BAB 3
KRIPTOGRAFI RSA

3.1 Sistem ASCII

Sebelumnya, akan dijelaskan terlebih dahulu Sistem ASCII sebagai system standar pengkodean dalam pertukaran informasi yaitu Sistem ASCII. Plainteks yang akan dienkripsi dengan algoritma RSA merupakan angka-angka, sedangkan pesan yang dikirimkan biasanya berbentuk teks atau tulisan. Oleh karena itu, dibutuhkan suatu kode yang sifatnya universal untuk mengubah pesan teks menjadi plainteks yang berbentuk angka. ASCII (American Standard Code for Information Interchange) atau Kode Standar Amerika untuk Pertukaran Informasi adalah suatu standar internasional dalam kode huruf dan symbolyang bersifat universal. ASCII selalu digunakan oleh computer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenranya memiliki komposisi bilangan biner 8 bit dimulai dari 0000 0000 sampai 1111 1111. Total kombinasi yang dihasilkan sebanyak 256 dimulai dari 0 sampai 255. Kode ASCII terdiri dari karakter umum yang biasa digunakan dalam penulisan data. Kode ASCII yang biasa digunakan dalam penulisan data dapat dilihat dalam tabel berikut :

Tabel 3.1 Kode ASCII

Karakter	Kode ASCII
Spasi	32
!	33
“	34
#	35
$	36
%	37
&	38
‘	39
(	40
)	41

Karakter	Kode ASCII
@	64
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73

Karakter	Kode ASCII
`	96
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105

*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63

J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93
^	94
_	95

j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126

3.2 Algoritma Kriptografi RSA

RSA merupakan salah satu algoritma kriptografi kunci publik.Algoritma ini adalah algoritma pertama yang cocok dalam melakukan *digital signature*.Saat ini algoritma RSA merupakan algoritma yang paling sering dipakai dari algoritma kunci publik lainnya.Algoritma RSA dikembangkan pertama kali oleh Ron Rivest, Adi Shamir, dan Len Adleman dari Massachussets Institute of Technology pada tahun 1978. Nama RSA sendiri diambil dari nama ketiga peneliti tersebut yaitu, (R)ivest, (S)hamir, dan (A)dleman. RSA merupakan algoritma kunci publik

yang memilik dua kunci yaitu kunci publik (public key) dan kunci pribadi (private key).

RSA terbagi menjadi tiga proses, yaitu pembangkitan kunci, enkripsi dan dekripsi. Dasar proses enkripsi dan dekripsi pada algoritma RSA yaitu konsep bilangan prima dan aritmatika modulo. Kunci enkripsi tidak dirahasiakan dan diberikan kepada umum (disebut kunci publik), sedangkan kunci untuk dekripsi bersifat rahasia (disebut kunci pribadi).

Untuk menemukan kunci dekripsi, dilakukan dengan cara memfaktorkan bilangan bulat menjadi faktor-faktor primanya. Namun, memfaktorkan bilangan bulat menjadi faktor primanya tidak mudah karena belum ada cara yang efisien untuk melakukan pemfaktoran. Cara yang paling mungkin dilakukan adalah dengan pohon faktor. Namun semakin besar bilangan yang akan difaktorkan maka semakin lama pula waktu yang dibutuhkan untuk menyelesaikannya. Jadi semakin besar bilangan yang akan difaktorkan, semakin sulit pemfaktorannya, semakin kuat pula algoritma RSA. Oleh karena itu, dalam menggunakan algoritma RSA dianjurkan menggunakan bilangan yang sangat besar agar keamanannya dapat terjamin.

Besaran-besaran yang digunakan pada algoritma RSA 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 (kuni dekripsi) (rahasia)
6. m (plainteks) (rahasia)
7. c (chiperteks) (tidak rahasia)

Algoritma RSA didasarkan pada Teorema Euler yang menyatakan bahwa

$$a^{\varphi(n)} \equiv 1 \; mod(n) \quad (1)$$

yang dalam hal ini:

1. a relatif prima terhadap n.
2. $\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \dots (1 - \frac{1}{p_r})$dengan$p_1, p_2, \dots, p_r$ adalah faktor prima dari n.

Berdasarkan sifat $a^k \equiv b^k \ (mod n)$ dengan $k \geq 1$ bilangan bulat , maka persamaan (1) menjadi

$$a^{k\varphi(n)} \equiv 1^k mod(n) \Leftrightarrow a^{k\varphi(n)} \equiv 1 \ mod(n) \quad (2)$$

Ganti a dengan m sehingga

$$m^{k\varphi(n)} \equiv 1 \ mod(n) \quad (3)$$

Berdasarkan sifat $ac \equiv bc \ (mod n)$ dan persamaan (3) dikalikan dengan m diperoleh

$$m^{k\varphi(n)+1} \equiv m \ (mod n) \quad (4)$$

Misalkan dipilih

$$e.d \equiv 1 \ (mod \varphi(n)) \quad (5)$$

atau

$$e.d = k\varphi(n) + 1 \quad (6)$$

Substitusi (6) ke persamaan (4) menjadi

$$m^{e.d} \equiv m(mod n) \ (7)$$

Persamaan (7) dapat ditulis kembali menjadi

$$(m^e)^d \equiv m(mod n) \ (8)$$

yang berarti bahwa ketika m dipangkatkan e dan dipangkatkan lagi dengan d menghasilkan m lagi.

Berdasarkan persamaan (8), dirumuskan enkripsi dan enkripsi yaitu:

$$c \equiv m^e (mod n) \ (9)$$

dan

$$m \equiv c^d (mod n) \ (10)$$

3.2.1. Proses Pembangkitan Kunci

Algoritma RSA memiliki dua kunci yang berbeda untuk proses enkripsi dan dekripsi. Dalam menentukan dua bilangan prima sebagai kunci adalah

bilangan prima yang besar, karena pemfaktoran bilangan dari dua bilangan prima yang besar sangat sulit, sehingga keamanan pesan lebih terjamin.

Pasangan kunci adalah elemen penting dari algoritma RSA.Berikut ini langkah-langkah dalam membangkitkan dua kunci algoritma RSA.

1. Pilih dua bilangan prima sembarang, $p$dan q.
2. Hitung $n = p.q$.
3. Hitung $\varphi(n) = (p-1)(q-1)$.
4. Pilih kunci publik e, yang relatif prima terhadap $\varphi(n)$.
5. Bangkitkan kunci pribadi dengan menggunakan $e.d \equiv 1\, mod\, \varphi(n)$.

Hasil dari algoritma tersebut akan menghasilkan dua kunci, yaitu kunci publik (e, n) dan kunci pribadi (d, n).

Contoh 3.1.1.1

Misalkan B akan membangkitkan kunci publik dan kunci pribadi miliknya. B memilih $p = 7$ dan $q = 13$ (keduanya prima). Selanjutnya B menghitung

$$n = 7 \times 13 = 91$$

dan

$$\varphi(n) = (7-1)(13-1) = 72$$

B memilih $e = 5$ karena 5 relatif prima terhadap 72. B mengumumkan nilai $e$dan n.

Selanjutnya B menghitung nilai $d$dengan algoritma Euclid yang diperluas menjadi

$72 = 14.5 + 2$ $\quad n = 1, a_1 = 5, q_1 = 14$

$5 = 2.2 + 1$ $\quad n = 2, a_2 = 2, q_2 = 2$

$2 = 2.1$ $\quad n = 3, a_3 = 1, q_3 = 2$

$$t_2 = t_0 - q_1.t_1 = 0 - 14(1) = -14 = 58$$

$$t_3 = t_1 - q_2.t_2 = 1 - 2.(-14) = 29$$

Karena $e.d \equiv 1\, mod\, \varphi(n)$ dapat ditulis menjadi $e^{-1}\, mod\, \varphi(n) = d$, maka didapat$5^{-1}\, mod\, 72 = 29$. Sehingga diperoleh $d = 29$. Ini adalah kunci pribadi untuk mendekripsikan pesan dan harus dirahasiakan oleh B. Dari perhitungan tersebut didapat kunci publik dan kunci pribadi berturut-turut adalah

$$(e = 5, n = 91)$$

dan

$$(d = 29, n = 91)$$

Contoh 3.1.1.2.

Misalkan B akan membangkitkan kunci publik dan kunci pribadi miliknya. B memilih $p = 47$ dan $q = 71$ (keduanya prima). Selanjutnya B menghitung

$$n = 47 \times 71 = 3337$$

dan

$$\varphi(n) = (47 - 1)(71 - 1) = 3220$$

B memilih $e = 79$ karena 79 relatif prima terhadap 3220. B mengumumkan nilai $e$dan n.

