

BAB II

LANDASAN TEORI

2.1 Citra Digital

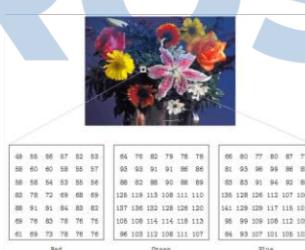
citra adalah suatu representasi (gambaran), kemiripan atau imitasi dari suatu objek. Citra yang berupa *output* dari suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada layar televisi atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan (Hestingsih, 2010).

Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dua dimensi. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam.

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat:

- a) Optik berupa foto.
- b) Analog berupa sinyal video seperti gambar pada monitor televisi.
- c) Digital yang dapat langsung disimpan pada suatu pita magnetik.

Citra yang dimaksudkan di dalam keseluruhan isi buku ini adalah “citra diam” (*still images*). Citra diam adalah citra tunggal yang tidak bergerak. Gambar 2.1 adalah dua buah citra diam. Untuk selanjutnya, citra diam disebut citra saja.



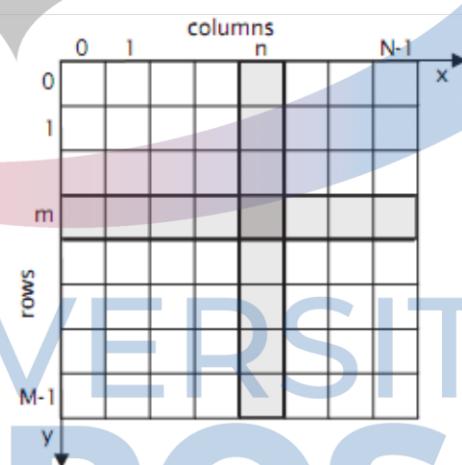
Gambar 2.1 Citra Diam

(R. Kusumanto et. al., 2011)

Secara matematis, citra merupakan fungsi kontinyu (*continue*) dengan intensitas cahaya pada bidang dua dimensi. Agar dapat diolah dengan komputer

digital, maka suatu citra harus dipresentasikan secara numerik dengan nilai-nilai diskrit. Representasi dari fungsi kontinyu menjadi nilai-nilai diskrit disebut digitalisasi citra (R. Kusumanto et. al., 2011). Meskipun sebuah citra kaya informasi, namun seringkali citra yang kita miliki mengalami penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang.

Citra digital merupakan suatu *array* dua dimensi atau suatu matriks yang elemen-elemennya menyatakan tingkat keabuan dari elemen gambar. Jadi informasi yang terkandung bersifat diskrit dan dapat didefinisikan sebagai fungsi dua variabel $f(x,y)$, dimana x dan y adalah koordinat spasial dan nilai $f(x,y)$ adalah intensitas citra pada koordinat tersebut, hal tersebut dapat dilihat pada gambar 2.2 berikut.



Gambar 2.2 Representasi Citra Digital dalam 2 Dimensi

(R. Kusumanto et. al., 2011)

Salah satu sistem yang digunakan untuk mewakili gambar yaitu sistem warna RGB (*Red, Green, Blue*). Sistem RGB adalah sistem yang menggabungkan warna primer gabungan (*additive primary colours*) untuk memperoleh gabungan-gabungan warna. Berikut ini adalah tabel 2.1 yaitu tabel warna yang merupakan gabungan warna primer.

Tabel 2.1 Kode Warna

Warna	<i>Red</i>	<i>Green</i>	<i>Blue</i>
Black	0	0	0
<i>Blue</i>	0	0	255
<i>Green</i>	0	255	0
<i>Red</i>	255	0	0
<i>Cyan (Green + Blue)</i>	0	255	255
<i>Magenta (Red + Blue)</i>	255	0	255
<i>Yellow (Red + Green)</i>	255	255	0
<i>White (Red + Green + Blue)</i>	255	255	255

(R. Kusumanto et. al., 2011)

2.2 Pengenalan Kriptografi

Jika anda bertukar pesan (misalnya surat) dengan orang lain, maka anda tentu ingin pesan yang anda kirim sampai ke pihak yang dituju dengan aman. Pengertian aman di sini sangat luas. Aman bisa berarti bahwa selama pengiriman pesan tentu anda berharap pesan tersebut tidak dibaca oleh orang yang tidak berhak. Sebab, mungkin saja pesan yang anda kirim berisi sesuatu yang rahasia sehingga jika pesan rahasia dibaca oleh pihak lawan atau pihak yang tidak berkepentingan, maka bocorlah kerahasiaan pesan yang anda kirim. Ini adalah masalah keamanan pesan yang dinamakan kerahasiaan (*confidentiality* atau *privacy*) (Munir, 2006).

Aman bisa juga berarti bahwa anda ingin pesan yang dikirim sampai dengan utuh ke tangan penerima, artinya isi pesan tidak diubah atau dimanipulasi selama pengiriman oleh pihak ketiga. Di sisi penerima pesan, ia tentu ingin memastikan bahwa pesan yang ia terima adalah pesan yang masih asli, bukan pesan yang sudah ditambah-tambah atau dikurangi. Ini adalah masalah keamanan pesan yang disebut integritas data (*data integrity*). Selain itu, penerima yakin bahwa pesan tersebut

memang benar berasal dari anda, bukan dari orang lain yang menyamar seperti anda, dan anda pun yakin bahwa orang yang anda kirim pesan adalah orang yang sesungguhnya. Ini adalah masalah keamanan pesan yang dinamakan otentikasi (*authentication*).

Jika anda sebagai penerima pesan, anda pun tidak ingin kelak pengirim pesan membantah pernah mengirim pesan kepada anda. Ini adalah masalah keamanan yang disebut penyangkalan (*repudiation*). Zaman sekarang, banyak orang yang membantah telah mengirim atau menerima pesan. Padahal anda yakin bahwa anda memang menerima pesan dari orang tersebut. Jika pengirim membantah telah mengirim pesan, maka anda perlu membuktikan ketidakbenaran penyangkalan tersebut (*non-repudiation*).

Keempat masalah keamanan yang disebutkan di atas, yaitu kerahasiaan, integritas data, otentikasi dan penyangkalan dapat diselesaikan dengan menggunakan kriptografi. Kriptografi tidak hanya menyediakan alat untuk keamanan pesan, tetapi juga sekumpulan teknik yang berguna.

