grup yang terdiri dari 6 bit. Setiap grup tersebut akan digunakan sebagai alamat pada tabel "S-boxes".

Tabel 3.6. Tabel S-Box

		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S_1	**0**	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	**0**	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	**0**	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	**0**	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	**0**	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	**0**	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	**0**	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2

	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	**0**	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-Box memiliki baris yang diberi nomor 0 sampai 3, dan kolom yang diberi nomor 0 sampai 15. Masukan untuk proses subtitusi adalah 6 bit, $b_1b_2b_3b_4b_5b_6$. Nomor baris dari tabel ditunjukkan oleh bit b_1b_6 (menyatakan 0 sampai 3 desimal). Nomor kolom ditunjukkan oleh bit $b_2b_3b_4b_5$ (menyatakan 0 sampai 15 desimal) .

48 bit $f(R_{n-1}, K_1)$ dibagi menjadi blok-blok 6 bit, sehingga menjadi

$S_1(b_1b_2b_3b_4b_5b_6)$ $S_2(b_1b_2b_3b_4b_5b_6)$ $S_3(b_1b_2b_3b_4b_5b_6)$ $S_4(b_1b_2b_3b_4b_5b_6)$

$S_5(b_1b_2b_3b_4b_5b_6)$ $S_6(b_1b_2b_3b_4b_5b_6)$ $S_7(b_1b_2b_3b_4b_5b_6)$ $S_8(b_1b_2b_3b_4b_5b_6)$

010100 100010 001100 001001 100101 011001 111110 011100

S_1(010100) = baris 00 dan kolom 1010

= baris 0 dan kolom 10 = 6 = 0110

S_2(100010) = baris 10 dan kolom 0001

= baris 2 dan kolom 1 = 14 = 1110

S_3(001100) = baris 00 dan kolom 0110

= baris 0 dan kolom 6 = 15 = 1111

S_4(001001) = baris 01 dan kolom 0100

= baris 1 dan kolom 4 = 6 = 0110

S_5(100101) = baris 11 dan kolom 0010

= baris 3 dan kolom 2 = 12 = 1100

S_6(011001) = baris 01 dan kolom 1100

= baris 1 dan kolom 12 = 0 = 0000

S_7(111110) = baris 10 dan kolom 1111

= baris 2 dan kolom 15 = 2 = 0010

S_8(011100) = baris 00 dan kolom 1110

= baris 0 dan kolom 14 = 12 = 1100

Jadi, $f(R_0, K_1) =$ 0110 1110 1111 0110 1100 0000 0010 1100

Selanjutnya proses terakhir dari f adalah hasil output dari S-BOX dikenai permutasi dengan tabel berikut :

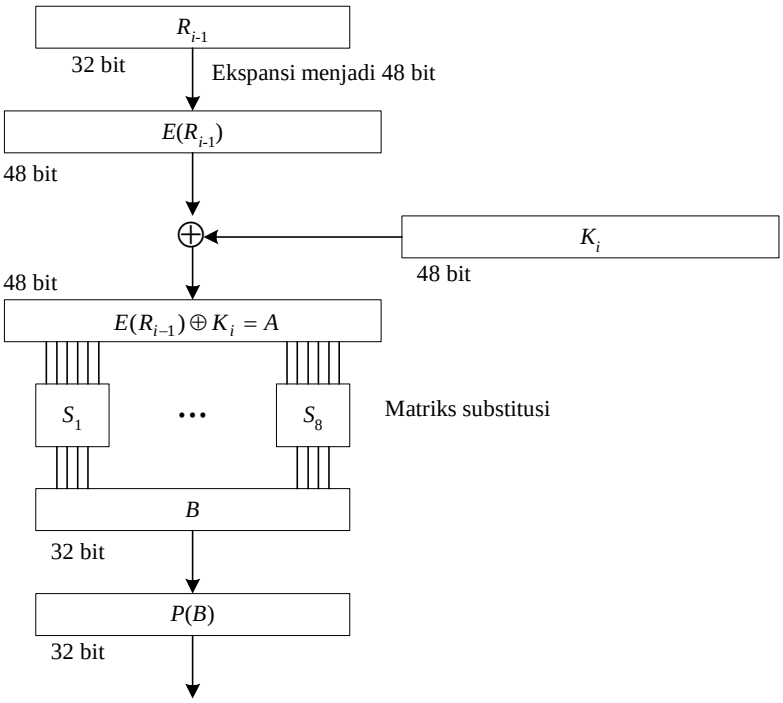
Tabel 3.7. Tabel Fungsi Permutasi

Input Bit	1	2	3	4	5	6	7	8
Output Bit	16	7	20	21	29	12	28	17
Input Bit	9	10	11	12	13	14	15	16
Output Bit	1	15	23	26	5	18	31	10
Input Bit	17	18	1`9	20	21	22	23	24
Output Bit	2	8	24	14	32	27	3	9
Input Bit	25	26	27	28	29	30	31	32
Output Bit	19	13	30	6	22	11	4	25

$f(R_0, K_1) = P$(01101110111101101100000000101100)

$f(R_0, K_1) =$ 01101100010011011001011110110100

Diagram komputasi fungsi f diperlihatkan dengan gambar berikut :



Gambar 3.1. Diagram komputasi fungsi f

Setelah melalui fungsi *f*, nilai R_n dapat dicari,

$R_1 = L_0 \oplus f(R_0, K_1)$

11111111000100011100011010101111

01101100010011011001011110110100 ⊕

10010011010111000101000100011011

$R_1 = 10010011010111000101000100011011$

Proses tersebut dilakukan sampai 16 putaran, dimana pada putaran 16 nilai L_{16} dan R_{16} saling ditukarkan yang kemudian disebut pre-output. Sehingga nilai masukkan invers permutasi adalah $R_{16}L_{16}$.