Selanjutnya B menghitung nilai d dengan algoritma Euclid yang diperluas menjadi

$72 = 14.5 + 2$ $\quad n = 1, a_1 = 5, q_1 = 14$

$5 = 2.2 + 1$ $\quad n = 2, a_2 = 2, q_2 = 2$

$2 = 2.1$ $\quad n = 3, a_3 = 1, q_3 = 2$

$$t_2 = t_0 - q_1.t_1 = 0 - 14(1) = -14 = 58$$

$$t_3 = t_1 - q_2.t_2 = 1 - 2.(-14) = 29$$

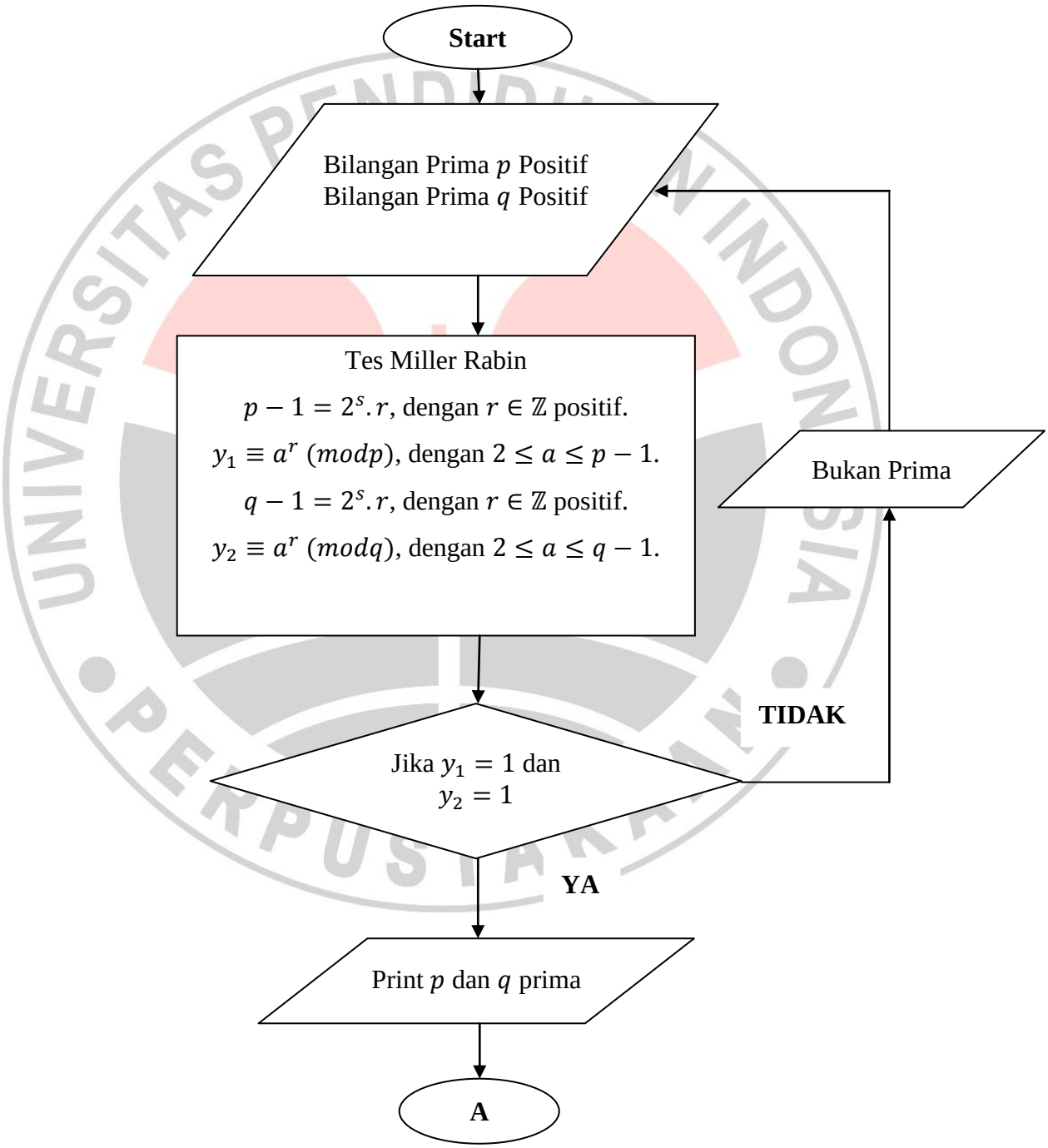
Karena $e.d \equiv 1\, mod\, \varphi(n)$ dapat ditulis menjadi $e^{-1}\, mod\, \varphi(n) = d$, maka didapat$5^{-1}\, mod\, 72 = 29$. Sehingga diperoleh $d = 29$. Ini adalah kunci pribadi untuk mendekripsikan pesan dan harus dirahasiakan oleh B. Dari perhitungan tersebut didapat kunci publik dan kunci pribadi berturut-turut adalah

$$(e = 79, n = 3337)$$

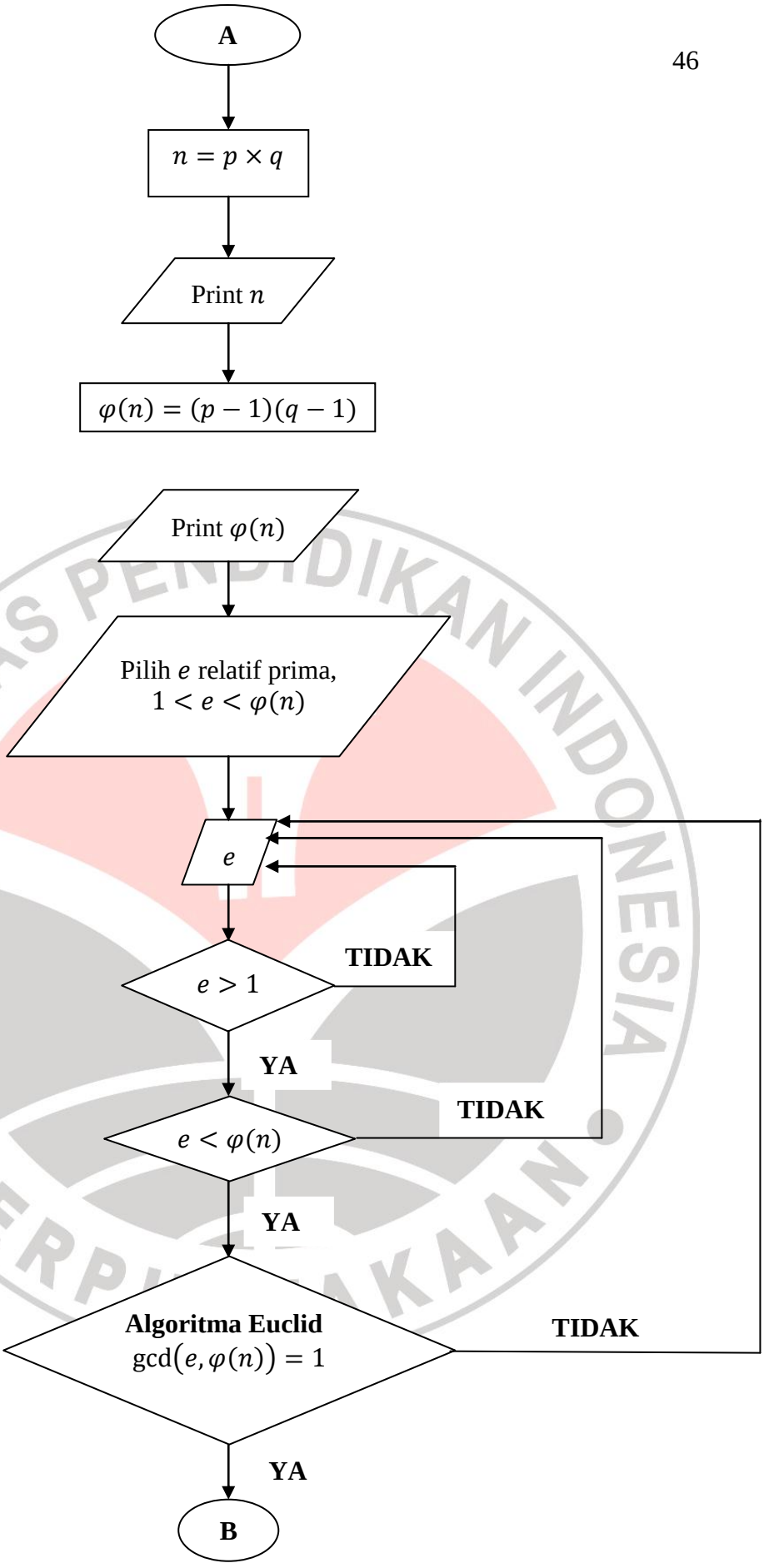
dan

$$(d = 1019, n = 3337)$$

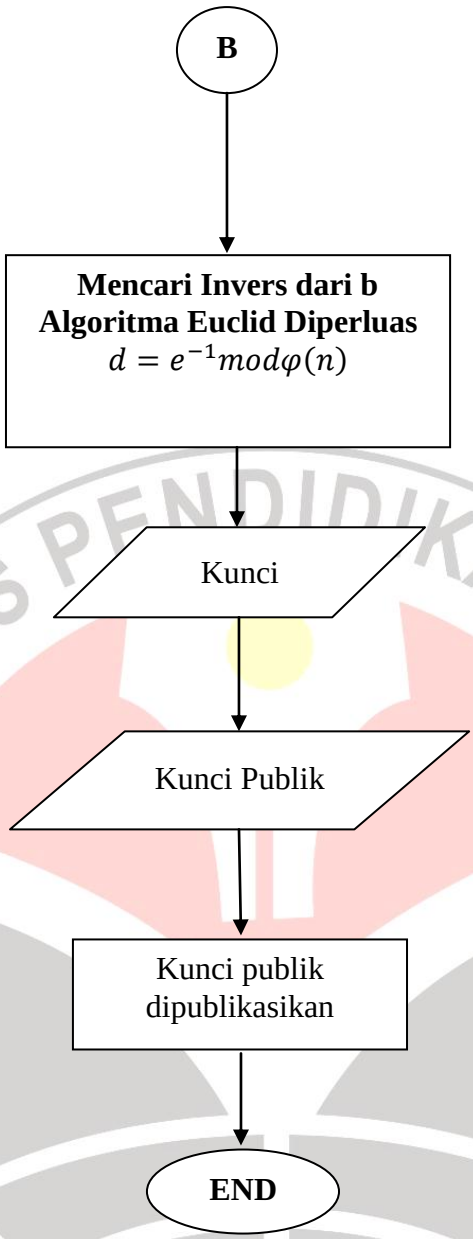
Berikut merupakan flowchart pembangkitan kunci dengan algoritma RSA.



