BAB II
TINJAUAN PUSTAKA

2.1 Kriptografi

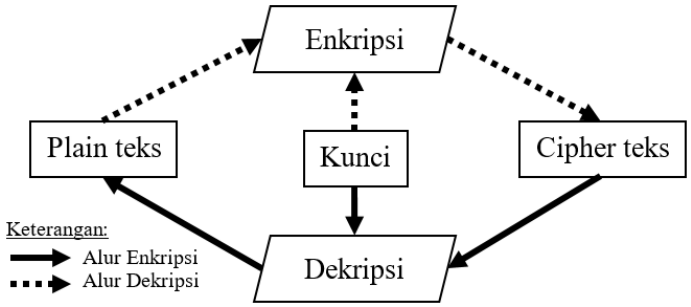
Metode matematis yang terkait dengan masalah keamanan sistem informasi seperti kerahasiaan, integritas data, dan otentikasi dikenal sebagai kriptografi (Diana & Zebua, 2018). Kata kriptografi (*cryptography*) berasal dari bahasa Yunani dan memiliki dua suku kata yaitu kripto dan graphia. Kripto adalah menyembunyikan, sedangkan graphia adalah tulisan. Jadi kriptografi dapat diartikan sebagai ilmu yang berkonsentrasi pada penyamaran huruf atau penulisan agar tulisan tidak dapat dibedakan oleh orang yang tidak berwenang. (Ginting & Ginting, 2017).

Dengan mengubah pesan rahasia (*plaintext*) menjadi sandi (ciphertext), maka keamanan data berdasarkan algoritma teknik kriptografi telah tercapai. Proses untuk mengubah *plaintext* menjadi *ciphertext* disebut enkripsi, sedangkan proses untuk mengembalikan *ciphertext* menjadi *plaintext* disebut dekripsi. Kunci merupakan sebuah kode yang digunakan untuk mengimplementasikan proses enkripsi dan dekripsi. Kunci harus bersifat rahasia dan tidak dibagikan kepada orang lain yang bukan merupakan penerima pesan.

Dalam kriptografi, setiap orang dimungkinkan untuk bebas memilih metode untuk merahasiakan pesan. Metode tersebut berbeda-beda untuk setiap pelaku kriptografi sehingga penulisan pesan rahasia mempunyai estetika tersendiri. Estetika penulisan pesan rahasia ini menjadikan kriptografi sebagai sebuah seni. Pada perkembangan selanjutnya, kriptografi dikenal sebagai disiplin ilmu yang menggunakan teknik matematika untuk keamanan informasi, seperti privasi dan autentikasi.

Hal yang harus dicapai dalam penerapan algoritma kriptografi adalah *confusion* (konfusi/pembingungan) yaitu harus mampu mempersulit pihak lain dalam merekonstruksi ulang *cipher* yang dihasilkan serta *diffusion* (peleburan) yaitu harus mampu menyembunyikan pola dari pesan asli (Diana & Zebua, 2018).

Kriptografi merupakan ilmu yang digunakan untuk mengamankan data dengan cara menyandian data dengan algoritma tertentu. Secara umum terdapat dua proses utama dalam kriptografi, yaitu enkripsi dan dekripsi. Kedua proses tersebut tentunya membutuhkan minimal sebuah kunci. Gambaran umum kedua proses enkripsi dan dekripsi (Setiadi et al., 2018b) dapat dilihat pada gambar 2.1.



Gambar 2.1 Gambaran Umum Proses Enkripsi dan Dekripsi (Setiadi et al., 2018b)

Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang - orang yang tidak berhak atas pesan tersebut. Algoritma kriptografi terdiri dari tiga fungsi dasar (Ginting & Ginting, 2017), yaitu sebagai berikut:

1. Enkripsi

   Enkripsi merupakan hal yang sangat penting dalam kriptografi. Enkripsi adalah pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti. Sama halnya dengan kita tidak mengerti akan sebuah kata maka kita akan melihatnya di dalam kamus atau daftar istilah. Beda halnya dengan enkripsi. untuk mengubah teks asli ke bentuk teks kode kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.

2. Dekripsi

   Dekripsi merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.