3. Hasil pre-output kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^{-1}) menjadi blok cipherteks. Tabel IP^{-1} sebagai berikut :

Tabel 3.8. Tabel Inverse Permutasi (IP^{-1})

Input Bit	1	2	3	4	5	6	7	8
Output Bit	40	8	48	16	56	24	64	32
Input Bit	9	10	11	12	13	14	15	16
Output Bit	39	7	47	15	55	23	63	31
Input Bit	17	18	19	20	21	22	23	24
Output Bit	38	6	46	14	54	22	62	30
Input Bit	25	26	27	28	29	30	31	32
Output Bit	37	5	45	13	53	21	61	29
Input Bit	33	34	35	36	37	38	39	40
Output Bit	36	4	44	12	52	20	60	28
Input Bit	41	42	43	44	45	46	47	48
Output Bit	35	3	43	11	51	19	59	27
Input Bit	49	50	51	52	53	54	55	56
Output Bit	34	2	42	10	50	18	58	26
Input Bit	57	58	59	60	61	62	63	64
Output Bit	33	1	41	9	49	17	57	25

Invers permutasi disusun sedemikian sehingga sesuai dengan initial permutasi.

Misalnya M adalah bilangan biner, kemudian dilakukan permutasi $X = IP(M)$, selanjutnya jika diambil inverse dari permutasi $Y = IP^{-1}(X) = IP^{-1}(IP(M))$ maka $Y = M$.

Contoh :

Plain teks = SEMARANG

X = 01010011 01000101 01001101 01000001 01010010 01000001 01001110 01000111

$X_0 = IP(X)$

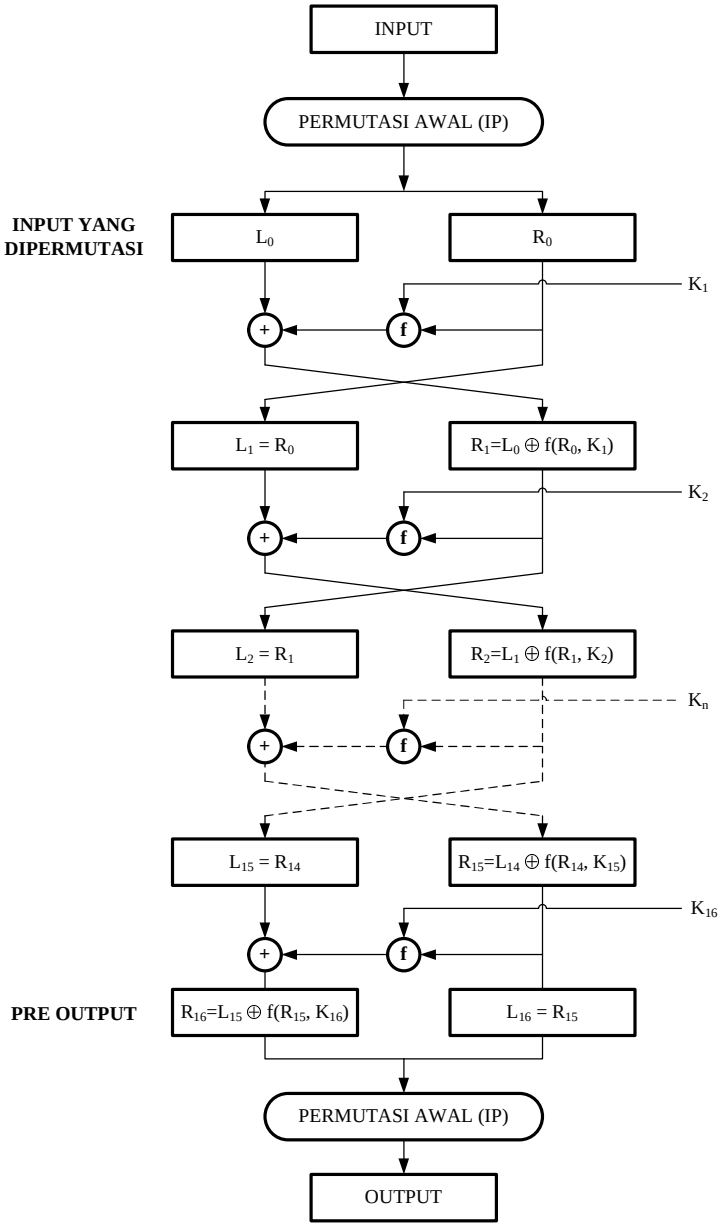
X_0 =11111111 00010001 11000110 10101111 00000000 00000000 01000100 11010001

$Y = IP^{-1}(X_0)$

Y = 01010011 01000101 01001101 01000001 01010010 01000001 01001110 01000111

Maka $Y = X$.