2.2.1 Definisi

Kriptografi (*cryptography*) berasal dari Bahasa Yunani: “cryptos” artinya “secret” (rahasia), sedangkan “graphein” artinya “writing” (tulisan). Jadi, kriptografi berarti “secret writing” (tulisan rahasia). Ada beberapa definisi kriptografi yang telah dikemukakan di dalam berbagai literatur, seperti:

1. Bruce Schneier di dalam bukunya ”Applied Cryptography” menyatakan bahwa: Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (*Cryptography is the art and science of keeping messages secure*).
2. Menezes, Alfred J., Paul C. van Oorschot dan Scott A. Vanstone dalam buku mereka ”Handbook of Applied Cryptography” menyatakan bahwa: Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi. (Munir, 2006)

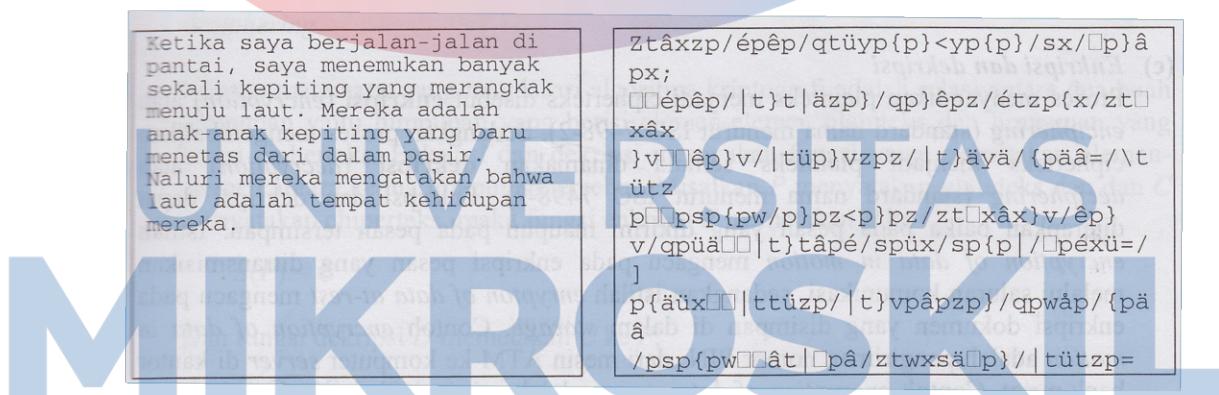
2.2.2 Terminologi

Di dalam kriptografi, akan sering ditemukan berbagai istilah atau terminologi. Beberapa istilah yang penting untuk diketahui diberikan di bawah ini.

1. Plainteks dan Cipherteks.

Pesan (*message*) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah plainteks (*plaintext*) atau teks-jelas (*cleartext*). Pesan dapat berupa data atau informasi yang dikirim (melalui kurir, saluran telekomunikasi) atau yang disimpan di dalam media perekaman (kertas, storage). Pesan yang tersimpan tidak hanya berupa teks, tetapi juga dapat berbentuk citra (*image*), suara/bunyi (*audio*) dan video atau berkas biner lainnya.

Agar pesan tidak dapat dimengerti maknanya oleh pihak lain, maka pesan perlu disandikan ke bentuk lain yang tidak dapat dipahami. Bentuk pesan yang tersandi disebut cipherteks (*ciphertext*) atau kriptogram (*cryptogram*). Cipherteks harus dapat ditransformasikan kembali menjadi plainteks semula agar pesan yang diterima bisa dibaca. Gambar 2.3 dan 2.4 memperlihatkan contoh dari dua buah plainteks, masing-masing berupa teks dan gambar, serta cipherteks yang berkoresponden (Munir, 2006).

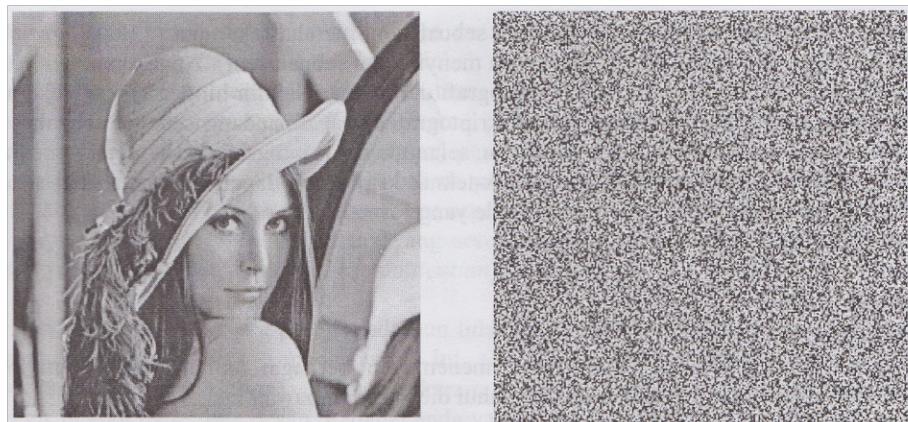


(a) *Plaintext*

(b) *Ciphertext*

Gambar 2.3 *Plaintext* dan *Ciphertext* berupa teks

(Munir, 2006)



(a) *Plaintext*

(b) *Ciphertext*

Gambar 2.4 *Plaintext* dan *Ciphertext* berupa gambar

(Munir, 2006)

2. Pengirim dan penerima

Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim (*sender*) adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima (*receiver*) adalah entitas yang menerima pesan. Entitas di sini dapat berupa orang, mesin (komputer), kartu kredit, dan sebagainya. Jadi, orang bisa bertukar pesan dengan orang lainnya (contoh: Alice berkomunikasi dengan Bob), sedangkan di dalam jaringan komputer, mesin (komputer) berkomunikasi dengan mesin (contoh: mesin ATM berkomunikasi dengan komputer *server* di bank).

Pengirim tentu menginginkan pesan dapat dikirim secara aman, yaitu ia yakin bahwa pihak lain tidak dapat membaca isi pesan yang ia kirim. Solusinya adalah dengan cara menyandikan pesan menjadi cipherteks. (Munir, 2006)

3. Enkripsi dan dekripsi

Proses menyandikan plainteks menjadi cipherteks disebut enkripsi (*encryption*) atau *enciphering*. Sedangkan, proses mengembalikan cipherteks menjadi plainteks semula dinamakan dekripsi (*decryption*) atau *deciphering*. Enkripsi dan dekripsi dapat diterapkan baik pada pesan yang dikirim maupun pada pesan tersimpan. Istilah *encryption of data in motion* mengacu pada enkripsi pesan yang ditransmisikan melalui saluran komunikasi, sedangkan istilah *encryption of data at-rest* mengacu pada enkripsi dokumen yang disimpan di dalam *storage*.