3. Kunci

   Kunci yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci privat (*private key*) dan kunci umum (*public key*)

2.1.1 Super Enkripsi

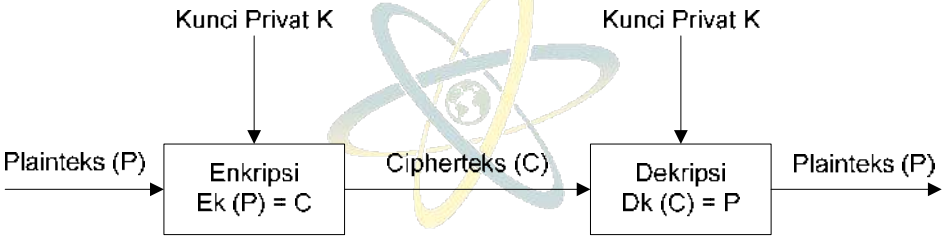
Super enkripsi adalah salah satu teknik kriptografi yang menggabungkan dua atau lebih teknik subtitusi dan permutasi, Teknik Super Enkripsi sangatlah baik digunakan dalam keamanan berganda tepatnya pada menanggulangi kebocoran data karena dengan menggunakan super enkripsi dapat menambah keamanan yang lebih (Megantara & Rafrastara, 2019), pada penelitan ini akan mengimplementasikan kombinasi algoritma *Beaufort Cipher* dan algoritma RSA (*Rivest Shamir Adleman*) untuk pengamanan *file* teks. Tujuan menggabungkan kedua algoritma tersebut adalah untuk menciptakan *ciphertext* yang lebih kuat dan lebih sulit dipecahkan. Ini terjadi karena ada dua tahap dalam proses enkripsi dan dekripsi, dan setiap tahap menggunakan dua jenis kunci yang berbeda. Kombinasi antara kedua algoritma juga akan mengatasi penggunaan *ciphertext* tunggal yang secara komparatif lemah. Metode gabungan antara dua algoritma enkripsi ini disebut dengan super enkripsi.

2.2 Sistem Kriptografi (*Cryptosystem*)

Sistem kriptografi (*cryptosystem*) sering disebut juga dengan sistem *cipher* (*cipher system*) adalah sistem yang terdiri dari algoritma enkripsi, algoritma dekripsi dan tiga komponen teks (*plaintext*, *ciphertext* dan kunci) (Muchlis et al., 2017). Secara umum ada dua jenis sistem kriptografi berbasis kunci, yaitu sistem kriptografi simetris (*Symmetric Cryptosystem*) dan sistem kriptografi asimetris (*Asymmetric Cryptosystem*).

2.2.1 *Symmetric Cryptosystem*

Sistem kriptografi simetris (*Symmetric Cryptosystem*), sering disebut algoritma konvensional, adalah algoritma dimana kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya, artinya kunci enkripsi sama dengan kunci dekripsi. Keamanan algoritma simetris terletak pada kunci, kebocoran kunci berarti siapapun bisa mengenkripsi dan mendekripsi pesan (Muchlis et al., 2017). Contoh sistem kriptografi simetris ini adalah algoritma *Beaufort Cipher*. Adapun skema dari *symmetric cryptosystem* dapat dijelaskan pada gambar 2.2.



Gambar 2.2 Skema *Symmetric Cryptosystem* (Muchlis et al., 2017)

Berdasarkan skema pada gambar 2.2, kriptografi simetris menggunakan kunci tunggal yang mengharuskan pengirim dan penerima menyetujui dan mengetahui satu kunci rahasia tertentu sebelum mereka dapat berkomunikasi dengan aman. Keamanan kriptografi simetris ini terletak pada kerahasiaan kuncinya. Semua algoritma kriptografi klasik termasuk dalam algoritma simetris.

2.2.2 *Asymmetric Cryptosystem*

Sistem kriptografi kunci-publik (*Public-key Cryptosystem*), sering disebut algoritma asimetris (*Asymmetric Cryptosystem*), adalah algoritma dimana kunci yang digunakan untuk enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Selain itu, kunci dekripsi tidak dapat (setidaknya dalam jumlah waktu yang wajar) dihitung dari suatu kunci enkripsi. Algoritma ini disebut kunci publik karena kunci enkripsi dapat dibuat publik yaitu pihak luar dapat menggunakan kunci enkripsi untuk mengenkripsi pesan, tetapi hanya orang tertentu dengan kunci dekripsi yang sesuai dapat mendekripsi pesan (Muchlis et al., 2017).