4. Skema global dari algoritma DES dapat digambarkan sebagai berikut :



Gambar 3.2. Skema Global Algoritma DES

Implementasi DES

Plain teks = SEMARANG

$X =$ 01010011 01000101 01001101 01000001 01010010 01000001 01001110 01000111

$X_0 = IP(X) = L_0R_0$

X_0 =11111111 00010001 11000110 10101111 00000000 00000000 01000100 11010001

Maka, L_0= 11111111 00010001 11000110 10101111

R_0 = 00000000 00000000 01000100 11010001

Kunci Awal Hexa = 14 6D 25 C5 9B AD 37 1E

$K =$ 00010100 01101101 00100101 11000101 10011011 10101101 00110111 00011110

$K_0 = PC^{-1}(K)$

$K_0 =$ 00111000 00001010 01100110 11011101 00001110 11111011 00100001

$C_0 =$ 0011100000001010011001101101

$D_0 =$ 1101000011101111101100100001

Putaran 1	**Bilangan Biner**
C_1	0111000000010100110011011010
D_1	1010000111011111011001000011

C_1D_1	011100000001010011001101101010100001110111111011001000011
K_1	110100100010001100001001101111010000100100111110
$L_1 = R_0$	00000000000000000100010011010001
$E(R_0)$	10000000000000000000000001000001001011010100010
$\mathrm{E(R_0)} \oplus \mathrm{K_1}$	010100100010001100001001100111011001111110011100
$SBOX(E(R_0) \oplus K_1)$	011011101111011001110000000101100
$P(SBOX(E(R_0) \oplus K_1))$	011011000100110110010111110110100
$R_1 = L_0 \oplus f(R_0, K_1)$	100100110101110001010001000110 11
Putaran 2	**Bilangan Biner**
C_2	111000000010100110011011010 0
D_2	010000111011111011001000011 1
C_2D_2	111000000010100110011011010001000011101111101100100001111
K_2	011110100010100010100111100101110100011100110100
$L_2 = R_1$	100100110101110001010001000110 11
$E(R_1)$	110010100110101011111000001010100010100011110111
$E(R_1) \oplus K_2$	101100000100001001011111101111010110111111000011
$SBOX(E(R_1) \oplus K_2)$	001010000011100111010100110011 11
$P(SBOX(E(R_1) \oplus K_2))$	101011010001111000001010011011 01
$R_2 = L_1 \oplus f(R_1, K_2)$	101011010001111001001110101111 00
Putaran 3	**Bilangan Biner**
C_3	100000001010011001101101001 1
D_3	000011101111101100100001110 1
C_3D_3	100000001010011001101101001100001110111111011001000011101
K_3	101110011010010000001000110110010010101111000000
$L_3 = R_2$	101011010001111001001110101111 00
$E(R_2)$	010101011010100011111100001001011101010111111001
$E(R_2) \oplus K_3$	111011000000110011110100111111001111111000111001

$SBOX(E(R_2) \oplus K_3)$	00001111111100110011010100000011
$P(SBOX(E(R_2) \oplus K_3))$	11100100010010110110100110011100
$R_3 = L_2 \oplus f(R_2, K_3)$	01110111000101110011100010000111
Putaran 4	**Bilangan Biner**
C_4	00000010100110011011010011110
D_4	00111011111011001000011101 00
C_4D_4	0000001010011001101101001110001110111110110010000111010 0
K_4	01000000000001101011111011110000111000100001 1101
$L_4 = R_3$	01110111000101110011100010000111
$E(R_3)$	10111010111010001010111010011111000101000000 1110
$E(R_3) \oplus K_4$	11111010111011100001000001101111111011000010011
$SBOX(E(R_3) \oplus K_4)$	00000001010100011001110101010101
$P(SBOX(E(R_3) \oplus K_4))$	10110111000100010110100000101000
$R_4 = L_3 \oplus f(R_3, K_4)$	00011010000011110010011010010100
Putaran 5	**Bilangan Biner**
C_5	00001010011001101101001110 00 0
D_5	11101111101100100001110100 00
C_5D_5	0000101001100110110100111000111011111011001000011101000 0
K_5	11110100100110000001010001110011001101101000 1110
$L_5 = R_4$	00011010000011110010011010010100
$E(R_4)$	00001111010000000101111010010000110101001010100 0
$E(R_4) \oplus K_5$	11111011110110000100101011100011111000100010 0110
$SBOX(E(R_4) \oplus K_5)$	00001110000101100110011011110001
$P(SBOX(E(R_4) \oplus K_5))$	01000110011111000001110010011001
$R_5 = L_4 \oplus f(R_4, K_5)$	00110001011010110010010000011110
Putaran 6	**Bilangan Biner**
C_6	00101001100110110100111 00000