Gambar 3.1 Flowchart Algoritma Pembangkitan Kunci



Gambar 3.2 Flowchart Pembangkitan Kunci Lanjutan



Gambar 3.3 Flowchart Pembangkitan Kunci Lanjutan

3.2.2. Proses Enkripsi

Langkah-langkah dalam melakukan proses enkripsi adalah sebagai berikut:

1. Ambil kunci public penerima pesan, e, dan modulus n.
2. Plainteks dibuat menjadi blok-blok $m_1, m_2, m_3, \ldots$. sedemikian sehingga setiap blok merepresentasikan nilai di selang $[0, n-1]$.
3. Setiap blok $m_i$dienkripsi menjadi blok $c_i$dengan rumus

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

Contoh 3.2.2.1.

Misalkan A akan mengirim pesan ke B. Pesan (Plainteks) yang akan dikirim adalah

$$m = PESAN$$

atau dalam sistem desimal pengkodean ASCII adalah

$$8069836578$$

A memecah m menjadi blok yang lebih kecil, misalkan membagi menjadi 5 blok yang berukuran 2 digit

$$m_1 = 80$$
$$m_2 = 69$$
$$m_3 = 83$$
$$m_4 = 65$$
$$m_5 = 78$$

Nilai-nilai m_i ini masih terletak di selang $[0, 91-1]$ agar transformasi menjadi satu-ke-satu.

A mengetahui kunci publik B adalah $e = 5$dan $n = 91$. A dapat mengenkripsikan setiap blok plainteks sebagai berikut

$$c_1 = 80^5 \ mod \ 91 = 19$$
$$c_2 = 69^5 \ mod \ 91 = 62$$
$$c_3 = 83^5 \ mod \ 91 = 83$$
$$c_4 = 65^5 \ mod \ 91 = 39$$
$$c_5 = 78^5 \ mod \ 91 = 78$$

Jadi chiperteks yang dihasilkan adalah

$$c = 19\ 62\ 83\ 39\ 78$$

Contoh 3.2.2.2.

Misalkan A akan mengirim pesan ke B. Pesan (Plainteks) yang akan dikirim adalah

$$m = HARI\ INI$$

atau dalam sistem desimal pengkodean ASCII adalah

$$7265827332737873$$

A memecah m menjadi blok yang lebih kecil, misalkan membagi menjadi 6 blok yang berukuran 3 digit

$$m_1 = 726$$
$$m_2 = 582$$
$$m_3 = 733$$
$$m_4 = 273$$
$$m_5 = 787$$
$$m_6 = 003$$

Nilai-nilai m_i ini masih terletak di selang $[0, 3337 - 1]$ agar transformasi menjadi satu-ke-satu.

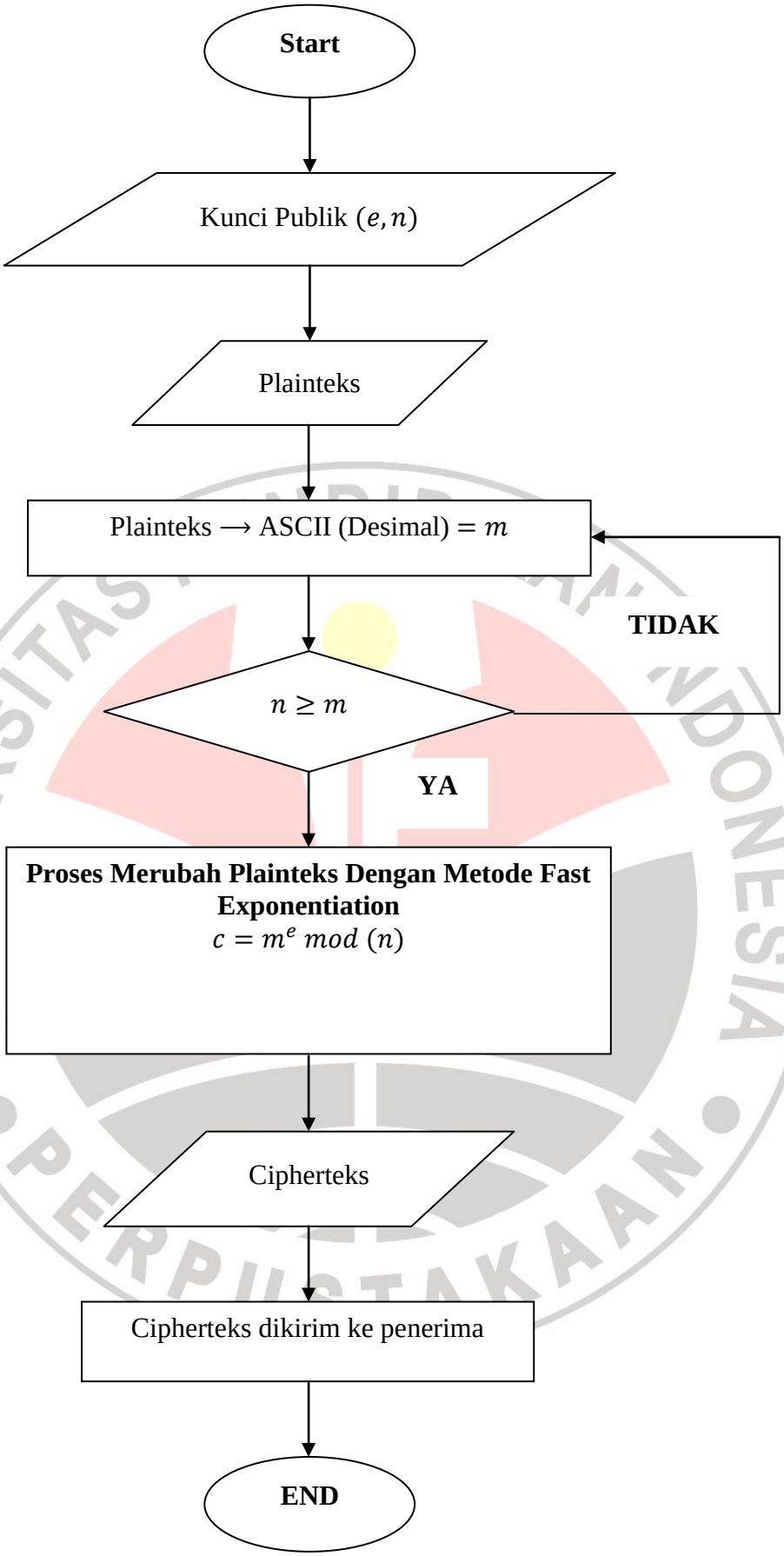
A mengetahui kunci publik B adalah $e = 79$dan $n = 3337$. A dapat mengenkripsikan setiap blok plainteks sebagai berikut

$$c_1 = 726^{79}\ mod\ 3337 = 215$$
$$c_2 = 582^{79}\ mod\ 3337 = 776$$
$$c_3 = 733^{79}\ mod\ 3337 = 1743$$
$$c_4 = 273^{79}\ mod\ 3337 = 933$$
$$c_5 = 787^{79}\ mod\ 3337 = 1731$$
$$c_6 = 003^{79}\ mod\ 3337 = 158$$

Jadi chiperteks yang dihasilkan adalah

$$c = 215\ 776\ 1743\ 933\ 1731\ 158$$

Berikut merupakan flowchart enkripsi pesan dengan algoritma RSA.



Gambar 3.4 Flowchart Algoritma Enkripsi

3.2.3. Proses Dekripsi

Langkah-langkah dalam melakukan proses dekripsi adalah sebagai berikut:

1. Setiap blok chiperteks c_i didekripsi kembali menjadi blok m_i dengan rumus

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

2. Blok-blok $m_1, m_2, m_3, \ldots$ diubah kembali menjadi bentuk huruf dengan kode ASCII.

Contoh 3.2.3.1.