Ada dua masalah matematika yang sering dijadikan dasar pembangkitan sepasang kunci pada algortima kunci-publik (Muchlis et al., 2017), yaitu:

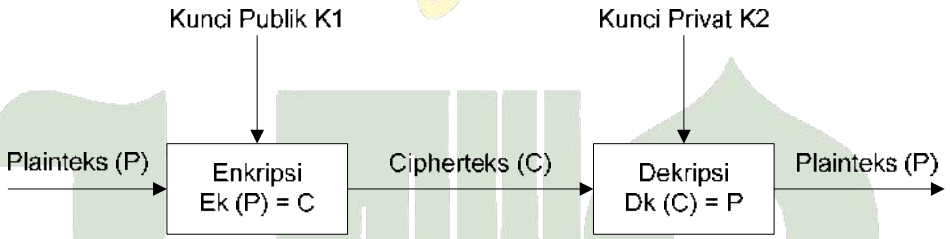
1. Pemfaktoran

   Diberikan bilangan bulat *n*. faktorkan *n* menjadi faktor primanya. Semakin besar *n*, semakin sulit memfaktorkannya (butuh waktu sangat lama). Salah satu algoritma yang menggunakan prinsip ini adalah RSA.

2. Logaritma Diskrit

   Temukan *x* sedemikian sehingga $a^x \equiv b(mod\ n)$ sulit dihitung. Semakin besar *a*, *b* dan *n* semakin sulit memfaktorkannya. Algoritma yang menggunakan prinsip ini adalah *Elgamal*.

Adapun skema dari *asymmetric cryptosystem* dapat dijelaskan pada gambar 2.3.



Gambar 2.3 Skema *Asymmetric Cryptosystem* (Muchlis et al., 2017)

Sistem kriptografi kunci asimetris juga disebut kriptografi *modern*, memiliki dua jenis kunci, yaitu kunci enkripsi dan kunci dekripsi yang berbeda. Kunci enkripsi dimiliki oleh pengirim dan kunci dekripsi dimiliki oleh penerima pesan. Dalam kriptografi kunci asimetris, hampir semua algoritma kriptografinya menggunakan konsep kunci publik, seperti algoritma RSA.

2.3 Algoritma *Beaufort Cipher*

Beaufort Cipher merupakan salah satu bentuk lain dari *Vigenere Cipher* yang menggunakan teknik subtitusi dan merupakan algoritma kriptografi dengan kunci simetris (Setiadi et al., 2018a). Seperti namanya *Beaufort Cipher* ditemukan oleh Sir Francis Beaufort (Irawan et al., 2020). Enkripsi menggunakan algoritma

Beaufort Cipher merupakan teknik substitusi kriptografi yang menggunakan operasi *modulo* bilangan bulat sebagai proses utama (Sugiarto et al., 2020).

Rumus yang digunakan pada *Beaufort Cipher* sangat identik dengan *Vigenere Cipher*. Kesamaan dari kedua teknik ini adalah penggunaan fungsi *modulo* atau sisa hasil bagi maupun jenis kunci yang yang digunakan. Perbedaan dari kedua metode ini adalah peranan kunci, dalam *Vigenere Cipher* kunci digunakan sebagai penambah *plaintext* dan pengurang *ciphertext*. Sedangkan dalam formula yang digunakan *Beaufort Cipher*, kunci digunakan untuk dikurangkan dengan *plaintext* maupun *ciphertext* (Setiadi et al., 2018a).

Kunci *(K)* pada Beaufort Cipher adalah urutan karakter-karakter *K = k1 ... kd* dimana *k1* didapat dari banyaknya pergeseran dari alfabet ke-*i* sama seperti *Viginere Cipher*. Artinya bahwa jumlah kunci yang dibangkitkan harus sama dengan jumlah karakter *plaintext* yang diamankan. Algoritma ini melakukan proses enkripsi dan dekripsi secara stream (masing-masing karakter *plaintext* harus memiliki pasangan kunci). Hal ini yang menyebabkan algoritma ini hampir sama dengan algoritma *Viginere Cipher* (Diana & Zebua, 2018).