D_6	10111110110010000111010000011
C_6D_6	00101001100110110100111000001011111011001000011101000011
K_6	00000110101010100110001010111100001100011010101 1
$L_6 = R_5$	00110001011010110010010000011110
$E(R_5)$	00011010001010110101011010010000100000011111100
$E(R_5) \oplus K_6$	00011100100000010011010000101100101100010101011 1
$SBOX(E(R_5) \oplus K_6)$	01000110100100110111110010111011
$P(SBOX(E(R_5) \oplus K_6))$	11111110010001101000110110011001
$R_6 = L_5 \oplus f(R_5, K_6)$	11100100010010011010101100001101
Putaran 7	**Bilangan Biner**
C_7	10100110011011010011100000000
D_7	11111011001000011101000011 10
C_7D_7	10100110011011010011100000001111101100100001110100001110
K_7	10101010011101000011011000100110011111010011001 11
$L_7 = R_6$	11100100010010011010101100001101
$E(R_6)$	11110000100000100101001111010101011010000101101 1
$E(R_6) \oplus K_7$	01011010111101100110010111110011000100100011110 0
$SBOX(E(R_6) \oplus K_7)$	11000010110000000000101111110101
$P(SBOX(E(R_6) \oplus K_7))$	01010010101100011010110100100001
$R_7 = L_6 \oplus f(R_6, K_7)$	01100011110110101000100100111111
Putaran 8	**Bilangan Biner**
C_8	10011001101101001110000000 10
D_8	11101100100001110100001 11011
C_8D_8	10011001101101001110000000101110110010000111010000111011
K_8	11101100000001110100100001110110101010011111001 0
$L_8 = R_7$	01100011110110101000100100111111
$E(R_7)$	10110000011111101111010101000101001010011111111 0

$E(R_7) \oplus K_8$	010111000111100110111101001100111000000000001100
$SBOX(E(R_7) \oplus K_8)$	10110111100100101011000101001011
$P(SBOX(E(R_7) \oplus K_8))$	01101101110100100110101110010010
$R_8 = L_7 \oplus f(R_7, K_8)$	10001001100110111100000010011111
Putaran 9	**Bilangan Biner**
C_9	00110011011010011100000000101
D_9	11011001000011101000011101111
C_9D_9	001100110110100111000000010111011001000011101000011101111
K_9	011000110001000111110010110001101110000100011111
$L_9 = R_8$	10001001100110111100000010011111
$E(R_8)$	110001010011110011110111111000000001010011111111
$E(R_8) \oplus K_9$	101001100010110100000101001001101111010111100000
$SBOX(E(R_8) \oplus K_9)$	01001110001010110100101011000111
$P(SBOX(E(R_8) \oplus K_9))$	11010000011111101000100001110101
$R_9 = L_8 \oplus f(R_8, K_9)$	10110011101001000000000101001010
Putaran 10	**Bilangan Biner**
C_{10}	11001101101001110000000010100
D_{10}	01100100001110100000111011111
$C_{10}D_{10}$	11001101101001110000000010100011001000011101000001110111111
K_{10}	10111100110000001110000111100111000101111100100 0
$L_{10} = R_9$	10110011101001000000000101001010
$E(R_9)$	01011010011111010000100000000000001010100101010 1
$E(R_9) \oplus K_{10}$	11100110101111011110100111100111001111011001110 1
$SBOX(E(R_9) \oplus K_{10})$	10101111001110101010111010001001
$P(SBOX(E(R_9) \oplus K_{10}))$	01011101111010000100101011011101
$R_{10} = L_9 \oplus f(R_9, K_{10})$	11010100011100111000101001000010
Putaran 11	**Bilangan Biner**

C_{11}	00110110100111000000001010011
D_{11}	10010000111010000111011111101
$C_{11}D_{11}$	0011011010011100000001010011100100001110100001110111110 1
K_{11}	10010011010000110001101011011000100100110110101 1
$L_{11} = R_{10}$	11010100011100111000101001000010
$E(R_{10})$	01101010100000111010011111000101010000100000010 1
$E(R_{10}) \oplus K_{11}$	11111001110000001011110100011101110100010110111 0
$SBOX(E(R_{10}) \oplus K_{11})$	00000101000000101100001110110010
$P(SBOX(E(R_{10}) \oplus K_{11}))$	00000011011001100110010000010001
$R_{11} = L_{10} \oplus f(R_{10}, K_{11})$	10110000110000100110010101011011
Putaran 12	**Bilangan Biner**
C_{12}	11011010011100000001010011 00
D_{12}	01000011101000011101111101 10
$C_{12}D_{12}$	1101101001110000000101001100010000111010000111011111011 0
K_{12}	00101100000100111001010101010110110111100010110 0
$L_{12} = R_{11}$	10110000110000100110010101011011
$E(R_{11})$	11011010000101100000010000110000101010101111011 1
$E(R_{11}) \oplus K_{12}$	11110110000001011001000101100110011101001101101 1
$SBOX(E(R_{11}) \oplus K_{12})$	01100000011101000011110000111110
$P(SBOX(E(R_{11}) \oplus K_{12}))$	00111110000000111001011010101100
$R_{12} = L_{11} \oplus f(R_{11}, K_{12})$	11101010011100000001110011101110
Putaran 13	**Bilangan Biner**
C_{13}	01101001110000000101001100 11
D_{13}	00001110100001110111111011 001
$C_{13}D_{13}$	0110100111000000010100110011000011101000011101111101100 1
K_{13}	00010111000110000100110101011000001111011111100 0
$L_{13} = R_{12}$	11101010011100000001110011101110