B akan mendekripsi pesan dengan menggunakan kunci pribadi $(d = 59, n = 91)$. Blok-blok chiperteks didekripsikan dengan cara

$$m_1 = 19^{29} \ mod \ 91 = 80$$
$$m_2 = 62^{59} \ mod \ 91 = 69$$
$$m_3 = 83^{59} \ mod \ 91 = 83$$
$$m_4 = 39^{59} \ mod \ 91 = 65$$
$$m_5 = 78^{59} \ mod \ 91 = 78$$

Akhirnya diperoleh plainteks semula yaitu $m = 8069836578$ yang dalam sistem karakter pengkodean ASCII $m = PESAN$

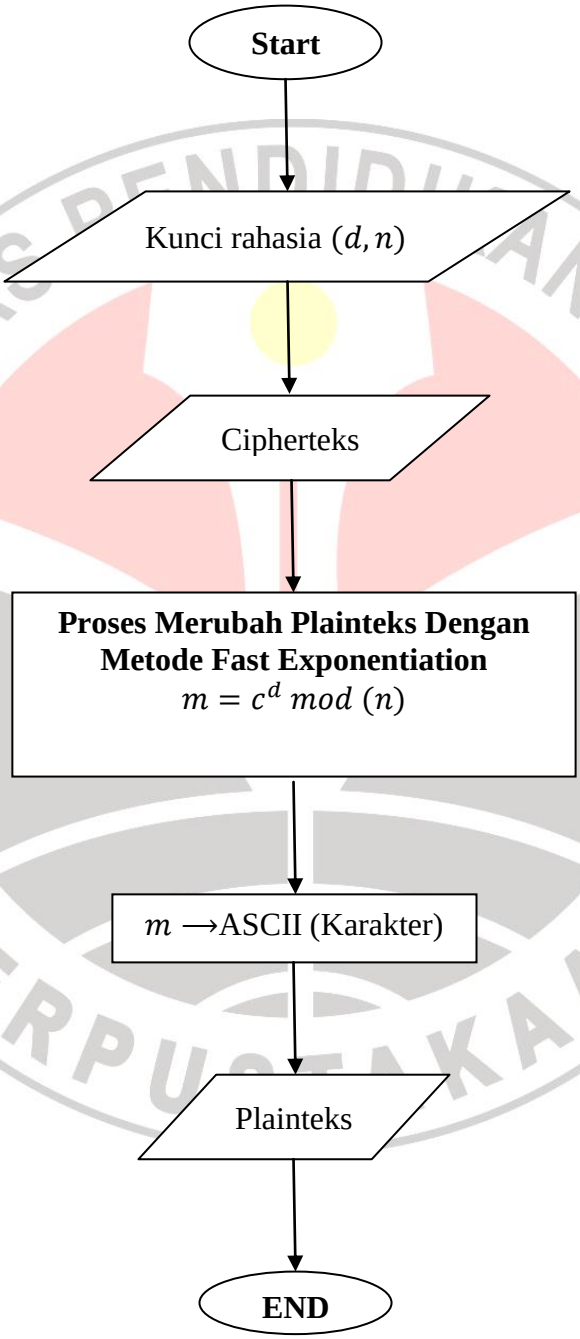
Contoh 3.2.3.2.

B akan mendekripsi pesan dengan menggunakan kunci pribadi $(d = 1019, n = 3337)$. Blok-blok chiperteks didekripsikan dengan cara

$$m_1 = 215^{1019} \ mod \ 3337 = 726$$
$$m_2 = 776^{1019} \ mod \ 3337 = 582$$
$$m_3 = 1743^{1019} \ mod \ 3337 = 733$$
$$m_4 = 933^{1019} \ mod \ 3337 = 273$$
$$m_5 = 1731^{1019} \ mod \ 3337 = 787$$
$$m_6 = 158^{1019} \ mod \ 3337 = 003$$

Akhirnya diperoleh plainteks semula yaitu $m = 7265827332737873$ yang dalam sistem karakter pengkodean ASCII $m = HARI\ INI$

Berikut merupakan flowchart dekripsi pesan dengan algoritma RSA.



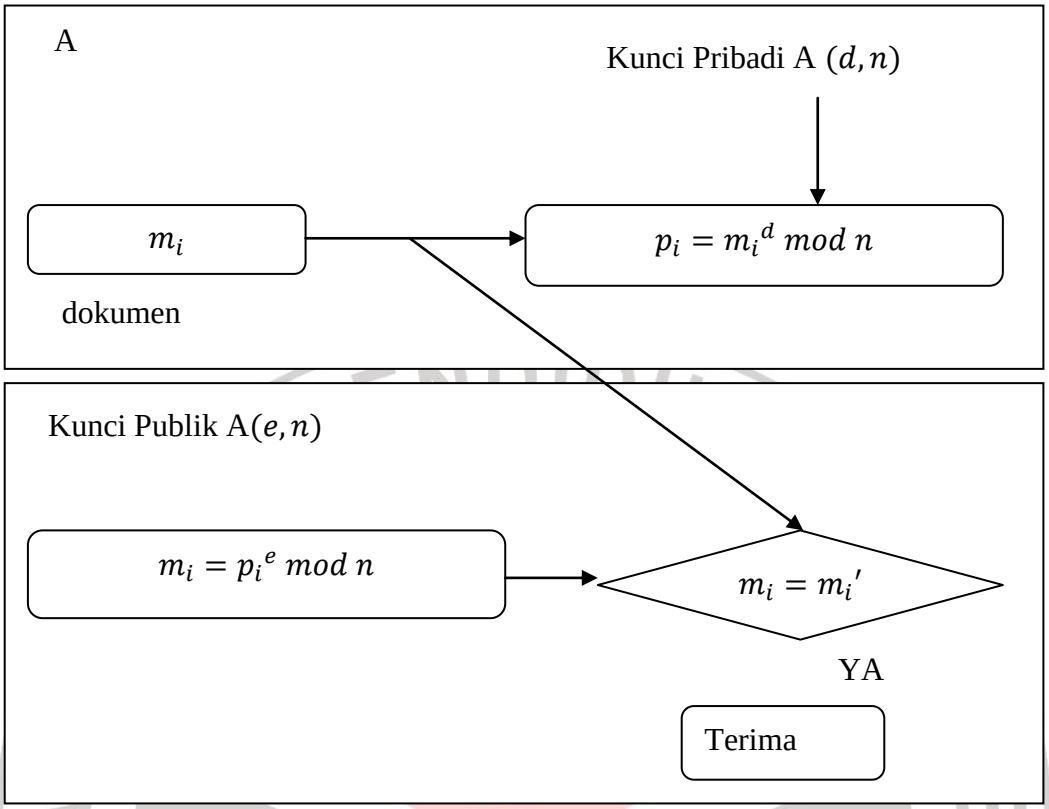
Gambar 3.5 Flowchart Proses Dekripsi

3.3 Digital Signature Algoritma Kriptografi RSA

Sistem Kriptografi Algoritma Kriptografi RSA dapat dimodifikasi sehingga memenuhi sistem digital signature.Ada beberapa perbedaan dari algoritma RSA yang dibahas di awal, dimana pihak pembentuk kunci merupakan si pengirim pesan, berbeda dengan algoritma RSA, pembentukan kunci dilakukan oleh penerima pesan. Perbedaan lainnya terletak pada pembuatan nilai tanda tangan dan verifikasi nilai tanda tangan, dengan proses enkripsi dan dekripsi pesan pada algoritma RSA. Jika pada enkripsi pesan digunakan kunci public, dan proses dekripsi pesan menggunakan kunci pribadi, sebaliknya pada proses pembuatan nilai tanda tangan justru menggunakan kunci pribadi, sedangkan verifikasi pesan menggunakan kunci public. Berikut merupakan konsep serta algoritma digital signature dengan kriptografi RSA.

3.2.1 Konsep Digital Signature RSA

Secara garis besarnya, konsep digital signature dengan algoritma RSA diawali dengan pembangkitan kunci oleh si pengirim pesan.Setelah itu, dengan kunci pribadi yang dimilikinya, si pengirim pesan mencari nilai tanda tangan yang bersesuaian dengan isi pesan, dan mengirimkan nilai pesan beserta nilai tanda tangan ke penerima.Lalu dengan kunci public dari pengirim pesan, penerima mencari nilai pesan dengan menggunakan nilai tanda tangan yang diberikan. Jika nilai pesan dari pengirim sama dengan hasil perhitungan dari penerima, maka pesan tersebut terbukti valid. Berikut skema digital signature demgam algoritma RSA.



Gambar 3.6 Skema Digital Signature Kriptografi RSA

3.2.2 Algoritma Digital Signature Kriptografi RSA

Algoritma digital signature dengan kriptografi RSA terdiri dari tiga proses, yaitu proses pembangkitaan kunci, proses sign digital signature, dan proses verify digital signature. Pada dasarnya, ketiga proses tersebut mempunyai langkah yang sama dengan tiga proses algoritma RSA, hanya terdapat beberapa perbedaan, yaitu pembuat pembangkitan kunci merupakan si pengirim pesan, berbeda dengan kriptografi RSA, dengan penerima pesan sebagai pembangkitan kunci. Selain itu ada perbedaan dari proses sign dan verify dengan proses enkripsi dan dekripsi. Jika dalam proses enkripsi menggunakan kunci public dan proses dekripsi menggunakan kunci pribadi, sedangkan pada proses sign digital signature menggunakan kunci pribadi, dan proses verify menggunakan kunci public. Berikut merupakan algoritma pembuatan digital signature dengan kriptografi RSA.