Adapun formulasi algoritma *Beaufort Cipher* dalam proses enkripsi (Naing & Aye, 2020) dengan menggunakan persamaan (2.1).

$$C_i = E_k(M_i) = (K_i - M_i) \bmod 26 \quad (2.1)$$

Sedangkan formulasi algoritma *Beaufort Cipher* dalam proses dekripsi (Naing & Aye, 2020) dengan menggunakan persamaan (2.2).

$$M_i = D_k(C_i) = (K_i - C_i) \bmod 26 \quad (2.2)$$

Keterangan:

M_i = Pesan yang akan dienkripsi (*plaintext*)

C_i = Sandi atau pesan hasil enkripsi (*ciphertext*)

K_i = Kunci

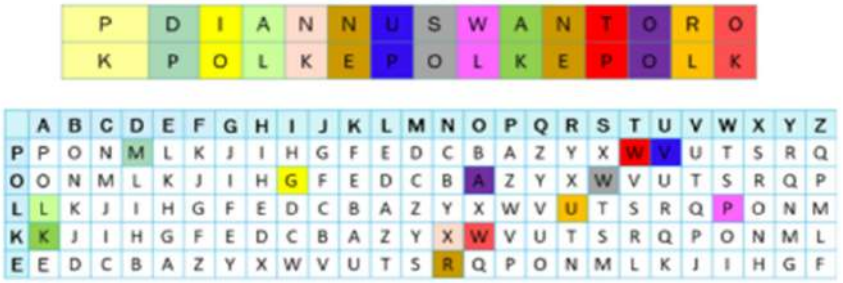
E_k = Fungsi Enkripsi

D_k = Fungsi Dekripsi

Nilai *mod 26* pada persamaan (2.1) dan (2.2) tergantung dari jumlah kebutuhan karakter yang digunakan, pada awalnya *Beaufort Cipher* hanya

menggunakan 26 karakter, namun seiring dengan perkembangan teknologi komputer saat ini, maka dapat menggunakan *mod 256* (mengguakan seluruh tabel ASCII) (Diana & Zebua, 2018).

Enkripsi dengan *Beaufort Cipher* dapat diselesaikan dengan tabel *tabula r*ecta. Untuk menyelesaikan sebuah persoalan tentang mengenkripsi dengan *Beaufort Cipher*, kita harus membuat sebuah kolom dimana *header* atas diisi dengan huruf abjad, kemudian header samping diisi dengan kunci (*key*) dengan penghilangan huruf yang kembar. Kunci (*key*) dijabarkan sepanjang huruf *plaintext* (Irawan et al., 2020) seperti pada gambar 2.4.



Gambar 2.4 Enkripsi *Beaufort Cipher* Teknik *Tabula Recta* (Irawan et al., 2020)

Berdasarkan gambar 2.4, menjelaskan proses enkripsi pada algoritma *Beaufort Cipher* dengan menggunakan tabel *tabula recta*. *Plaintext* dan kunci yang digunakan pada gambar 2.1 yaitu:

Plaintext (M_i) = DIANNUSWANTORO

Kunci (K_i) = POLKE

Untuk mendapatkan *ciphertext* dari *plaintext* dan kunci pada gambar 2.4 diatas, untuk huruf *plaintext* pertama **[D]**, ditarik garis vertikal dari huruf **[D]** dan ditarik garis horizontal dari huruf [**P]**, perpotongannya adalah pada kotak yang berisi huruf [**M]**. Dengan cara yang sama, maka diperoleh hasil enkripsi seluruhnya seperti disajikan pada tabel 2.1.

Tabel 2.1 Proses Enkripsi *Beaufort Cipher* Teknik *Tabula Recta*

Plaintext (M_i)	D	I	A	N	N	U	S	W	A	N	T	O	R	O
Kunci (K_i)	P	O	L	K	E	P	O	L	K	E	P	O	L	K
Ciphertext (C_i)	M	G	L	X	R	V	W	P	K	R	W	A	U	W

Berdasarkan tabel 2.1, hasil enkripsi dengan teknik tabel *tabula recta* (bujur sangkar) diperoleh *ciphertext*, yaitu "MG LXRVWPKRWAUW".