$E(R_{12})$	011101010100001110100000000011111001011101011101
$E(R_{12}) \oplus K_{13}$	011000100101101111101101010101111010101010100101
$SBOX(E(R_{12}) \oplus K_{13})$	010110100111110111111010011110
$P(SBOX(E(R_{12}) \oplus K_{13}))$	111111110000111110110100111011110
$R_{13} = L_{12} \oplus f(R_{12}, K_{13})$	010011111100110111010001101101 01
Putaran 14	**Bilangan Biner**
C_{14}	101001110000000101001100 1101
D_{14}	001110100001110111110110 0100
$C_{14}D_{14}$	101001110000000101001100 1101001110100001110111110110 0100
K_{14}	000010110110000011110000101010011111100000111001
$L_{14} = R_{13}$	010011111100110111010001101101 01
$E(R_{13})$	101001011111111001011011111010100011110110101010
$\mathrm{E}(\mathrm{R}_{13}) \oplus \mathrm{K}_{14}$	101011101001111010101011010000111100010110010011
$SBOX(E(R_{13}) \oplus K_{14})$	100100111010000110001011011 0101
$P(SBOX(E(R_{13}) \oplus K_{14}))$	110100111011000001101101001 00110
$R_{14} = L_{13} \oplus f(R_{13}, K_{14})$	001110011100000001110001110 01000
Putaran 15	**Bilangan Biner**
C_{15}	100111000000010100110011 0110
D_{15}	111010000111011111011001 0000
$C_{15}D_{15}$	100111000000010100110011 0110111010000111011111011001 0000
K_{15}	100111000100110110101100011000110111111000110010
$L_{15} = R_{14}$	001110011100000001110001110 01000
$E(R_{14})$	000111110011111000000000001110100011111001010000
$E(R_{14}) \oplus K_{15}$	100000110111001110101100010110010100000001100010
$SBOX(E(R_{14}) \oplus K_{15})$	010011000101011111110011 11011
$P(SBOX(E(R_{14}) \oplus K_{15}))$	101011110111111110111000 10010001
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$	111000001011001001101001 00100100

Putaran 16	**Bilangan Biner**
C_{16}	00111000000010100110011011 01
D_{16}	11010000111011111011001000 01
$C_{16}D_{16}$	00111000000010100110011011011010000111011111011001000 01
K_{16}	00000111100011011000001010011010110110011000 10011
$L_{16} = R_{15}$	11100000101100100110100100100100
$E(R_{15})$	01110000000101011010010000110101001010010000 1001
$E(R_{15}) \oplus K_{16}$	01110111100110000010011010101111111100100001 1010
$SBOX(E(R_{15}) \oplus K_{16})$	00110000110100001110110111110000
$P(SBOX(E(R_0) \oplus K_1))$	00010111000101010010011110001011
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$	00101110110101010101011001000011

Plain teks = SEMARANG

L_{16} = 11100000101100100110100100100100

R_{16} = 00101110110101010101011001000011

Maka pre outputnya adalah

$R_{16}L_{16}$ = 00101110 11010101 01010110 01000011 11100000 10110010 01101001 00100100

Kemudian pre-output diinvers permutasi menjadi chipper teks.

Chipper teks = $IP^{-1}(R_{16}L_{16})$

Biner = 00011001 01100101 01010110 01001000 00110100 11101010 10011101 10110000

Heksa = 19 65 56 48 34 EA 9D B0

Proses Dekripsi DES

Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah $K_1, K_2, K_3, \ldots, K_{16}$, maka pada proses dekripsi urutan kunci yang digunakan adalah $K_{16}, K_{15}, \ldots, K_1$.

Untuk tiap putaran 16, 15, …, 1, keluaran pada setiap putaran deciphering adalah

$$L_n = R_{n-1}$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n),$$

Dalam hal ini, (R_{16}, L_{16}) adalah blok masukan awal untuk deciphering. Blok (R_{16}, L_{16}) diperoleh dengan mempermutasikan cipherteks dengan matriks permutasi IP^{-1}. Pre-output dari deciphering adalah (L_0, R_0). Selanjutnya dengan mempermutasikan L_0, R_0 dengan permutasi awal IP akan didapatkan kembali blok plainteks semula.

Proses Pembangkitan Kunci Dekripsi

Tinjau kembali proses pembangkitan kunci internal pada proses enkripsi. Selama deciphering, K_{16} dihasilkan dari (C_{16}, D_{16}) dengan permutasi PC-2. Tentu saja (C_{16}, D_{16}) tidak dapat diperoleh langsung pada permulaan

deciphering. Tetapi karena (C_{16}, D_{16})=(C_0, D_0), maka K_{16} dapat dihasilkan dari (C_0, D_0) tanpa perlu lagi melakukan pergeseran bit.

Selanjutnya, K_{15} dihasilkan dari (C_{15}, D_{15}) yang mana (C_{15}, D_{15}) diperoleh dengan menggeser C_{16} (yang sama dengan C_0) dan D_{16} (yang sama dengan D_0) satu bit ke kanan. Sisanya, K_{14} sampai K_1 dihasilkan dari (C_{14}, D_{14}) sampai (C_1, D_1). Catatlah bahwa (C_{i-1}, D_{i-1}) diperoleh dengan menggeser C_i dan D_i dengan cara yang sama seperti pada Tabel 1, tetapi pergeseran kiri (left shift) diganti menjadi pergeseran kanan (right shift).

Keamanan DES

Hal-hal yang disorot menyangkut keamanan DES adalah :

1. Panjang kunci

   Panjang kunci eksternal DES hanya 64 bit atau 8 karakter, itupun yang dipakai hanya 56 bit. Hal tersebut dianggap kunci DES terlalu pendek.

   Namun, dengan panjang kunci 56 bit akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci. Pada awal pembuatannya diasumsikan serangan *exhaustive key search* dengan menggunakan prosesor paralel akan memerlukan waktu 1142 tahun untuk menemukan kunci yang benar.