Contoh *encryption of data in motion* adalah pengiriman nomor PIN dari mesin ATM ke komputer *server* di kantor bank pusat. Contoh *encryption of data at-rest* adalah enkripsi *file* basis data di dalam *hard disk*. Gambar 2.5 memperlihatkan enkripsi *file* basis data, di mana enkripsi hanya dilakukan terhadap *field-field* tertentu saja (Nama, Tinggi dan Berat) (Munir, 2006).

NIM	Nama	Tinggi	Berat
000001	Elin Jamilah	160	50
000002	Fariz RM	157	49
000003	Taufik Hidayat	176	65
000004	Siti Nurhaliza	172	67
000005	Oma Irama	171	60
000006	Aziz Burhan	181	54
000007	Santi Nursanti	167	59
000008	Cut Yanti	169	61
000009	Ina Sabarina	171	62

NIM	Nama	Tinggi	Berat
000001	tüp}vzp/z t}äyä/{ää	äzp}	épêp
000002	□□ t tâpé/spüx/sp	péxü=	ztwxsa□
000003	□□åt □pâ/ztxwsä□p}/	} tü	spüx/
000004	épêp/ t t äzp}/qpépz	qp}êpz	wxsä
000005	étpz{x/ztx□xâx)v □ép}	pää/psp	étpz{
000006	spüx/sp{p /□péxü=/]	xâx)v	ttüzp
000007	Ztâxzp/épêp/qtüypp}<	äzp}	}äyä/{
000008	qpwâp/{pää/psp{pw□	Ztxws	xâx)v□
000009	t äzp}/qp}êpz/ép{	qp}êp	äzp}/qp

Gambar 2.5 Enkripsi Data tertentu di dalam Arsip Basisdata
(Munir, 2006)

4. *Cipher* dan kunci

Algoritma kriptografi disebut juga *cipher* yaitu aturan untuk *enciphering* dan *deciphering*, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa *cipher* memerlukan algoritma yang berbeda untuk *enciphering* dan *deciphering*. Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara dua buah himpunan yaitu himpunan yang berisi elemen-elemen plainteks dan himpunan yang berisi cipherteks. Enkripsi dan dekripsi merupakan fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan P menyatakan plainteks dan C menyatakan cipherteks, maka fungsi enkripsi E memetakan P ke C ,

$$E(P) = C \quad (1)$$

Dan fungsi dekripsi D memetakan C ke P,

$$D(C) = P \quad (2)$$

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan asal, maka kesamaan berikut harus benar (Munir, 2006):

$$D(E(P)) = P \quad (3)$$

Keamanan algoritma kriptografi sering diukur dari banyaknya kerja (*work*) yang dibutuhkan untuk memecahkan cipherteks menjadi plainteksnya tanpa mengetahui kunci yang digunakan. Kerja ini dapat diequivalekan dengan waktu, memori, uang, waktu, dan lain-lain. Semakin banyak kerja yang diperlukan, yang berarti juga semakin lama waktu yang dibutuhkan, maka semakin kuat algoritma kriptografi tersebut, yang berarti semakin aman digunakan untuk menyandikan pesan.

Jika keamanan kriptografi ditentukan dengan menjaga kerahasiaan algoritmanya, maka algoritma kriptografinya dinamakan algoritma *restricted*. Algoritma *restricted* mempunyai sejarah tersendiri di dalam kriptografi. Algoritma *restricted* biasanya digunakan oleh sekelompok orang untuk bertukar pesan satu sama lain. Mereka membuat suatu algoritma enkripsi dan algoritma enkripsi tersebut hanya diketahui oleh anggota kelompok itu saja. Tetapi, algoritma *restricted* tidak cocok lagi saat ini, sebab setiap kali ada anggota kelompok keluar, maka algoritma kriptografi harus diganti lagi.

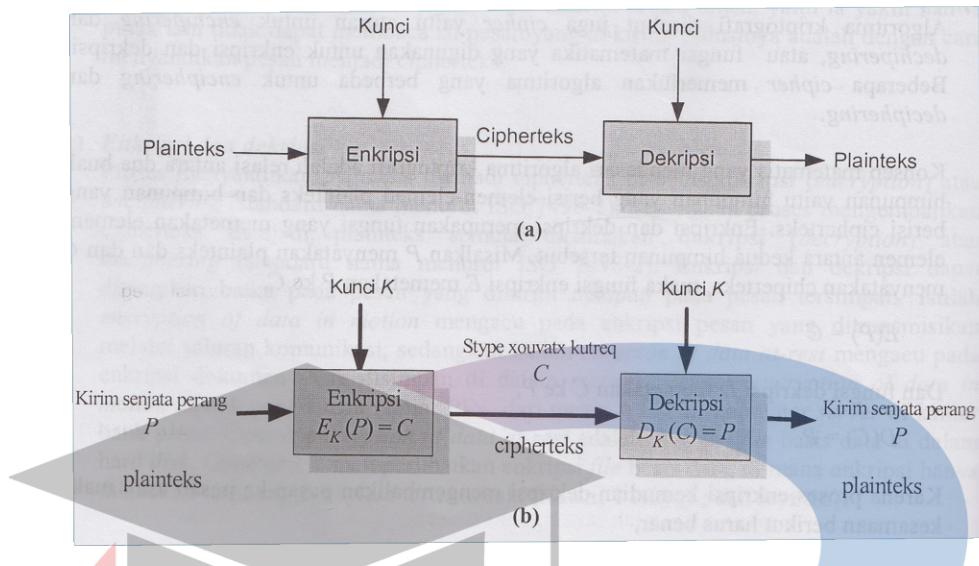
Kriptografi modern mengatasi masalah di atas dengan penggunaan kunci, yang dalam hal ini algoritma tidak lagi dirahasiakan, tetapi kunci harus dijaga kerahasiaannya. Kunci (*key*) adalah parameter yang digunakan untuk transformasi *enciphering* dan *deciphering*. Kunci biasanya berupa *string* atau deretan bilangan. Dengan menggunakan kunci K, maka fungsi enkripsi dan dekripsi dapat ditulis sebagai :

$$E_K(P) = C \text{ dan } D_K(C) = P \quad (4)$$

dan kedua fungsi ini memenuhi :

$$D_K(E_K(P)) = P \quad (5)$$

Gambar 2.6 memperlihatkan skema enkripsi dan dekripsi dengan menggunakan kunci.



Gambar 2.6 Skema Enkripsi dan Dekripsi dengan Menggunakan Kunci
(Munir, 2006)

2.3 Sistem Kriptografi

Kriptografi membentuk sebuah sistem yang dinamakan sistem kriptografi. Sistem kriptografi (*cryptosystem*) adalah kumpulan yang terdiri dari algoritma kriptografi, semua plainteks dan cipherteks yang mungkin dan kunci. Di dalam sistem kriptografi, *cipher* hanyalah salah satu komponen saja.