Sedangkan proses enkripsi algoritma *Beaufort Cipher* dengan menggunakan persamaan (2.1) dapat dilakukan dengan menukarkan huruf dengan angka. Karakter huruf yang digunakan pada algoritma *Beaufort Cipher* yaitu *A, B, C, ..., Z* dan disamakan dengan angka *0, 1, 2, ..., 25* seperti disajikan pada tabel 2.2 berikut.

Tabel 2.2 Tabel Substitusi Angka Algoritma *Beaufort Cipher*

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	0	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Dengan menggunakan *plaintext* yang sama pada proses enkripsi dengan teknik *tabula recta* dapat diketahui panjang *plaintext* = 14, sedangkan panjang kunci = 5, karena panjang kunci < panjang *plaintext*, maka kunci tersebut akan diulang secara periodik hingga panjang kunci tersebut sama dengan panjang *plaintext*-nya, yaitu :

Plaintext (M_i) = DIANNUSWANTORO

Kunci (K_i) = POLKE POLKE POLK

Pada contoh diatas kata kunci POLKE diulang sedemikian rupa hingga panjang kunci sama dengan panjang *plaintext*-nya. Kemudian setelah panjang kunci sama dengan panjang *plaintext*, proses enkripsi dilakukan dengan mensubstitusi *plaintext* dengan kunci seperti terlihat pada tabel 2.3.

Tabel 2.3 Proses Substitusi Angka *Beaufort Cipher*

Plaintext (M_i)	D	I	A	N	N	U	S	W	A	N	T	O	R	O
Subtitusi	3	8	0	13	13	20	18	22	0	13	19	14	17	14
Kunci (K_i)	P	O	L	K	E	P	O	L	K	E	P	O	L	K
Subtitusi	15	14	11	10	4	15	14	11	10	4	15	14	11	10

Enkripsi (E_k)	(subtitusi kunci - subtitusi *plaintext*) mod 26													
Hasil	12	6	11	-3	-9	-5	-4	-11	10	-9	-4	0	-6	-4
Ciphertext (C_i)	M	G	L	X	R	V	W	P	K	R	W	A	U	W

Berdasarkan tabel 2.3 dapat dijelaskan proses enkripsi dengan teknik substitusi pada algoritma *Beaufort Cipher* sebagai berikut:

Plaintext huruf pertama (M_i = 0): D

Kunci huruf pertama (K_i = 0) : P

selanjutnya lakukan proses enkripsi dengan menggunakan persamaan (2.1) sehingga diperoeleh hasilnya sebgai berikut:

$$C_i = (K_i - M_i)\ mod\ 26$$

$$C_i = (P - D)\ mod\ 26$$

$$C_i = (15 - 3)\ mod\ 26$$

$$C_i = (12)\ mod\ 26 = 12$$

Angka 12 jika diterjemahkan kembali menjadi huruf sesuai urutan awal pada tabel substitusi angka algoritma *Beaufort Cipher*, maka diperoleh *ciphertext* menjadi huruf "M". Ulangi langkah yang sama sampai semua *plaintext* berhasil dienkripsi sehingga diperoleh *ciphertext* yang sama dengan teknik menggunakan *tabula recta*, yaitu "MG LXRVWPKRWAUW".

2.4 Algoritma RSA

Algoritma RSA merupakan algoritma kriptografi asimetris dimana kunci enkripsi tidak sama dengan kunci dekripsinya. Algortima RSA dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976. Nama RSA merupakan singkatan dari nama tiga orang penemunya, yaitu Ron Rivest, Adi Shamir, dan Len Adleman (Ginting & Ginting, 2017).

Algoritma RSA terdiri dari kunci publik dan kunci privat dimana kunci publik dapat diketahui oleh semua orang sedangkan kunci privat hanya diketahui oleh pemilik data. Proses enkripsi menggunakan kunci publik dan proses dekripsi menggunakan kunci privat pemilik data (Suhandinata et al., 2019). Algoritma RSA ini mengambil dua bilangan prima secara acak yang akan dijadikan kunci

sehingga didapat dua kunci yaitu kunci publik (*public key*) dan kunci privat (*private key*) (Muchlis et al., 2017).