   Namun pada Tahun 1998, *Electronic Frontier Foundation* (EFE) merancang dan membuat perangkat keras khusus untuk menemukan kunci DES secara *exhaustive search key* dengan biaya $250.000 dan diharapkan dapat menemukan kunci selama 5 hari. Dan pada Tahun

1999, kombinasi perangkat keras EFE dengan kolaborasi internet yang melibatkan lebih dari 100.000 komputer dapat menemukan kunci DES kurang dari 1 hari.

2. Jumlah putaran

   Pada proses DES membutuhkan 16 putaran. Sebenarnya, 8 putaran sudah cukup untuk membuat cipher teks sebagai fungsi acak dari setiap bit plain teks dan setiap bit cipher teks. Dari penelitian, DES dengan jumlah putaran yang kurang dari 16 ternyata dapat lebih mudah dipecahkan dengan *known-plaintext attack* daripada dengan *brute force attack.*

3. Kotak-S (S-BOX)

   Pengisian kotak-S DES masih menjadi misteri tanpa ada alasan mengapa memilih nilai-nilai di dalam kotak itu.

2. Kriptografi Knapsack Merkle-Hellman

Kriptografi Knapsack pertama kali ditemukan oleh Merkle dan Hellman pada tahun 1978. Kriptografi Knapsack bersandar pada Problem Subset Sum.

Definisi 3.1.

Problem Subset Sum. Jika $I = (s_1, s_2, .., s_n, T)$ dimana $s_1, s_2, .., s_n$ dan T adalah bilangan bulat positif. n disebut sizes dan T disebut Target sum.

Masalah yang timbul adalah apakah ada vector $x = (x_1, x_2, .., x_n)$ dimana $x_i \in \{0, 1\}$ sedemikian sehingga $\sum_{i=1}^{n} x_i S_i = T$.

Algoritma yang dapat digunakan untuk memecahkan problem subset sum dapat dituliskan sebagai berikut :

Input :

Suatu barisan Superincreasing (sangat naik) $(s_1, s_2, .., s_n)$ dan bilangan bulat T yang merupakan jumlah dari subset s_i.

Output :

$(x_1, x_2, .., x_n)$ dimana $x_i \in \{0, 1\}$ sedemikian sehingga $\sum_{i=1}^{n} x_i S_i = T$.

Contoh :

Diberikan himpunan Superincreasing (3, 7, 12, 30, 60, 115) dan $T = 82$. Maka vector $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ adalah :

$82 < 115 \rightarrow x_6 = 0,$

$82 > 60 \rightarrow x_5 = 1, 82-60 = 22$

$22 < 30 \rightarrow x_4 = 0,$

$22 > 12 \rightarrow x_3 = 1, 22-12 = 10$

$10 > 7 \rightarrow x_2 = 1, 10-7 = 3$

$3 > 3 \rightarrow x_1 = 1, 3-3 = 0$

Maka vector $x = (111010)$

Langkah-langkah Kriptografi Knapsack adalah sebagai berikut :

1. Pembangkitan Kunci Umum dan Kunci Pribadi.
   - Ditentukan sebuah barisan superincreasing.

     Untuk menentukan barisan superincreasing, pertama dipilih bilangan inisial (terkecil). Selanjutnya dipilih bilangan selanjutnya dengan dimana

bilangan yang lebih besar daripada yang pertama. Kemudian dipilih bilangan yang lebih besar daripada penjumlahan bilangan pertama dan kedua. Selanjutnya dipilih bilangan-bilangan baru yang lebih besar daripada jumlah semua bilangan yang sebelumnya dipilih.

- Kemudian dipilih bilangan prima p dimana $p > \sum_{i=1}^{n} S_i$
- Dipilih bilangan acak *a*, dimana $1 < a < p-1$. Dan *a* relatif prima terhadap *p*.
- Barisan superincreasing *s*, bilangan *p* dan *a* selanjutnya disebut kunci pribadi yang bersifat rahasia.
- Setelah ditentukan kunci pribadi selanjutnya akan ditentukan kunci umum.

  Ditentukan barisan t dimana $t_i = a * s_i \ mod \ p$

2. Proses Enkripsi
   - Pesan plain teks dikonversi ke dalam bentuk deretan biner (0, 1) sepanjang jumlah kunci umum, sehingga membentuk vector $x = (x_1, x_2, .., x_n)$
   - Nilai chipper teks adalah $\sum_{i=1}^{n} x_i t_i$
3. Proses Dekripsi
   - Dicari bilangan *z* sebagai target sum dari chipper teks yang nantinya dikonversi menjadi bilangan biner dengan kunci pribadi
   - $z = a^{-1} y \ mod \ p$
   - Karena a^{-1} belum diketahui, maka nilai a^{-1} dapat dicari dengan melihat nilai *a*.

Karena $a \equiv 1 \pmod{p}$ maka $aa^{-1} \equiv 1 \pmod{p}$

$aa^{-1} \equiv 1 \pmod{p}$

$aa^{-1} \equiv 1 + kp$

$a^{-1} = \frac{1+kp}{a}$, dengan $k = 1, 2, 3, \ldots$ sehingga membentuk a^{-1} sebagai bilangan bulat.