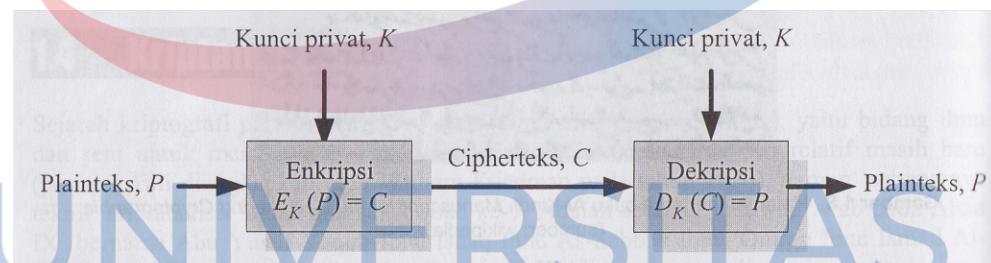
Berdasarkan kunci yang digunakan untuk enkripsi dan dekripsi, kriptografi dapat dibedakan menjadi kriptografi kunci-simetri (*symmetric-key cryptography*) dan kriptografi kunci-nirsimetri (*asymmetric-key cryptography*). (Munir, 2006)

2.3.1 Kriptografi Kunci Simetri

Pada sistem kriptografi kunci-simetri, kunci untuk enkripsi sama dengan kunci untuk dekripsi, oleh karena itulah dinamakan kriptografi simetri. Istilah lain untuk kriptografi kunci-simetri adalah kriptografi kunci privat (*private-key cryptography*), kriptografi kunci rahasia (*secret-key cryptography*). Sistem kriptografi kunci-simetri (atau disingkat menjadi "kriptografi simetri"), mengasumsikan pengirim dan penerima pesan sudah berbagi kunci yang sama sebelum bertukar pesan. Keamanan sistem kriptografi simetri terletak pada kerahasiaan kuncinya. Kriptografi simetri merupakan satu-satunya jenis kriptografi

yang dikenal dalam catatan sejarah hingga pada tahun 1976. Beberapa algoritma kriptografi modern yang termasuk ke dalam sistem kriptografi simetri, diantaranya adalah *Data Encryption Standard* (DES), *Blowfish*, *Twofish*, *Triple-DES*, *International Data Encryption Standard* (IDES), *Serpent*, dan yang terbaru adalah *Advanced Encryption Standard* (AES) (Munir, 2006).

Secara umum, *cipher* yang termasuk ke dalam kriptografi simetri beroperasi dalam mode blok (*block cipher*), yaitu setiap kali enkripsi/dekripsi dilakukan terhadap satu blok data (yang berukuran tertentu), atau beroperasi dalam mode aliran (*stream cipher*), yaitu setiap kali enkripsi/dekripsi dilakukan terhadap satu *bit* atau satu *byte* data. Aplikasi kriptografi sismetri yang utama adalah melindungi kerahasiaan data yang dikirim melalui saluran tidak aman dan melindungi kerahasiaan data yang disimpan pada media yang tidak aman. Kelemahan dari sistem ini adalah baik pengirim maupun penerima pesan harus memiliki kunci yang sama, sehingga pengirim pesan harus mencari cara yang aman untuk memberitahukan kunci kepada penerima pesan. Gambaran skema kriptografi simetris dapat dilihat pada gambar 2.7:



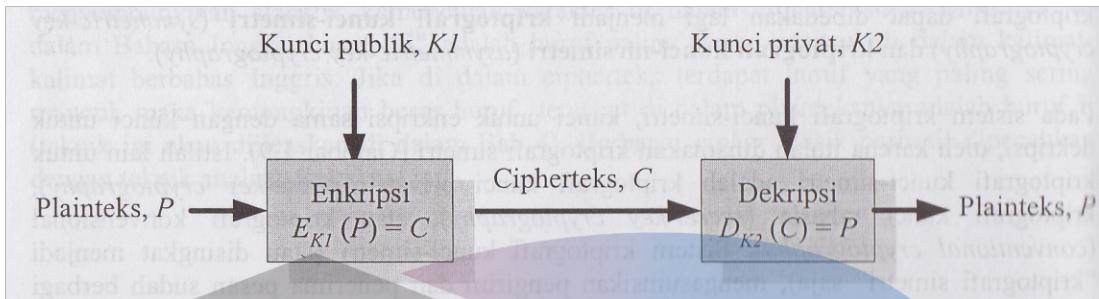
Gambar 2.7 Skema Kriptografi Simetris

(Munir, 2006)

2.3.2 Kriptografi Kunci Asimetri

Jika kunci enkripsi tidak sama dengan kunci untuk dekripsi, maka kriptografinya dinamakan sistem kriptografi Asimetri. Nama lainnya adalah kriptografi kunci-publik (*public-key cryptography*), sebab kunci untuk enkripsi tidak rahasia dan dapat diketahui oleh siapapun (diungkapkan ke publik), sementara kunci untuk dekripsi hanya diketahui oleh penerima pesan (karena itu rahasia). Pada kriptografi jenis ini, setiap orang yang berkomunikasi mempunyai sepasang kunci, yaitu kunci privat dan kunci publik. Pengirim mengenkripsi pesan dengan

menggunakan kunci publik si penerima pesan (*receiver*). Hanya penerima pesan yang dapat mendekripsi pesan karena hanya ia yang mengetahui kunci privatnya sendiri. Gambaran dari skema kriptografi Asimetri ini dapat dilihat pada gambar 2.8:



Gambar 2.8 Skema Kriptografi Asimetris
(Munir, 2006)