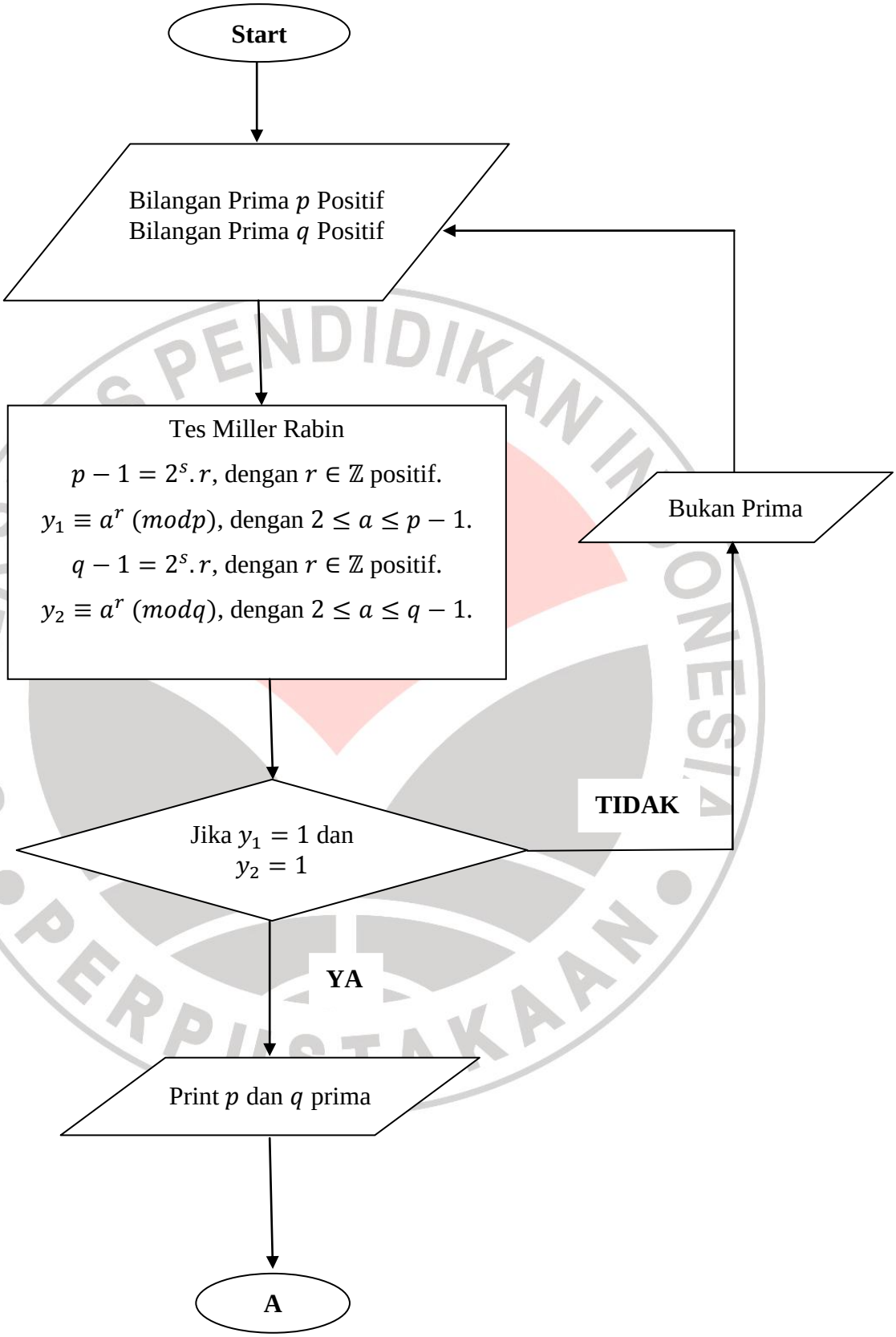
3.2.3.1 Proses Pembangkitan Kunci

Sama halnya dengan proses pembangkitan kunci dengan algoritma RSA, proses pembangkitan kunci untuk digital signature dengan algoritma RSA menghasilkan kunci public dan kunci privat. Perbedaannya terletak pada si pembangkit kunci.Jika pada pembangkitan kunci algoritma RSA, si penerima pesan yang membangkitkan kunci, namun pada digital signature algoritma RSA, si pengirim pesan lah yang membangkitkan kunci. Untuk tahap-tahap pembangkitan kunci digital signature sama dengan pembangkitan algoitma RSA, yaitu:

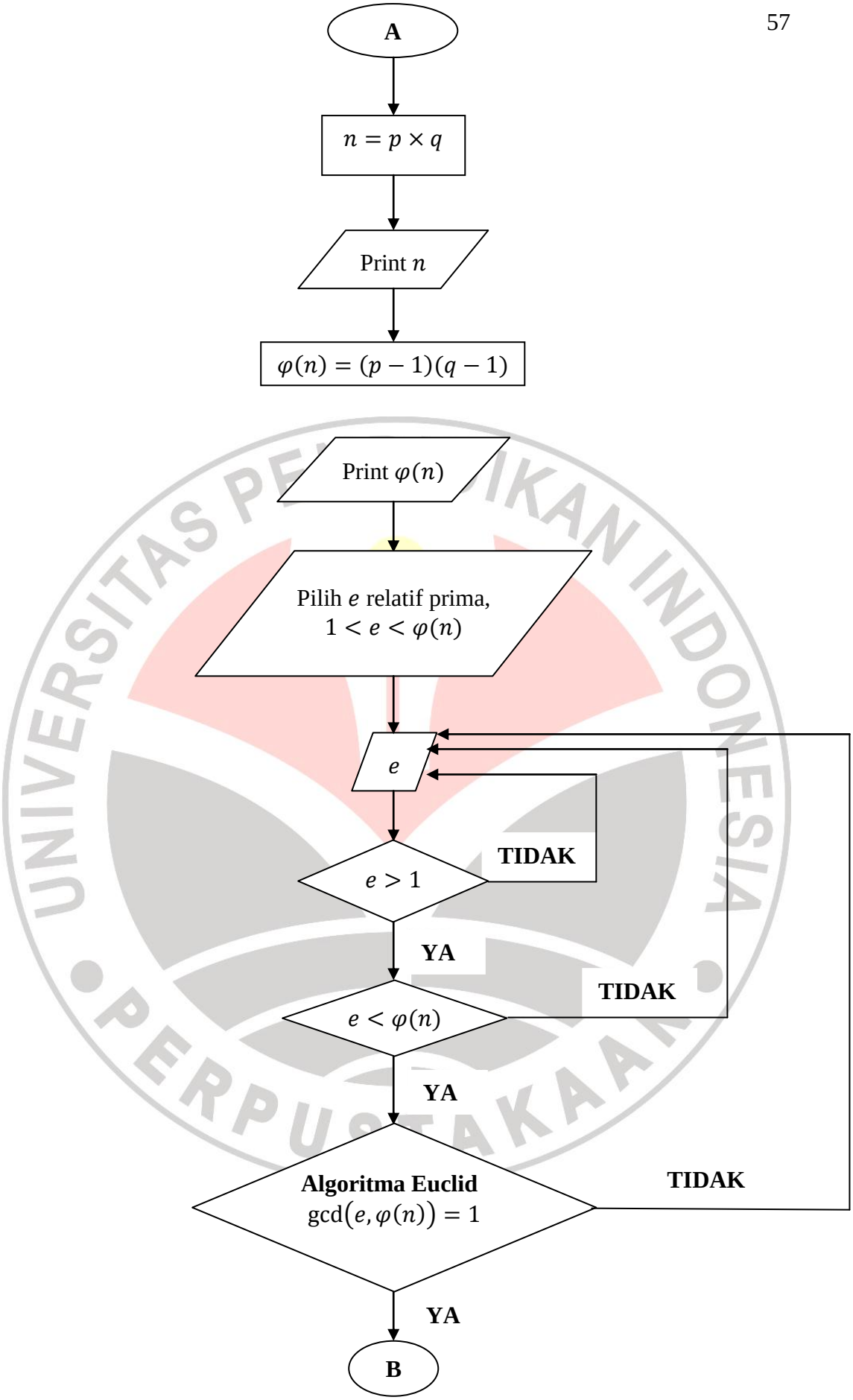
1. Pilih dua bilangan prima sembarang, $p$dan q.
2. Hitung $n = p.q$.
3. Hitung $\varphi(n) = (p-1)(q-1)$.
4. Pilih kunci publik e, yang relatif prima terhadap $\varphi(n)$.
5. Bangkitkan kunci pribadi dengan menggunakan $e.d \equiv 1\ mod\ \varphi(n)$.

Hasil dari algoritma tersebut akan menghasilkan dua kunci, yaitu kunci publik (e, n) dan kunci pribadi (d, n).

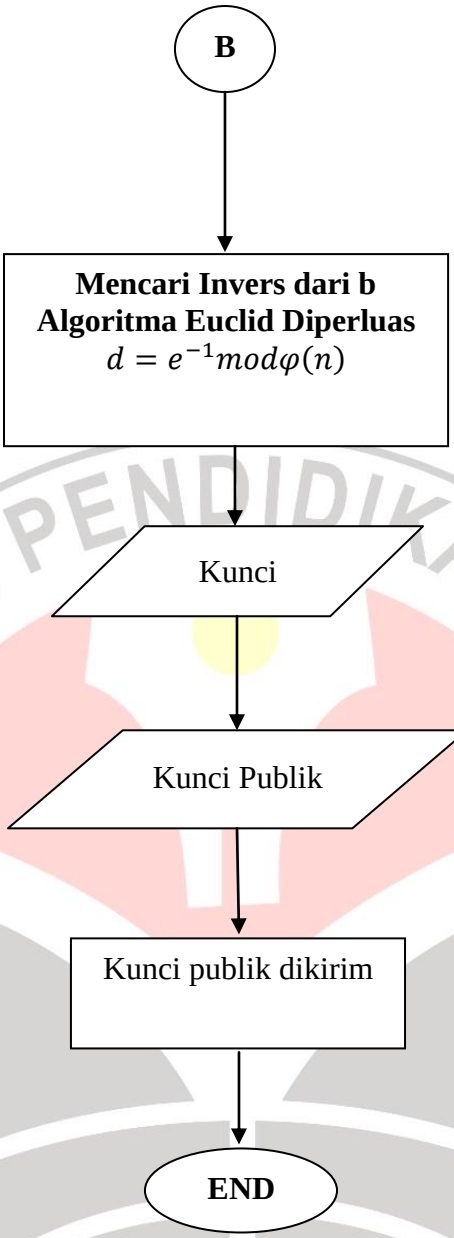
Berikut merupakan flowchart pembangkitan kunci dengan algoritma RSA.