Dalam sistem algoritma kriptografi RSA terdapat proses utama, yaitu proses pembuatan kunci Privat dan kunci Publik dimana masing masing digunakan dalam proses enkripsi dan dekripsi. Dalam algoritma RSA terdapat besaran-besaran yang penting (Sutejo, 2021) yaitu sebagai berikut:

1. Nilai p dan q merupakan bilangan prima diambil secara acak atau lebih baiknya langsung dipilih oleh orang yang akan menerima data. Sifat dari kedua bilangan ini adalah rahasia, dimana hanya pengirim data dan penerima data yang dapat mengetahuinya.
2. Nilai n merupakan bersifat publik, karena sifat n adalah tidak rahasia
3. $\varphi(n) = (p-1)(q-1)$, sifat dari bilangan ini adalah rahasia.
4. e (kunci enkripsi), sifat bilangan kunci enkripsi tidak rahasia.
5. d (kunci dekripsi), kunci dekripsi bersifat rahasia.
6. m (*plaintext*), merupakan informasi awal yang bersifat rahasia.
7. c (*chipher*teks), merupakan infomasi yang telah di enkripsi yang bersifat tidak rahasia.

2.4.1 Pembangkit Kunci Algoritma RSA

Untuk mengenkripsi dan dekripsi pesan dengan menggunakan algoritma RSA terlebih dahulu membangkitkan sepasang kunci, yaitu kunci publik (*public key*) dan kunci privat (*private key*). Hal pertama yang dilakukan algoritma pembangkit kunci adalah membangkitkan 2 bilangan prima besar. Berikut ini algoritma penyelesaiannya (Muchlis et al., 2017), yaitu sebagai berikut:

1. Pilih dua bilangan prima sembarang, p dan q
2. Hitung $n = p * q$

   Sebaiknya $p \neq q$, sebab jika $p = q$ maka $n = p^2$ sehingga p dapat diperoleh dengan menarik akar pangkat dua dari n.
3. Hitung $\varphi(n) = (p-1)(q-1)$
4. Pilih kunci publik e, yang relatif prima terhadap $\varphi(n)$ yaitu $1 < e < \varphi(n)$ dan $\gcd(e, \varphi(\mathrm{n})) = 1$.

5. Bangkitkan kunci privat dengan menggunakan persamaan (2.3) berikut:

   $d = 1 \left(mod\ \varphi(n)\right)$ (2.3)

   syarat $(0 \leq d \leq n)$

   Sehingga pasangan kunci enkripsi dan dekripsi hasil dari algoritma algoritma RSA di atas adalah sebagai berikut:

   a. Kunci enkripsi (*public key*) adalah pasangan (e, n)
   b. Kunci dekripsi (*private key*) adalah pasangan (d, n)

Contoh:

Misalkan A akan membangkitkan kunci publik dan kunci privat miliknya. A memilih $p = 47$ dan $q = 71$ (keduanya bilangan prima). Selanjutnya A menghitung:

$n = p * q$

$n = 47 * 71 = 3337$

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

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

$\varphi(n) = 46 * 70 = 3220$

A memilih kunci publik $e = 79$ karena relatif prima dengan 3220. e dengan *greatest common divisor* atau $\gcd(e, \varphi(\mathrm{n})) = 1$. Karena e mempunyai ketentuan $e > 1\ \ dan\ e < m$, maka e dimulai dari $e = 2$

Pembuktiannya:

Untuk nilai $e = 2$

Jadi, $\gcd(2,\ 3220) = 2$

Karena $\gcd(2,\ 3220) = 2$, maka tidak memenuhi $\gcd(e,\ \varphi(\mathrm{n})) = 1$

Untuk nilai $e = 5$

Jadi, $\gcd(5,\ 3220) = 5$

Karena $\gcd(5,\ 3220) = 5$, maka tidak memenuhi $\gcd(e,\ \varphi(\mathrm{n})) = 1$

……..

Untuk nilai $e = 79$

Jadi, $\gcd(79,\ 3220) = 1$

Karena $\gcd(79,\ 3220) = 1$, maka memenuhi $\gcd(e,\ \varphi(\mathrm{n})) = 1$

A mengumumkan nilai e dan n. Selanjutnya A menghitung kunci private (dekripsi) d, sehingga dituliskan berdasarkan persamaan (2.3) yaitu:

$d = 1\ (mod\ \varphi(n))$

$79 * d = 1(mod\ 3220)$

Dengan mencoba nilai-nilai $d = 1, 2, 3, \ldots$, diperoleh nilai d memenuhi persamaan (2.3) yaitu 1019.