- Setelah ditemukan nilai *z*, maka dicari nilai vector $x = (x_1, x_2, .., x_n)$ dimana $z = \sum_{i=1}^{n} x_i s_i$
- Nilai vector $x = (x_1, x_2, .., x_n)$ adalah nilai plain teks awal

Implementasi Kriptografi Knapsack

Diberikan plain teks=KRIPTOGRAFI

Ditentukan $s = (1, 6, 13, 27, 60, 135, 280, 567)$

Dipilih bilangan prima $p = 1093$

Dipilih bilangan $a = 89$ dimana $gcd(a, p) = 1$.

Plain teks diubah menjadi barisan bilangan biner sepanjang jumlah *s*.

K = 75 = 01001011

R = 82 = 01010010

I = 73 = 01001001

P = 80 = 01010000

T = 84 = 01010100

O = 79 = 01001111

G = 71 = 01000111

R = 82 = 01010010

A = 65 = 01000001

F = 70 = 01000110

I = 73 = 01001001

Kemudian akan ditentukan kunci umum $t(t_1, t_2, \dots, t_n)$

$t_i = a * s_i \ mod \ p$

$t_1 = a * s_1 \ mod \ p \rightarrow t_1 = 89 * 1 \ mod \ 1093 = 89 \ mod \ 1093 = 89$

$t_2 = a * s_2 \ mod \ p \rightarrow t_2 = 89 * 6 \ mod \ 1093 = 534 \ mod \ 1093 = 534$

$t_3 = a * s_3 \ mod \ p \rightarrow t_3 = 89 * 13 \ mod \ 1093 = 1157 \ mod \ 1093 = 64$

$t_4 = a * s_4 \ mod \ p \rightarrow t_4 = 89 * 27 \ mod \ 1093 = 2403 \ mod \ 1093 = 217$

$t_5 = a * s_5 \ mod \ p \rightarrow t_5 = 89 * 60 \ mod \ 1093 = 5340 \ mod \ 1093 = 968$

$t_6 = a * s_6 \ mod \ p \rightarrow t_6 = 89 * 135 \ mod \ 1093 = 12015 \ mod \ 1093 = 1085$

$t_7 = a * s_7 \ mod \ p \rightarrow t_7 = 89 * 280 \ mod \ 1093 = 24920 \ mod \ 1093 = 874$

$t_8 = a * s_8 \ mod \ p \rightarrow t_8 = 89 * 567 \ mod \ 1093 = 50463 \ mod \ 1093 = 185$

Maka kunci umum $t = (89, 534, 64, 217, 968, 1085, 874, 185)$

Proses enkripsi : $ci = \sum_{i=1}^{n} x_i t_i$

K = 01001011 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (1 * 968) + (0 * 1085) + (1 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 968 + 0 + 874 + 185 = 2561

R = 01010010 = (0 * 89) + (1 * 534) + (0 * 64) + (1 * 217) + (0 * 968) + (0 * 1085) + (1 * 874) + (0 * 185)

= 0 + 534 + 0 + 217 + 0 + 0 + 874 + 0 = 1625

I = 01001001 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (1 * 968) + (0 * 1085) + (0 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 968 + 0 + 0 + 185 = 1687

P = 01010000 = (0 * 89) + (1 * 534) + (0 * 64) + (1 * 217) + (0 * 968) + (0 * 1085) + (0 * 874) + (0 * 185)

= 0 + 534 + 0 + 217 + 0 + 0 + 0 + 0 = 751

T = 01010100 = (0 * 89) + (1 * 534) + (0 * 64) + (1 * 217) + (0 * 968) + (1 * 1085) + (0 * 874) + (0 * 185)

= 0 + 534 + 0 + 217 + 0 + 1085 + 0 + 0 = 1836

O = 01001111 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (1 * 968) + (1 * 1085) + (1 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 968 + 1085 + 874 + 185 = 3646

G = 01000111 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (0 * 968) + (1 * 1085) + (1 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 0 + 1085 + 874 + 185 = 2678

R = 01010010 = (0 * 89) + (1 * 534) + (0 * 64) + (1 * 217) + (0 * 968) + (0 * 1085) + (1 * 874) + (0 * 185)

= 0 + 534 + 0 + 217 + 0 + 0 + 874 + 0 = 1625

A = 01000001 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (0 * 968) + (0 * 1085) + (0 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 0 + 0 + 0 + 185 = 719

F = 01000110 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (0 * 968) + (1 * 1085) + (1 * 874) + (0 * 185)

= 0 + 534 + 0 + 0 + 0 + 1085 + 874 + 0 = 2493

I = 01001001 = (0 * 89) + (1 * 534) + (0 * 64) + (0 * 217) + (1 * 968) + (0 * 1085) + (0 * 874) + (1 * 185)

= 0 + 534 + 0 + 0 + 968 + 0 + 0 + 185 = 1687

Jadi didapatkan chipper teks sebagai berikut :

K	R	I	P	T	O	G	R	A	F	I
2561	1625	1687	751	1836	3646	2678	1625	719	2493	1687

Proses dekripsi : $z_i = a^{-1} c_i \ mod \ p$, dimana z_i dikonversi ke dalam bentuk barisan biner menggunakan kunci pribadi $s = (1, 6, 13, 27, 60, 135, 280, 567)$.