Gambar 3.7 Flowchart Pembangkitan Kunci Digital Signature Kriptografi RSA



Gambar 3.8 Flowchart Pembangkitan Kunci Digital Signature Kriptografi RSA



Gambar 3.9 Flowchart Pembangkitan Kunci Digital Signature Kriptografi RSA

3.2.3.2 Proses Sign Digital Signature

Pada proses sign digital signature, langkah yang dilakukan sama dengan proses enkripsi pada algoritma RSA. Perbedaannya selain terletak di pembuat kunci, hasil yang dihasilkannya merupakan nilai dari digital signature. Dan proses mendapatkan nilai sign digital signature menggunakan kunci privat, berbeda dengan proses enkripsi yang menggunakan kunci publik. Jika pada tanda tangan manual, hanya ada satu tanda tangan untuk satu pengirim, namun pada digital signature bisa banyak nilai tanda tangan karena bergantung pada isi pesan yang akan dikirim. Berikut adalah langkah pembuatan sign digital signature.

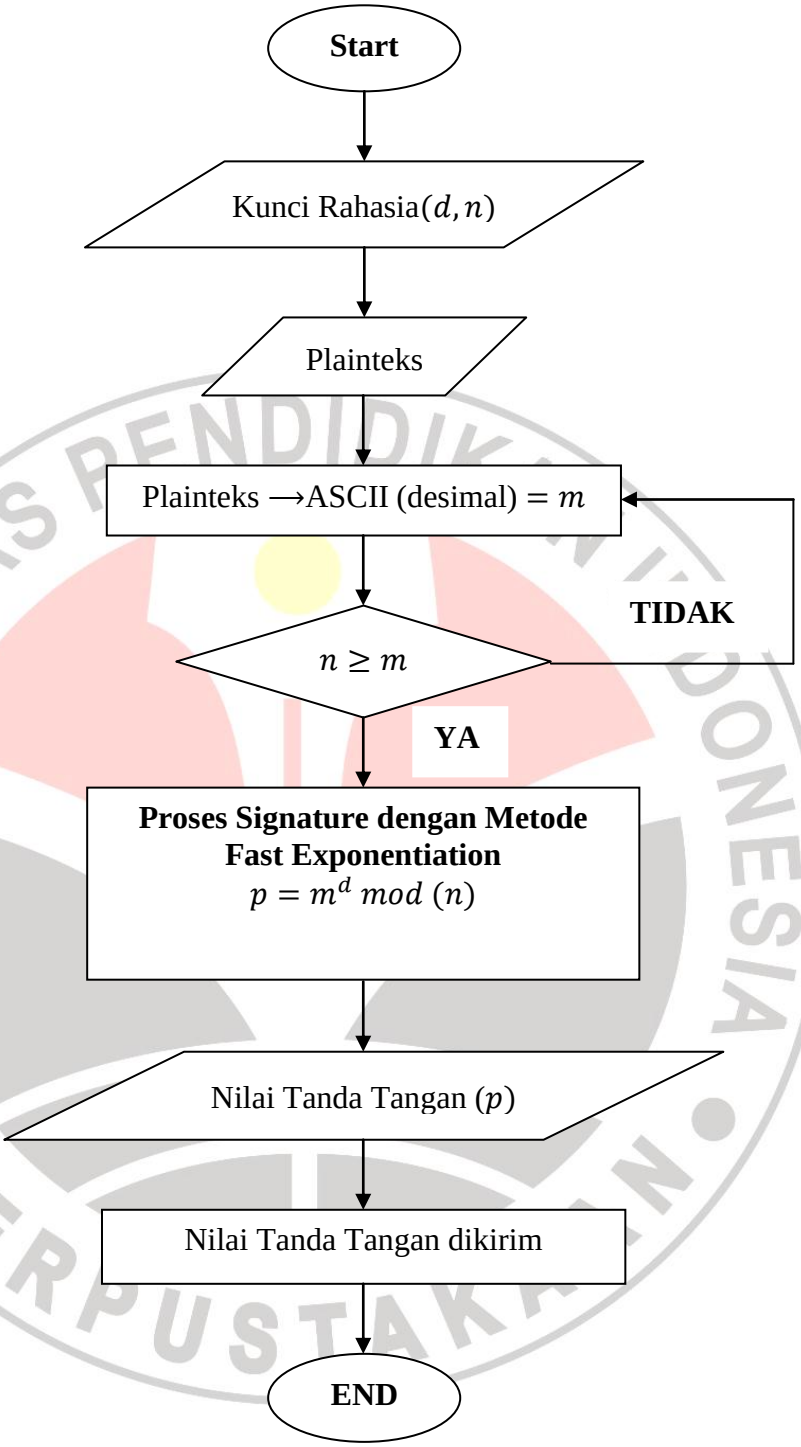
Langkah-langkah dalam melakukan proses enkripsi adalah sebagai berikut:

1. Ambil kunci privat, d, yang telah dibangkitkan pengirim dan modulus n.
2. Plainteks dibuat menjadi blok-blok $m_1, m_2, m_3, \ldots$ sedemikian sehingga setiap blok merepresentasikan nilai di selang $[0, n-1]$.
3. Nilai digital signature p_i didapat dengan menggunakan rumus berikut:

$$p_i = {m_i}^d \; mod \; n$$

4. Pengirim mengirim nilai pesan beserta nilai tanda tangan (m_i, p_i)

Berikut merupakan flowchart proses sign digital signature dengan algoritma RSA.



Gambar 3.10 Flowchart Sign Digital Signature Kriptografi RSA

3.2.3.3 Proses Verifikasi Digital Signature

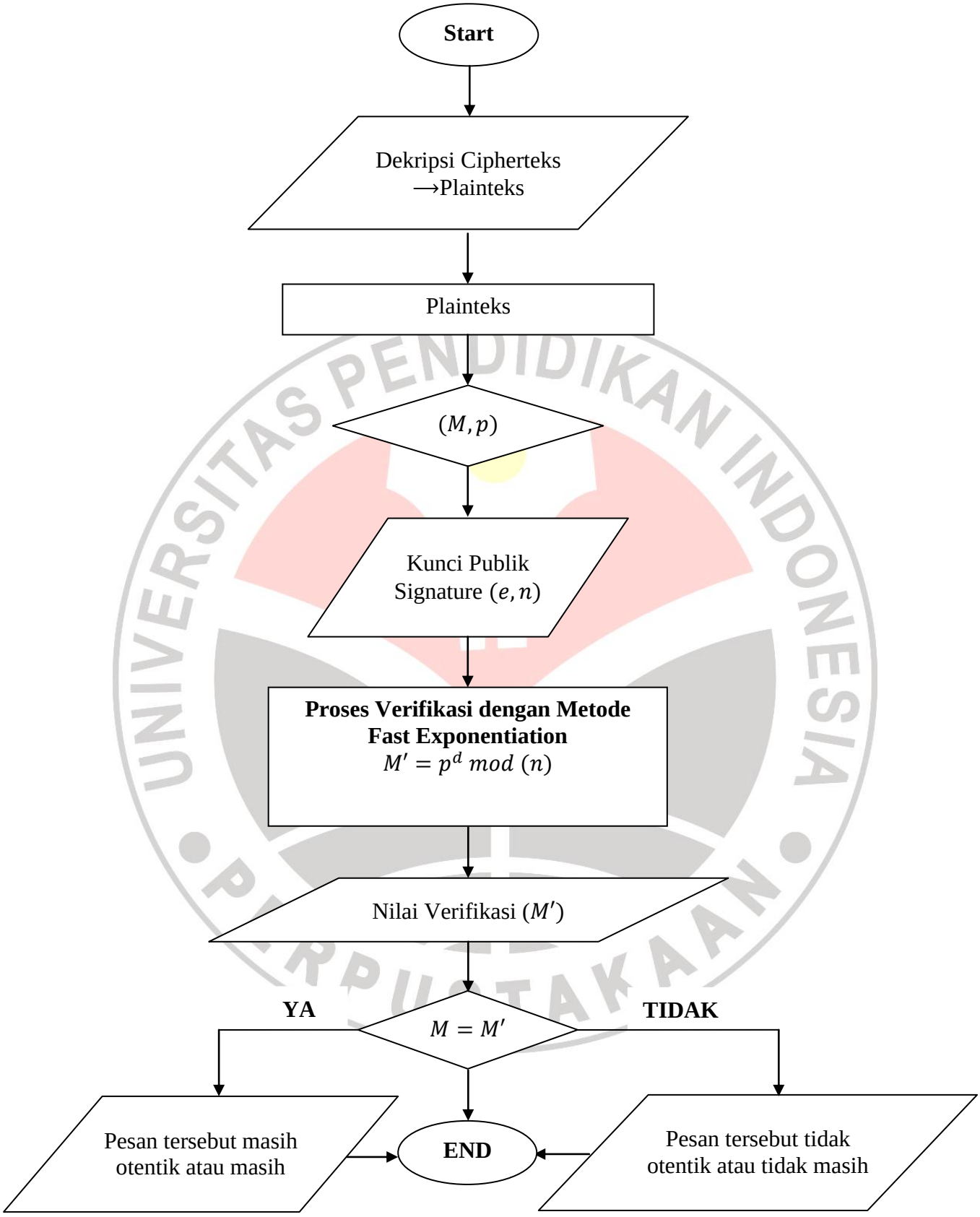
Sama halnya dengan proses sign digital signature, langkah dalam proses verifikasi pun sama dengan proses dekripsi pada algoritma RSA. Namun pada proses verifikasi, menggunakan kunci public dari pengirim, berbeda dengan dekripsi algoritma RSA yang menggunakan kunci pribadi dari penerima itu sendiri. Langkah-langkah dalam melakukan proses verifikasi adalah sebagai berikut:

1. Penerima pesan menerima nilai pesan beserta tanda tangan pesan (m_i, p_i).
2. Setiap blok nilai digital signature p_i diubah menjadi blok m_i' dengan rumus

$$m_i' = {p_i}^e \ mod \ n$$

3. Bandingkan nilai $m_i$dan m_i', apabila $m_i = m_i'$ maka pesan tersebut dapat dipercaya keasliannya.
4. Blok-blok $m_1, m_2, m_3, \ldots$. diubah kembali menjadi bentuk huruf dengan kode ASCII.