Pembuktian:

untuk $d = 1,$ maka $79 * 1 = 1(mod\ 3220) = 79$

untuk $d = 2,$ maka $79 * 2 = 1(mod\ 3220) = 158$

untuk $d = 3,$ maka $79 * 3 = 1(mod\ 3220) = 237$

……

untuk $d = 1019,$ maka $79 * 1019 = 1(mod\ 3220) = 1$

Kunci private digunakan untuk mendekripsi pesan dan harus dirahasiakan A. Jadi, perhitungan kunci ini menghasilkan pasangan kunci sebagai berikut:

1. Kunci enkripsi (*public key*) adalah pasangan $(e, \mathrm{n}) = (79,\ 3337)$
2. Kunci dekripsi (*private key*) adalah pasangan $(d, \mathrm{n}) = (1019,\ 3337)$

Pada algoritma RSA hanya diberikan kunci publik yaitu e dan n. Sedangkan kunci privat d dirahasiakan. Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan (2.3). Kemudian dilakukan dekripsi *ciphertext* ci menjadi *plaintext* mi dengan menggunakan persamaan (2.5).

2.4.2 Proses Enkripsi Algoritma RSA

Pengamanan data berdasarkan algoritma teknik kriptografi dilakukan dengan merubah pesan yang akan dirahasiakan (*plaintext*) menjadi sandi (*ciphertext*). Proses untuk mengkonversi *plaintext* menjadi *ciphertext* disebut dengan proses enkripsi. Adapun proses enkripsi pesan (*plaintext*) dengan menggunakan algoritma RSA dapat dijelaskan tahapannya sebagai berikut:

1. Ambil kunci publik (e, n) penerima pesan.
2. Nyatakan *plaintext* m menjadi blok-blok $m_1, m_2, m_3, \ldots, m_n$
3. Setiap blok m_i akan dienkripsi menjadi blok c_i dengan rumus persamaan (2.4) berikut:

   $$c_i = m_i^e mod\ n \qquad (2.4)$$

Contoh: Misalkan B mengirim pesan kepada A. Pesan (*plaintext*) yang akan dikirim ke A adalah sebagai berikut:

$m = \mathrm{BUDI}$

B mengubah m ke dalam desimal pengkodean ASCII dan sistem akan memecah menjadi blok-blok yang lebih kecil yaitu sebagai berikut.

$m_1 = 66$

$m_2 = 85$

$m_3 = 68$

$m_4 = 73$

B mengetahui kunci publik A adalah $e = 79$ dan $n = 3337$. B dapat mengenkripsi setiap blok *plaintext* dengan rumus persamaan (2.4) sebagai berikut:

$c_1 = 66^{79}\ mod\ 3337 = 795$

$c_2 = 85^{79}\ mod\ 3337 = 3048$

$c_3 = 68^{79}\ mod\ 3337 = 2753$

$c_4 = 73^{79}\ mod\ 3337 = 725$

Dalam penerapannya, untuk memudahkan sistem membagi *ciphertext* menjadi blok-blok yang mewakili tiap karakter maka ditambahkan digit semu (biasanya 0) pada blok *cipher* sehingga tiap blok memiliki panjang yang sama

sesuai ketetapan (dalam hal ini panjangnya 4 digit). Jadi, *ciphertext* yang dihasilkan adalah $c = 0795\ 3048\ 2753\ 0725$

2.4.3 Proses Dekripsi Algoritma RSA

Proses untuk mengkonversi *ciphertext* menjadi *plaintext* disebut dengan proses dekripsi. Adapun proses dekripsi *ciphertext* dengan menggunakan algoritma RSA dapat dijelaskan tahapannya sebagai berikut:

1. Ambil kunci privat (d, n) penerima pesan.
2. Nyatakan *ciphertext* c menjadi blok-blok $c_1, c_2, c_3, \ldots . c_n$
3. Setiap bok c_i akan didekripsi menjadi blok m_i dengan rumus persamaan (2.5) berikut:

   $$m_i = c_i^d mod\ n \qquad (2.5)$$

Contoh: Dengan kunci privat $d = 1019$ dan $n = 3337$, *chipertext* yang telah dibagi menjadi blok-blok *cipher*, $c = 0795\ 3048\ 2753\ 0725$ kembali diubah ke dalam *plaintext* dengan rumus persamaan (2.5):

$$m_1 = 795^{1019}\ mod\ 3337 = 66$$
$$m_2 = 3048^{1019}\ mod\ 3337 = 85$$
$$m_3 = 2753^{1019}\ mod\ 3337 = 68$$
$$m_4 = 725^{1019}\ mod\ 3337 = 73$$