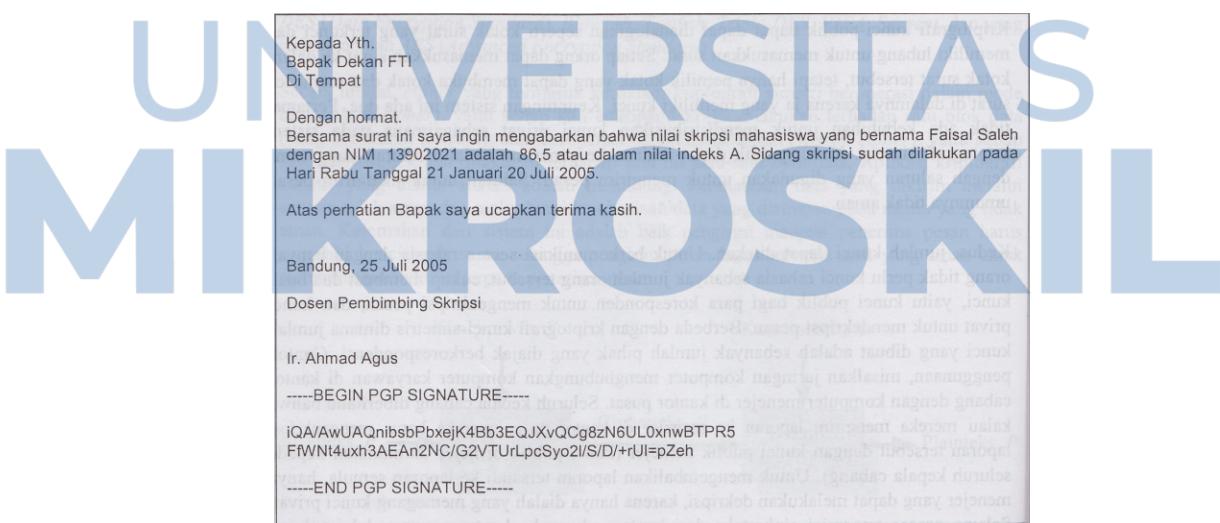
Contoh algoritma kriptografi kunci-publik diantaranya RSA, Elgamal, DSA, NTRU dan sebagainya (Munir, 2006).

Kriptografi kunci-publik dapat dianalogikan seperti kotak surat yang terkunci dan memiliki lubang untuk memasukkan surat. Setiap orang dapat memasukkan surat ke dalam kotak surat tersebut, tetapi hanya pemilik kotak yang dapat membuka kotak dan membaca surat di dalamnya karena ia yang memiliki kunci. Keuntungan sistem ini ada dua. Pertama, tidak ada kebutuhan untuk mendistribusikan kunci privat sebagaimana pada sistem kriptografi simetri. Kunci publik dapat dikirim ke penerima melalui saluran yang sama dengan saluran yang digunakan untuk mengirim pesan, saluran untuk mengirim pesan umumnya tidak aman.

Kedua, jumlah kunci dapat ditekan. Untuk berkomunikasi secara rahasia dengan banyak orang tidak perlu kunci rahasia sebanyak jumlah orang tersebut, cukup membuat dua buah kunci, yaitu kunci publik bagi para koresponden untuk mengenkripsi pesan, dan kunci privat untuk mendekripsi pesan. Berbeda dengan kriptografi kunci-simetris dimana jumlah kunci yang dibuat adalah sebanyak jumlah pihak yang diajak berkorespondensi. Contoh penggunaan, misalkan jaringan komputer menghubungkan komputer karyawan di kantor cabang dengan komputer manajer di kantor pusat. Seluruh kepala cabang diberitahukan bahwa kalau mereka mengirim laporan ke manajer di kantor pusat, mereka harus

mengenkripsi laporan tersebut dengan kunci publik manajer (kunci publik manajer diumumkan kepada seluruh kepala cabang). Untuk mengembalikan laporan tersandi ke laporan semula, hanya manajer yang dapat melakukan dekripsi, karena hanya dia lah yang memegang kunci privat. Selama proses transmisi cipherteks dari kantor cabang ke kantor pusat melalui saluran komunikasi, mungkin saja data yang dikirim disadap oleh pihak ketiga, namun pihak ketiga ini tidak dapat mengembalikan cipherteks ke plainteksnya karena ia tidak mengetahui kunci untuk dekripsi.

Kriptografi kunci-publik mempunyai kontribusi yang luar biasa dibandingkan dengan sistem kriptografi simetri. Kontribusi yang paling penting adalah tanda-tangan digital pada pesan untuk memberikan aspek keamanan otentifikasi dan integritas data. Tanda-tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci yang digunakan. Pengirim pesan mengenkripsi pesan (yang sudah diringkas) dengan kunci privatnya, hasil enkripsi inilah yang dinamakan tanda-tangan digital. Tanda –tangan digital dilekatkan (*embed*) pada pesan asli. Penerima pesan memverifikasi tanda-tangan digital dengan menggunakan kunci publik. Gambar 2.9 memperlihatkan sebuah surat elektronik yang di bagian bawah sudah dibubuhi tanda-tangan digital (di antara *BEGIN* dan *END SIGNATURE*) (Munir, 2006).



Gambar 2.9 Sebuah Surat yang Dibubuhi Tanda Tangan Digital
(Munir, 2006)

2.4 Landasan Matematika Kriptografi

Adapun landasan matematika kriptografi terdiri dari permutasi, substitusi, XOR, pergeseran bit (*shift*), rotasi bit (*rotate*).

2.4.1 Permutasi (*Permutation*)

Permutasi merupakan suatu proses korespondensi dari satu ke banyak. Dalam kriptografi, permutasi sering digunakan untuk memindahkan posisi bit ke posisi yang telah ditentukan dalam tabel permutasi. Pada umumnya, permutasi dalam kriptografi sering digunakan pada awal (*initial permutation*) dan akhir (*final permutation*) dari proses enkripsi dan dekripsi. Kadang – kadang juga digunakan dalam menghasilkan beberapa subkunci yang diperlukan dalam proses enkripsi dan dekripsi, seperti contoh berikut ini:

Misalkan diberikan tabel permutasi 16 bit berikut:

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

Tabel permutasi diatas akan diimplementasikan pada barisan bit input 1111000011110000:

Bit ke-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0

Hasil Permutasi:

Bit ke-	12	8	4	0	14	10	6	2	13	9	5	1	15	11	7	3
Bit	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

2.4.2 Substitusi

Substitusi *cipher* adalah *block cipher* yang menggantikan simbol dengan simbol lain. Contoh, jika A adalah sejumlah alfabet pengganti dan M adalah string yang akan dienkripsi, maka:

$$E(m) = (e(m_1)e(m_2) \dots e(m_t)) = (c_1 c_2 \dots c_t) \quad (6)$$

Dimana $m = (m_1, m_2, \dots, m_t) \in M$. Dengan kata lain, setiap simbol pada M akan diganti (disubstitusi) dengan simbol lain dari A. Untuk mendekripsi pesan *cipher* maka:

$$D(c) = (d(c_1) \ d(c_2) \dots \ d(c_t)) = (m_1 \ m_2 \dots \ m_t) \quad (7)$$

Contoh dari substitusi *cipher* adalah Julius Caesar *cipher*. Dimana substitusi diperoleh dengan cara menggeser huruf alfabet sebanyak 3 karakter ke kanan sehingga huruf A diubah menjadi huruf D, huruf B diubah menjadi huruf E dan seterusnya. Jika pesan yang dikirim adalah “KRIPTOGRAFI” maka chipertextnya adalah “NULSWRJUDIL” (Bruce Schneier, 1996).