Berikut merupakan flowchart proses verifikasi digital signature dengan algoritma RSA.



Gambar 3.11 Flowchart Verifikasi Digital Signature Kriptografi RSA

Berikut adalah contoh dari digital signature dengan menggunakan algoritma RSA.

Contoh 3.2.3.4.

Seorang direktur akan mengirimkan sebuah pesan kepada seluruh pegawainya. Karena pesan tersebut sangat penting, direktur tersebut akan memberikan digital signature pada pesan yang akan ia sampaikan dengan menggunakan digital signature algoritma kriptografi RSA untuk menghindari perubahan isi pesan yang dilakukan pihak tak berwenang. Hal yang pertama dilakukan direktur tersebut yaitu membangkitkan kunci public dan pribadi. Direktur tersebut memilih bilangan $p = 67$dan $q = 73$. Dari dua bilangan prima tersebut, didapat

$$n = 67 \times 73 = 4891$$

dan

$$\varphi(n) = (67 - 1)(73 - 1) = 66 \times 72 = 4752.$$

Direktur tersebut memilih bilangan $e = 7$ karena 7 relatif prima dengan $\varphi(n) = 4752$.

Selanjutnya Direktur tersebut menghitung nilai d dengan algoritma Euclid yang diperluas menjadi

$4752 = 678.7 + 6$ $\quad n = 1, a_1 = 7, q_1 = 678$

$7 = 1.6 + 1$ $\quad n = 2, a_2 = 6, q_2 = 1$

$6 = 6.1$ $\quad n = 3, a_3 = 1, q_3 = 6$

$$t_2 = t_0 - q_1.t_1 = 0 - 678(1) = -678 = 3074$$

$$t_3 = t_1 - q_2.t_2 = 1 - 1.(-678) = 679$$

Dari perhitungan tersebut didapat kunci publik dan kunci pribadi berturut-turut adalah$(e = 7, n = 4891)$dan$(d = 679, n = 4891)$

Misalkan pesan (plainteks) yang akan dikirim direktur tersebut yaitu$m = BESOK\ RAPAT\ JAM\ 9$

atau dalam sistem desimal pengkodean ASCII adalah

$$66698379753282658065843274657732 57$$

A memecah m menjadi blok yang lebih kecil, misalkan membagi menjadi 12 blok yang berukuran 3 digit

$$m_1 = 666$$
$$m_2 = 983$$
$$m_3 = 797$$
$$m_4 = 532$$
$$m_5 = 826$$
$$m_6 = 580$$
$$m_7 = 658$$
$$m_8 = 432$$
$$m_9 = 746$$
$$m_{10} = 577$$
$$m_{11} = 325$$
$$m_{12} = 007$$

Nilai-nilai m_i ini masih terletak di selang $[0, 4891 - 1]$ agar transformasi menjadi satu-ke-satu.

Dengan menggunakan kunci pribadi miliknya yaitu $d = 679$dan $n = 4891$, direktur tersebut akan menghitung nilai digital signature dari pesan yang akan ia kirim sebagai berikut:

$$p_1 = 666^{679}\ mod\ 4891 = 1104$$
$$p_2 = 983^{679}\ mod\ 4891 = 1635$$
$$p_3 = 797^{679}\ mod\ 4891 = 4209$$
$$p_4 = 532^{679}\ mod\ 4891 = 4655$$
$$p_5 = 826^{679}\ mod\ 4891 = 2251$$
$$p_6 = 580^{679}\ mod\ 4891 = 2664$$
$$p_7 = 658^{679}\ mod\ 4891 = 4381$$
$$p_8 = 432^{679}\ mod\ 4891 = 4720$$
$$p_9 = 746^{679}\ mod\ 4891 = 4581$$
$$p_{10} = 577^{679}\ mod\ 4891 = 3839$$
$$p_{11} = 325^{679}\ mod\ 4891 = 865$$
$$p_{12} = 007^{679}\ mod\ 4891 = 4702$$

Jadi pesan yang dikirim oleh direktur tersebut adalah

$$(666, 1104) - (983, 1635) - (797, 4209) - (532, 4655) -$$
$$(826, 2251) - (580, 2664) - (658, 4381) - (432, 4720) -$$
$$(746, 4581) - (577, 3839) - (325, 865) - (007, 4702)$$

Setelah menerima pesan dari direktur, para karyawan akan mendekripsi pesan dan memverifikasi pesan tersebut. Langkah pertama adalah mengambil kunci public dari direktur, yaitu $(e = 7, n = 4891)$. Dengan kunci public tersebut, karyawan memverifikasi pesan yang dikirim oleh direktur denga cara

$$m'_1 = 1104^7 \ mod \ 4891 = 666$$
$$m'_2 = 1635^7 \ mod \ 4891 = 983$$
$$m'_3 = 4209^7 \ mod \ 4891 = 797$$
$$m'_4 = 4655^7 \ mod \ 4891 = 532$$
$$m'_5 = 2251^7 \ mod \ 4891 = 826$$
$$m'_6 = 2664^7 \ mod \ 4891 = 580$$
$$m'_7 = 4381^7 \ mod \ 4891 = 658$$
$$m'_8 = 4720^7 \ mod \ 4891 = 432$$
$$m'_9 = 4581^7 \ mod \ 4891 = 746$$
$$m'_{10} = 3839^7 \ mod \ 4891 = 577$$
$$m'_{11} = 865^7 \ mod \ 4891 = 325$$
$$m'_{12} = 4702^7 \ mod \ 4891 = 007$$

Karena nilai m_i yang dikirim oleh direktur sama dengan nilai m'_i yang didapat dari hasil perhitungan karyawan, maka dapat disimpulkan bahwa keutuhan pesan tersebut dapat dijamin. Dan dengan mengubah kode tersebut dalam tabel ASCII, didapat pesan asli dari direktur yaitu

$$m = BESOK \ RAPAT \ JAM \ 9$$

3.3 Keamanan RSA

Pengamanan pesan dengan menggunakan algoritma RSA didasarkan pada dua masalah matematika,yaitu:

1. Penanganan masalah faktorisasi pada bilangan bilangan yang sangat besar.

2. Permasalahan perhitungan modulus n yang bersesuaian dengan persamaan $m^e \equiv c \; mod \; n$. Sampai saat ini, untuk memecahkan algoritma yang paling baik adalah dengan memfaktorkan nilai n menjadi nilai p dan q sehingga akan didapat nilai $\varphi(n) = (p-1)(q-1)$ sehingga didapat nilai d yang didapat dari nilai e. Oleh karena itu, sangat disarankan agar nilai dari p dan q memiliki panjang 100 digit sehingga menghasilkan nilai n berukuran 200 digit. Dengan cara tersebut, akan sangat menyulitkan untuk memfaktorkan nilai n. Menurut Rivest, Shamir, dan Adleman,butuh waktu komputasi selama 400 milyar untuk memfaktorkan bilangan berukuran 200 digit. Meskipun belum ada bukti yang menyatakan bahwa memfaktorkan nilai n merupakan satu-satunya cara untuk membongkar algoritma RSA, namun tetap saja belum ada metode yang lebih efisien dari pemfaktoran nilai n.

3.4 Kelebihan dan Kekurangan RSA

Dalam penggunaan kriptografi RSA, tentu terdapat eberapa kekurangan dan kelebihan yang didapat. Sekarang akan dibahas beberapa kekurangan dan kelebihan serta perbandingan kriptografi RSA dengan kriprografi lainnya.

Menurut Yudi Retanto dalam artikelnya yang berjudul "Perbandingan Algoritma RSA dan Diffie-Hellman" terdapat beberapa perbedaan yang terdapat pada kedua algoritma tersebut.Yang pertama adalah perbandingan waktu. Menurut penelitian yang dilakukan oleh Yudi Retanto, didapat rata-rata waktu yang diperlukan algoritma RSA adalah 433,9 ms sedangkan untuk algoritma Diffie-Hellman membutuhkan rata-rata 318,5 ms. Sehingga dapat dikatakan bahwa kecepatan proses algoritma Diffie-Hellman lebih baik dari algoritma RSA. Hal ini terjadi karena algoritma pada kriptografi Diffie-Hellman lebih sederhana dibandingkan dengan algoritma RSA. Berikut tabel waktu proses dari algoritma kriptografi RSA dan Diffie-Hellman.

a. Algoritma RSA

Tabel 3.2 Waktu Proses RSA

NO	$Waktu\ (m/s)$
1	365
2	431
3	451
4	407
5	396
6	579
7	472
8	378
9	432
10	428

b. Algoritma Diffie-Hellman.