Sehingga *plaintext* yang dihasilkan $m =$ BUDI

2.5 *File* Teks

Menuurut KBBI (Kamus Besar Bahasa Indonesia) teks adalah naskah yang berupa kata-kata asli dari pengarang. Sedangkan menurut Alex Sobur teks adalah seperangkat tanda yang ditransmisikan dari seorang pengirim kepada seorang penerima melalui medium tertentu atau kode-kode tertentu. *File* teks merupakan berkas yang mengandung informasi-informasi dalam bentuk teks yang terdiri dari karakter, angka dan tanda baca. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data

merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca.

Masukan dan keluaran data teks direpresentasi sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga macam set karakter yang umum digunakan untuk masukan dan keluaran pada komputer, yaitu ASCII, Unicode, dan EBCDIC. ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex, dan Unicode tetapi ASCII bersifat lebih universal. ASCII digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII memiliki komposisi bilangan *biner* sebanyak 8 *bit*, dimulai dari 00000000 dan 11111111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan desimal. Unicode adalah suatu standar industri yang dirancang untuk mengizinkan teks dan simbol dari semua sistem tulisan di dunia untuk ditampilkan dan dimanipulasi secara konsisten oleh komputer, EBCDIC (*Extended Binary Code Decimal Interchange Code*) merupakan set karakter yang diciptakan oleh komputer merk IBM (*International Business Machines*). EBCDIC terdiri dari 256 karakter yang masing-masing berukuran 8 *bit* (Subada, 2018).

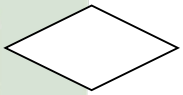
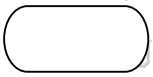
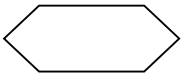
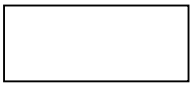
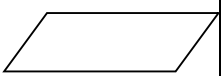
Format data teks (**.txt*) merupakan format teks yang digunakan untuk menyimpan huruf, angka, karakter kontrol (tabulasi, pindah baris, dan sebagainya) atau simbol-simbol lain yang biasa digunakan dalam tulisan seperti titik, koma, tanda petik, dan sebagainya. Satu huruf angka, karakter kontrol atau simbol pada arsip teks memakan tempat satu *byte*. Berbeda dengan jenis teks terformat yang satu huruf saja dapat memakan tempat beberapa *byte* menyimpan format dari huruf tersebut seperti *font*, ukuran, tebal atau tidak dan sebagainya. Kelebihan dari format data teks ini adalah ukuran datanya yang kecil karena tidak ada fitur untuk memformat tampilan teks. Saat ini perangkat lunak yang paling banyak digunakan untuk memanipulasi format data data teks (**.txt*) adalah *Notepad*.

2.6 *Flowchart*

Flowchart merupakan penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program, biasanya mempengaruhi penyelesaian masalah yang khusunya perlu dipelajari dan dievaluasi lebih lanjut (Budiman et al., 2021). *Flowchart* merupakan representasi secara simbolik dari suatu algoritma atau prosedur untuk menyelesaikan suatu masalah. Tujuan utama penggunaan *flowchart* adalah untuk menyederhanakan rangkaian proses atau prosedur untuk memudahkan pemahaman pengguna terhadap informasi tersebut.

Berikut ini adalah simbol-simbol yang digunakan dalam *flowchart*, seperti yang terlihat pada table 2.4.

Tabel 2.4 Simbol-simbol *Flowchart* (Budiman et al., 2021)

No.	Simbol	Keterangan
1.		Permulaan sub program
2.		Perbandingan, pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
3.		Penghubung bagian-bagian *flowchart* yang berada pada satu halaman.)
4.		Penghubung bagian-bagian *flowchart* yang berada pada halaman berbeda
5.		Permulaan/akhir program
6.		Arah aliran program
7.		Proses inisialisasi/pemberian harga awal
8.		Proses penghitung/ proses pengolahan data
9.		Proses *input/output*