2.4.3 XOR

XOR adalah operasi Exclusive-OR yang dilambangkan dengan “ \oplus ”. Dalam operasi XOR akan menghasilkan nilai bit “0” untuk dua nilai bit yang sama dan nilai bit “1” untuk dua nilai bit yang berbeda, seperti terlihat pada tabel 2.2 berikut:

Tabel 2.2 Tabel Operasi XOR untuk bit

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(Bruce Schneier, 1996).

2.4.4 Pergeseran Bit (*Shift*)

Pergeseran bit (*shift*) merupakan operasi terhadap bit dengan menggeser suatu deretan bit sebanyak yang diinginkan. Bit yang telah tergeser akan diberikan nilai nol. Pergeseran bit terbagi atas:

1. Operasi geser kiri (*shift left*)

Operasi geser kiri dilakukan dengan cara menggeser deretan bit ke kiri sebanyak nilai geser yang diberikan, kemudian bit kosong yang telah tergeser di sebelah kanannya akan diberikan nilai nol. Operasi geser kiri dilambangkan dengan “ $<<$ ”. Berikut ini adalah contoh operasi geser kiri :

0101 0101 $<< 1 \Rightarrow 1010 1010$

0101 0101 $<< 2 \Rightarrow 0101 0100$

2. Operasi geser kanan (*shift right*)

Operasi geser kanan dilakukan dengan cara menggeser deretan bit ke kanan sebanyak nilai geser yang diberikan, kemudian bit kosong yang telah tergeser di sebelah kirinya akan diberikan nilai nol. Operasi geser kanan dilambangkan dengan “>>”. Berikut ini adalah contoh operasi geser kanan (Bruce Schneier, 1996):

0101 0101 >> 1 => 0010 1010

0101 0101 >> 2 => 0001 0101

2.4.5 Rotasi Bit (*Rotate*)

Rotasi bit merupakan operasi bit yang dilakukan dengan memutar deretan bit sebanyak yang diinginkan. Bit yang telah tergeser tidak akan hilang karena hanya dipindahkan ke sisi deretan bit yang berlawanan dengan arah putaran bit. Rotasi bit terbagi atas :

1. Operasi rotasi kiri (*rotate left*)

Operasi rotasi kiri dilakukan dengan cara memutar deretan bit ke kiri sebanyak nilai rotasi yang diberikan, kemudian bit kosong yang telah tergeser di sebelah kanannya akan digantikan dengan bit yang telah tergeser disebelah kirinya. Operasi rotasi kiri dilambangkan dengan “<<<”. Berikut ini adalah contoh operasi rotasi kiri :

0101 0111 <<< 1 => 1010 1110

0101 0111 <<< 2 => 0101 1101

2. Operasi rotasi kanan (*rotate right*)

Operasi rotasi kanan dilakukan dengan cara memutar deretan bit ke kanan sebanyak nilai rotasi yang diberikan, kemudian bit kosong yang telah tergeser di sebelah kirinya akan digantikan dengan bit yang telah tergeser disebelah kanannya. Operasi rotasi kanan dilambangkan dengan “>>>”. Berikut ini adalah contoh operasi rotasi kanan :

0101 0111 >>> 1 => 1010 1011

0101 0111 >>> 2 => 1101 0101

2.5 Algoritma AES (*Advanced Encryption Standard*)

AES merupakan nama untuk *Federal Information Processing Standards Publication* 197. AES menjelaskan mengenai algoritma kriptografi yang digunakan untuk melindungi data sebagai pengganti DES. Algoritma AES adalah sebuah *symmetric block cipher* yang dapat melakukan enkripsi dan dekripsi pada data. Algoritma AES dapat menggunakan kunci 128, 192 dan 256 bit untuk melakukan enkripsi dan dekripsi terhadap data dengan ukuran blok 128 bit.

The *Advanced Encryption Standard* (AES) dengan metode Rijndael ini diperkenalkan oleh 2 orang kriptografer asal Belgia yaitu Joan Daemen dan Vincent Rijmen. Karena menggunakan kunci dengan ukuran 128, 192 atau 256 bit maka algoritma ini sering disebut dengan "AES-128", "AES-192", atau "AES-256" sesuai dengan kunci yang dipakai, sedangkan ukuran blok yang digunakan adalah 128 bit (FIPS 197, 2001).

2.5.1 Notasi

1. Input dan Output

Input dan *Output* pada algoritma AES terdiri dari deretan 128 bit, deretan ini disebut juga dengan blok dan jumlah dari bit akan menentukan panjang dari blok. Kunci untuk algoritma AES adalah deretan dari 128, 192 atau 256 bit.

2. Byte

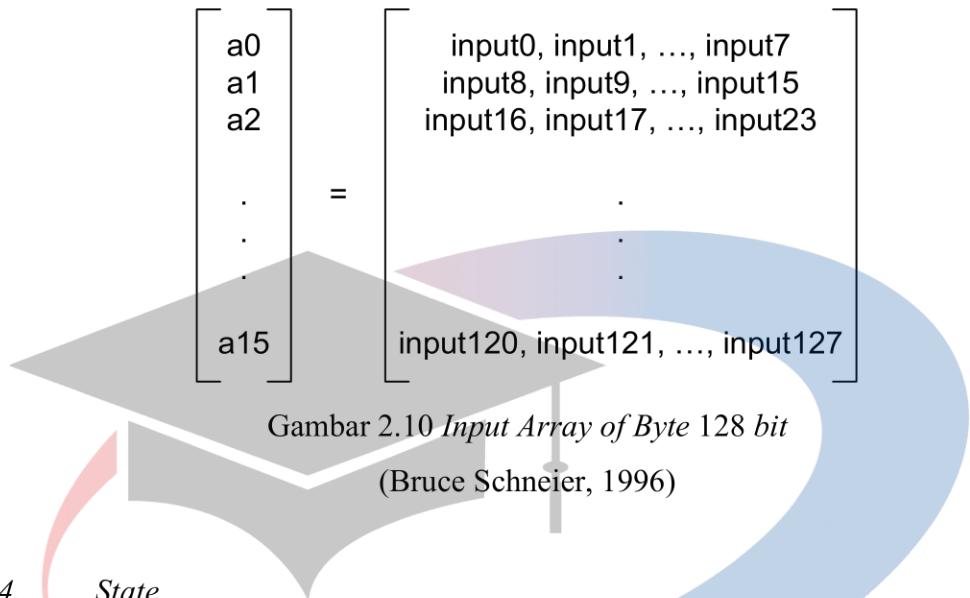
Unit dasar yang digunakan untuk proses di dalam algoritma AES adalah *byte*, yaitu deretan yang terdiri dari 8 bit. Setiap nilai *byte* pada algoritma AES dinyatakan sebagai konkatenasi dari nilai masing-masing bitnya (0 atau 1) yang disusun dengan $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. *Byte - byte* ini dianggap sebagai elemen dari *finite field* dengan menggunakan representasi *polynomial*:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 = \sum b_i x^i \quad (8)$$

Akan lebih mudah bila menggunakan notasi heksadesimal dimana setiap dua grup yang terdiri atas 4 bit dinotasikan dengan sebuah karakter.