Tabel 3.3 Waktu Proses Diffie-Hellman

NO	$Waktu\ (m/s)$
1	353
2	297
3	319
4	314
5	321
6	291

7	352
8	334
9	349
10	255

Lalu dari perbandingan tingkat keamanan pun algoritma Diffie-Hellman lebih unggul dari algoritma RSA. Hal ini karena pada algoritma RSA,nilai$n$ dipublikasikan sehingga ada kemungkinan dibongkar oleh pihak lain sengan cara pemfaktoran nilai n menjadi p dan q sehingga didapat nilai d. Sedangkan pada algoritma Diffie-Hellman, nilai $x_a$dan x_b yang dipilih oleh masing-masing pihak tidak dikirimkan sehingga terjamin keamanannya. Namun dalam efektifitas kunci, algoritma RSAunggul dari algoritma Diffie-Hellman,karena dalam algoritma Diffie-Hellman, membutuhkan pertukaran data antar pihak untuk saling berkomunikasi, sehingga akan butuh data baru dan pertukaran data baru jika ingin melakukan komunikasi dengan pihak lain. Sedangkan pada algoritma RSA, tidak perlu lagi melakukan pembangkitan kunci. Setiap pihak cukup mengambil kunci publik yang telah tersedia tanpa harus melakukan pembangkitan kunci dari awal. Berikut merupakan proses pembangkitan kunci kriptografi Diffie-Hellman.

Algoritma Diffie Hellman merupakan salah satu algoritma kunci simetris karena hanya menggunakan kunci pribadi dalam proses enkripsi dekripsinya. Berikut ini adalah langkah-langkah yang dilakukan:

1. Ada dua pihak yang akan saling berkomunikasi, yaitu A dan B.
2. Untuk berkomunikasi, A dan B perlu mengetahui kunci pribadi masing-masing.
3. Caranya yaitu A dan B memilih sebuah bilangan prima p dan bilangan bulat g dimana $p > g$ dan p relatif prima dengan g.
4. A memilih satu bilangan acak rahasia x_a lalu menghitung

$$y_a = g^{x_a} \, mod \, p$$

5. B juga memilih bilangan acak rahasia x_b lalu menghitung

$$y_b = g^{x_b} \, mod \, p$$

6. Lalu A dan B saling mengirim nilai y_a dan y_b dan menjaga nilai acak masing-masing yaitu $x_a$dan x_b.
7. A mulai menghitung kunci rahasia yang akan digunakan untuk enkripsi dan dekripsi yaitu dengan menggunakan perhitungan $kunci_a = {y_b}^{x_a} \, mod \, p$.
8. B pun mulai menghitung kunci rahasia yang akan digunakan untuk enkripsi dan dekripsi yaitu dengan menggunakan perhitungan $kunci_b = {y_a}^{x_b} \, mod \, p$.
9. Maka nilai $kunci_a = kunci_b$ sehingga A dan B dapat memulai bertukar pesan.

Menurut penelitian Nikolaus Indra dalam artikelnya yang berjudul "Analisis dan Perbandingan Kecepatan Algoritma RSA dan Algoritma Elgamal"didapat bahwa waktu proses algoritma RSA lebih baik dibandingkan dengan algoritma Elgamal. Hal ini dikarenakan proses algoritma pada Elgamal lebih kompleks daripada algoritma RSA. Berikut tabel waktu proses algoritma kriptografi RSA dan Elgamal.

a. Algoritma RSA.

Tabel 3.4 Waktu Proses Enkripsi dan Dekripsi RSA

NO	*Enkripsi*	*Dekripsi*
1	0,136	0,100
2	0,334	0,296
3	0,894	0,802

b. Algoritma Elgamal

Tabel 3.5 Waktu Proses Enkripsi dan Dekripsi Elgamal

NO	$Enkripsi$	$Dekripsi$
1	0,186	0,121
2	0,401	0,309
3	1,002	0,909

Namun karena hal tersebut, tingkat keamanan pesan dengan algoritma Elgamal lebih baik dari algoritma RSA. Karena kompleksnya algoritma Elgamal, membuat algoritma Elgamal lebih sulit dipecahkan daripada algoritma RSA. Untuk perbandingan dalam hal efektifitas, algoritma RSA lebih unggul dari algoritma Elgamal. Dalam algoritma RSA,chiperteks yang dihasilkan hanya memiliksatu nilai,sedangkan pada algoritma Elgamal, chiperteks yang dihasilkan ada dua untuk setiap bloknya. Berikut sekilas tentang algoritma dari kriptografi Elgamal.

Keamanan kriptografi Elgamal terletak pada tingkat kesulitan dalam menghitung logaritma diskrit. Logaritma diskrit dalam kriptografi Elgamal adalah sebagai berikut: terdapat bilangan prima p dan terdapat bilangan bulat αdan β. Maka harus dicari nilai a sedemikian sehingga

$$\beta = \alpha^a \ mod \ p$$

Bilangan a disebut logaritma diskrit terhadap β dengan basis α ($a = \log \alpha^{\beta}$).

Parameter yang dibutuhkan algoritma Elgamal adalah:

1. p (tidak rahasia)
2. α (tidak rahasia)
3. $a, 1 \leq a \leq p-2$ (rahasia)
4. $\beta = \alpha^a \ mod \ p$ (tidak rahasia)

Langkah dalam membangkitkan kunci public dan kunci pribadi adalah :

1. Pilih sembarang bilangan prima p dan elemen primitif α.
2. Pilih bilangan $a$dengan $1 \leq a \leq p-2$.

3. Hitung $\beta = \alpha^a \ mod \ p$.
4. Publikasikan nilai p, α, β dan rahasiakan nilai a.

dari langkah tersebut, maka akan didapat kunci public dan kunci pribadi yaitu

$$(p, \alpha, \beta)$$

dan

$$(a)$$

Setelah mendapatkan kunci public dan kunci pribadi, selanjutnya akan dilakukan proses enkripsi terhadap plainteks. Proses enkripsi algoritma Elgamal adalag sebagai berikut:

1. Plainteks dipecah menjadi blok yang kecil $m_1, m_2, \dots, m_n$.
2. Pilih bilangan acak k yang terletak pada nilai $1 \leq k \leq p-2$.
3. Setiap blok dienkripsi dengan rumus

   $$\gamma = \alpha^k \ mod \ p$$

   dan

   $$\delta = \beta^k m \ mod \ p$$

   diperoleh chiperteks (γ, δ). Jadi ukuran chiperteks dua kali dari ukuran plainteksnya.Bilangan k ditentukan oleh pengirim dan harus dijaga kerahasiaannya jadi hanya pengirim saja yang mengetahuinya, tetapi nilai k hanya digunakan saat enkripsi, sehingg tidak perlu disimpan.

Selanjutnya akan dijelaskan proses dekripsi dari algoritma Elgamal, yaitu:

1. Gunakan kunci privat a untuk menghitung ${\gamma_i}^{p-1-a} \ mod \ p$.
2. Hitung $m_i = \delta {\gamma_i}^{p-1-a} \ mod \ p$.
3. Diperoleh plainteks $m_1, m_2, \dots, m_n$

Contoh :

Misalkan A dan B akan berkomunikasi dengan menggunakan kriptografi Elgamal. B membangkitkan kunci public dan kunci pribadi dengan memilih bilangan prima $p = 2579$ dan elemen primitive $\alpha = 2$. Selanjutnya dipilih $a = 765$ dan dihitunh

$$\beta = 2^{765} \ mod \ 2579 = 949.$$

diperolehkunci public $(p, \alpha, \beta) = (2579, 2, 949)$ dan kunci pribadi $a = 765$. Lalu B memberikan kunci $(2579, 2, 949)$ kepada A.

Selanjutnya A akan mengirim pesan kepada B yaitu "Temui aku". Pesan tersebut akan dienkripsi oleh A dengan menggunakan kunci public dari B. pesan tersebut jika diubah ke dalam kode ASCII akan menjadi

84 101 109 117 105 32 97 107 117

Pembagian blok sesuai kode dalam tiap kata. Sehingga jika A melakukan proses enkripsi akan menghasilkan chiperteks yaitu

Tabel 3.6 Proses Enkripsi Menggunakan Kriptografi Elgamal

i	m_i	k_i	$\gamma_i = 2^{k_i} \, mod \, 2579$	$\delta_i = 949^{k_i}.m_i \, mod \, 2579$
1	84	1414	716	814
2	101	1527	711	344
3	109	1843	1512	1252
4	117	2175	559	2337
5	105	1553	929	1032
6	32	2210	838	1014
7	97	1404	313	1998
8	107	2183	1259	257
9	117	1091	195	1173

Berdasarkan tabel tersebut, diperoleh chiperteks sebagai berikut

$(716, 814)$ $(711, 344)$ $(1512, 1252)$

$(559, 2337)$ $(929, 1032)$ $(838, 1014)$

$(313, 1998)$ $(1259, 257)$ $(195, 1173)$

Chiperteks ini dikirimkan oleh A kepada B.

Selanjutnya B akan melakukan proses dekripsi terhadap chiperteks yang dikirimkan A, dengan cara sebagai berikut.

B memiliki kunci $p = 2579$ dan kunci pribadi $a = 765$. Selanjutnya B melakukan perhitungan sebagai berikut.

Tabel 3.6 Proses Dekripsi Menggunakan Kriptografi Elgamal

i	γ_i	δ_i	${\gamma_i}^{1813}\ mod\ 2579$	$m_i = \delta_i {\gamma_i}^{1813}\ mod\ 2579$	Karakter m_i
1	716	814	1090	84	T
2	711	344	2047	101	e
3	1512	1252	2301	109	m
4	559	2237	202	117	u
5	929	1032	1552	105	i
6	838	1014	936	32	$< spasi >$
7	313	1998	133	97	a
8	1259	257	1887	107	k
9	195	1173	1128	117	u

Dari perhitungan tersebut, B mengetahui pesan yang dikirim A yaitu ‘Temui aku”