Karena a^{-1} belum diketahui, maka dicari nilai a^{-1} terlebih dahulu.

$$a^{-1} = \frac{1 + kp}{a} = \frac{1 + 32 * 1093}{89} = \frac{34977}{89} = 393$$

Dengan $k = 32$, didapatkan $a^{-1} = 393$.

$z_1 = a^{-1} c_1 \ mod \ p = 393 . 2561 \ mod \ 1093 = 1006473 \ mod \ 1093 = 913$

$= (1 . 567) + (1 . 280) + (0 . 135) + (1 . 60) + (0 . 27) + (0 . 13) + (1 . 6) + (0 . 1)$

$= 11010010$

$= 01001011$

$z_2 = a^{-1} c_2 \ mod \ p = 393 . 1625 \ mod \ 1093 = 638625 \ mod \ 1093 = 313$

$= (0 . 567) + (1 . 280) + (0 . 135) + (0 . 60) + (1 . 27) + (0 . 13) + (1 . 6) + (0 . 1)$

$= 01001010$

$= 01010010$

z_3

$= a^{-1}c_3 \; mod \; p = 393.1687 \; mod \; 1093 = 662991 \; mod \; 1093 = 633$

$= (1.567) + (0.280) + (0.135) + (1.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 10010010$

$= 01001001$

$z_4 \; = a^{-1}c_4 \; mod \; p = 393.751 \; mod \; 1093 = 295143 \; mod \; 1093 = 33$

$= (0.567) + (0.280) + (0.135) + (0.60) + (1.27) + (0.13) + (1.6) + (0.1)$

$= 00001010$

$= 01010000$

z_5

$= a^{-1}c_5 \; mod \; p = 393.1836 \; mod \; 1093 = 721548 \; mod \; 1093 = 168$

$= (0.567) + (0.280) + (1.135) + (0.60) + (1.27) + (0.13) + (1.6) + (0.1)$

$= 00101010$

$= 01010100$

$z_6 \; = a^{-1}c_6 \; mod \; p = 393.3646 \; mod \; 1093 = 1432878 \; mod \; 1093 = 1048$

$= (1.567) + (1.280) + (1.135) + (1.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 11110010$

$= 01001111$

z_7

$= a^{-1}c_7 \ mod \ p = 393.2678 \ mod \ 1093 = 1052454 \ mod \ 1093 = 988$

$= (1.567) + (1.280) + (1.135) + (0.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 11100010$

$= 01000111$

z_8

$= a^{-1}c_8 \ mod \ p = 393.1625 \ mod \ 1093 = 638625 \ mod \ 1093 = 313$

$= (0.567) + (1.280) + (0.135) + (0.60) + (1.27) + (0.13) + (1.6) + (0.1)$

$= 01001010$

$= 01010010$

$z_9 = a^{-1}c_9 \ mod \ p = 393.719 \ mod \ 1093 = 282567 \ mod \ 1093 = 573$

$= (1.567) + (0.280) + (0.135) + (0.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 10000010$

$= 01000001$

$z_{10} = a^{-1}c_{10} \ mod \ p = 393.2493 \ mod \ 1093 = 979749 \ mod \ 1093 = 421$

$= (0.567) + (1.280) + (1.135) + (0.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 01100010$

$= 01000110$

$z_{11} = a^{-1}c_{11} \ mod \ p = 393.1687 \ mod \ 1093 = 662991 \ mod \ 1093 = 633$

$= (1.567) + (0.280) + (0.135) + (1.60) + (0.27) + (0.13) + (1.6) + (0.1)$

$= 10010010$

$= 01001001$

Konversi bilangan ke plain teks

$z_1 = 01001011 = 75$ = K

$z_2 = 01010010 = 82$ = R

$z_3 = 01001001 = 73$ = I

$z_4 = 01010000 = 80$ = P

$z_5 = 01010100 = 84$ = T

$z_6 = 01001111 = 79$ = O

$z_7 = 01000111 = 71$ = G

$z_8 = 01010010 = 82$ = R

$z_9 = 01000001 = 65$ = A

$z_{10} = 01000110 = 70$ = F

$z_{11} = 01001001 = 73$ = I

plain teks awal telah didapat=KRIPTOGRAFI

BAB IV

PENUTUP

4.1 Kesimpulan

Perumusan RSA Coding didasarkan pada pengembangan Teorema Euler. Memanfaatkan sifat-sifat bilangan prima untuk membangkitkan pasangan kunci umum dan kunci pribadi.

RSA Coding mempunyai kelebihan dan kekurangan. Kelebihan dari RSA Coding antara lain :

3. Jumlah kunci lebih sedikit dari kriptografi kunci simetris.
4. Memenuhi aspek autentifikasi, karena chipper teks hanya dapat didekripsi dengan kunci pribadi pengirim. Jadi pengirim tidak dapat menyangkal bahwa pesan tersebut dikirim oleh dirinya.

Keamanan RSA Coding terletak pada sulitnya pemfaktoran sebuah bilangan hasil perkalian dari dua bilangan prima yang besar menjadi dua bilangan prima yang sesuai. Karena belum ada cara yang efektif.

4.2 Saran

Karena keamanan RSA Coding terletak pada sulitnya pemfaktoran bilangan, maka pemilihan kunci tepat juga menentukan keamanan RSA Coding.

Meski demikian, tetap harus dipertimbangkan nilai dari pesan yang dirahasiakan dan jangka waktu keperluan akan kerahasiaan pesan tersebut.