3. Array of Byte

Array of byte direpresentasikan dalam bentuk $a_0 \ a_1 \ a_2 \dots \ a_{15}$ dimana untuk 128 bit *input*, yang terdiri dari $input_0 \ input_1 \ input_2 \dots \ input_{127}$ yang ditunjukkan pada gambar 2.10 sebagai berikut :



4. State

Operasi pada algoritma AES dijalankan pada *array* dua dimensi yang disebut dengan *State*. *State* terdiri dari empat baris yang berisi Nb *byte*, dimana Nb adalah panjang blok dibagi dengan 32. *State* diberi simbol “*s*”, setiap *byte* mempunyai dua nilai index yaitu nomor baris “*r*” dimana $0 \leq r < 4$ dan nomor kolom “*c*” dimana $0 \leq c < Nb$. *State* dapat ditulis dengan $s_{r,c}$ atau $s[r,c]$.

5. State sebagai array of kolom

Empat *byte* dalam setiap kolom dari *State array* membentuk 32 bit (1 word), nomor baris *r* menunjukkan sebuah *index* untuk keempat *byte*. *State* dapat digambarkan sebagai array 1 dimensi yang terdiri atas 32 bit yaitu w_0, \dots, w_3 , nomor kolom *c* menunjukkan sebuah *index* pada *array*. Seperti ditunjukan pada gambar 2.11 berikut :

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} s_{0,0} \ s_{1,0} \ s_{2,0} \ s_{3,0} \\ s_{0,1} \ s_{1,1} \ s_{2,1} \ s_{3,1} \\ s_{0,2} \ s_{1,2} \ s_{2,2} \ s_{3,2} \\ s_{0,3} \ s_{1,3} \ s_{2,3} \ s_{3,3} \end{bmatrix}$$

Gambar 2.11 State Array terdiri atas 4 word

(Bruce Schneier, 1996)

2.5.2 Konsep Matematika pada Algoritma AES

Setiap *byte* dalam algoritma AES diperlakukan sebagai elemen dalam *finite field*. Elemen-elemen *finite field* dapat dijumlah dan dikali, tetapi operasi penjumlahan dan perkalian yang dilakukan berbeda dengan operasi penjumlahan dan perkalian pada bilangan.

1. Penjumlahan (*Addition*)

Penjumlahan pada algoritma AES dilakukan dengan menggunakan operasi XOR yang dilambangkan dengan “ \oplus ” dimana dapat dilihat dari tabel 2.2 diatas:

Contoh berikut memiliki nilai yang sama antara satu dengan lainnya :

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \quad (\text{Notasi Polynomial})$$

$$\{01010111\} \oplus \{10000011\} = \{11010100\} \quad (\text{Notasi Biner})$$

$$\{57\} \oplus \{83\} = \{d4\} \quad (\text{Notasi Heksadesimal})$$

2. Perkalian (*Multiplication*)

Dalam representasi *polynomial*, perkalian dalam $GF(2^8)$ (dinotasikan dengan “ \bullet ”) berhubungan dengan perkalian *polynomial* modulo *irreducible polynomial* berpangkat 8. *Polynomial* dikatakan *irreducible* jika hanya dapat dibagi dengan 1 dan dengan dirinya sendiri. Untuk algoritma AES, *irreducible polynomial* yang digunakan adalah:

$$m(x) = x^8 + x^4 + x^3 + x + 1 \text{ atau } \{01\} \{1B\} \text{ dalam heksadesimal.}$$

Sebagai contoh : $\{57\} \bullet \{83\} = \{c1\}$, karena

$$\{57\} = 01010111 = x^6 + x^4 + x^2 + x + 1$$

$$\{83\} = 10000011 = x^7 + x + 1$$

$$(x^6 + x^4 + x^2 + x + 1) (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

dan

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)$$

$$= x^7 + x^6 + 1$$

$$x^7 + x^6 + 1 = 11000001 = \{c1\}$$

Dengan notasi biner, hasilnya akan terlihat seperti berikut ini:

$$\begin{array}{r}
 57 = 01010111 \\
 83 = \underline{10000011} \\
 \hline
 01010111 \\
 01010111 \\
 \hline
 01010111 \oplus \\
 \hline
 10101101111001 \Rightarrow (\text{jumlah bit lebih dari 8 maka di-xor-kan dengan } 100011011) \\
 100011011 \oplus \\
 \hline
 100000011001 \Rightarrow (\text{jumlah bit lebih dari 8 maka di-xor-kan dengan } 100011011) \\
 100011011 \oplus \\
 \hline
 11000001 \Rightarrow c1
 \end{array}$$

3. *Polynomial* dengan koefisien pada $GF(2^8)$

Polynomial dengan koefisien dituliskan dengan $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ yang akan dinotasikan sebagai 1 word dalam bentuk $[a_0, a_1, a_2, a_3]$.

Penjumlahan dilakukan dengan menjumlahkan koefisien yang memiliki pangkat x yang sama. Penjumlahan menggunakan operasi XOR.

Contoh :

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

Perkalian dilakukan melalui 2 tahap. Tahap pertama yaitu perkalian $c(x) = a(x) \bullet b(x)$ secara aljabar dan pangkat yang sama disejajarkan sehingga menjadi:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

dimana:

$$c_0 = a_0 \bullet b_0$$

$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$$

$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$$

$$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

$$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$c_6 = a_3 \cdot b_3$$

Hasil yang diperoleh $c(x)$ tidak merepresentasikan empat *byte* (1 *word*). Oleh karena itu tahap kedua dari perkalian adalah memperkecil nilai $c(x)$ dengan memodulokan dengan *polynomial* berpangkat 4, untuk algoritma AES *polynomial* yang digunakan adalah x^4+1 , sehingga :

$$x^i \bmod (x^4+1) = x^{i \bmod 4} \quad (9)$$

Perkalian modular dari $a(x) \oplus b(x)$, dijabarkan sebagai berikut:

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$$

dimana:

$$d_0 = (a_0 \cdot b_0) \oplus (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$$

$$d_1 = (a_1 \cdot b_0) \oplus (a_0 \cdot b_1) \oplus (a_3 \cdot b_2) \oplus (a_2 \cdot b_3)$$

$$d_2 = (a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3)$$

$$d_3 = (a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3)$$

Jika $a(x)$ adalah *polynomial* yang mempunyai nilai tetap, operasi $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$

2.5.3 Spesifikasi Algoritma

Pada algoritma AES, panjang blok input, blok output dan *State* adalah 128 bit yang direpresentasikan dengan $Nb=4$ yang mana menunjukkan jumlah *word* pada *State*, sedangkan panjang kunci "K" adalah 128,192 atau 256 bit. Panjang kunci direpresentasikan dengan $Nk = 4, 6, \text{ atau } 8$ yang menunjukkan jumlah *word* pada kunci. Jumlah putaran pada algoritma AES tergantung pada ukuran kuncinya. Jumlah putaran direpresentasikan dengan Nr , dimana $Nr = 10$ jika $Nk = 4$, $Nr = 12$ jika $Nk = 6$, dan $Nr = 14$ jika $Nk = 8$ yang dapat dilihat tabel 2.3 berikut :

Tabel 2.3 Hubungan antara ukuran kunci, ukuran blok dengan jumlah putaran

	Ukuran Kunci (Nk word)	Ukuran Blok (Nb word)	Jumlah Putaran (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

(FIPS 197, 2001)

Pada algoritma AES, *key expansion* menghasilkan sebanyak $Nb(Nr+1)$ *word key schedule*. *Key schedule* terdiri dari sebuah *array 4-byte* yang dinotasikan dengan $[w_i]$ dimana nilai i berkisar pada *range* $0 \leq i < Nb(Nr+1)$. Fungsi-fungsi yang digunakan pada *key expansion* adalah sebagai berikut:

1. *SubWord*

Fungsi ini melakukan substitusi terhadap input yang terdiri atas 4 *byte* dengan tabel substitusi (S-box). Substitusi dilakukan secara *byte per byte*.

2. *RotWord*

Fungsi ini melakukan permutasi memutar (*cyclic permutation*) terhadap *input* $[a_0, a_1, a_2, a_3]$ sehingga diperoleh *output* $[a_1, a_2, a_3, a_0]$.

Contoh : $\text{RotWord}(09cf4f3c) \Rightarrow (\text{cf4f3c}09)$

3. *Rcon[i]*

Nilai *Rcon[i]* diperoleh dengan ketentuan $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, dimana x bernilai $\{02\}$ dalam bentuk heksadesimal.

Contoh :

$Rcon[1] = [02^{1-1}, \{00\}, \{00\}, \{00\}]$
$= [02^0, \{00\}, \{00\}, \{00\}]$
$= [01000000]$

Tabel 2.4 Rcon dengan I dimulai dari 1 sampai 10 dalam bentuk heksadesimal

i	r_{ci}
1	01
2	02
3	04
4	08
5	10
6	20
7	40
8	80
9	1B
10	36

(FIPS 197, 2001)

Dimana dengan batasan nilai i maksimal sebesar $Nb(Nr+1) - 1$.

2.5.4 Pembentukan Kunci

Langkah – langkah yang dilalui pada *key expansion* yaitu :

1. Membagi kunci menjadi blok – blok dimana masing – masing blok terdiri atas 1 *word* (32 bit). Untuk ukuran kunci 128 bit terdapat 4 *word* ($w[0], \dots, w[3]$),

untuk ukuran kunci 192 bit terdapat 6 word ($w[0], \dots, w[5]$) dan untuk ukuran kunci 256 bit terdapat 8 word ($w[0], \dots, w[7]$).

2. Untuk nilai i dengan range $\mathbf{Nk} \leq i < \mathbf{Nb}(\mathbf{Nr}+1)$, carilah $w[i]$ dengan ketentuan sebagai berikut :

- jika $(i \bmod \mathbf{Nk}) \neq 0$ dan $(i \bmod \mathbf{Nk}) \neq 4$ maka $w[i] = w[i-\mathbf{Nk}] \oplus w[i-1]$.
- jika $(i \bmod \mathbf{Nk}) = 0$ maka $w[i] = w[i-\mathbf{Nk}] \square (\text{SubWord}(\text{RotWord}(w[i-1])) \square Rcon[i/\mathbf{Nk}])$

3. $w[i] = w[i-\mathbf{Nk}] \oplus (\text{SubWord}(\text{RotWord}(w[i-1]))) \oplus Rcon[i/\mathbf{Nk}]$

- jika $(i \bmod \mathbf{Nk}) = 4$ maka $w[i] = w[i-\mathbf{Nk}] \square \text{Subword}(w[i-1])$

Subkunci (*round key*) diperoleh dengan ketentuan:

- Subkunci(i) = $(w[i*4], w[i*4+1], w[i*4+2], w[i*4+3])$; untuk $0 \leq i \leq \mathbf{Nr}$

2.5.5 Proses Enkripsi

Pada proses enkripsi, algoritma AES menggunakan empat transformasi yang berbeda yaitu:

1. *SubBytes()*

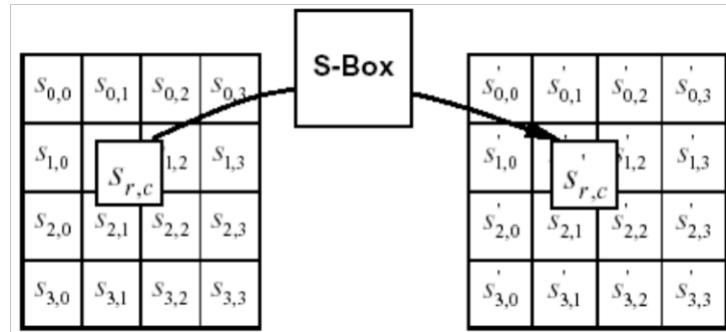
Transformasi *SubBytes()* merupakan substitusi *byte* yang beroperasi terhadap setiap *byte* pada *State* dengan menggunakan tabel substitusi (S-box).

Tabel 2.5 Tabel S-box

		y																
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
x		0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0	
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15	
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75	
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84	
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf	
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8	
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2	
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73	
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db	
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79	
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08	
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a	
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e	
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df	
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16	

(FIPS 197, 2001)

Adapun transformasi *SubByte* pada *State* dapat dilihat pada gambar 2.12 berikut:



Gambar 2.12 Transformasi *SubBytes()* pada *State*

(FIPS 197, 2001)

Sebagai contoh, jika $s_{1,1} = \{53\}$ maka nilai substitusi diperoleh dari perpotongan antara baris “5” dengan kolom “3” pada S-box sehingga didapat hasilnya {ed}.

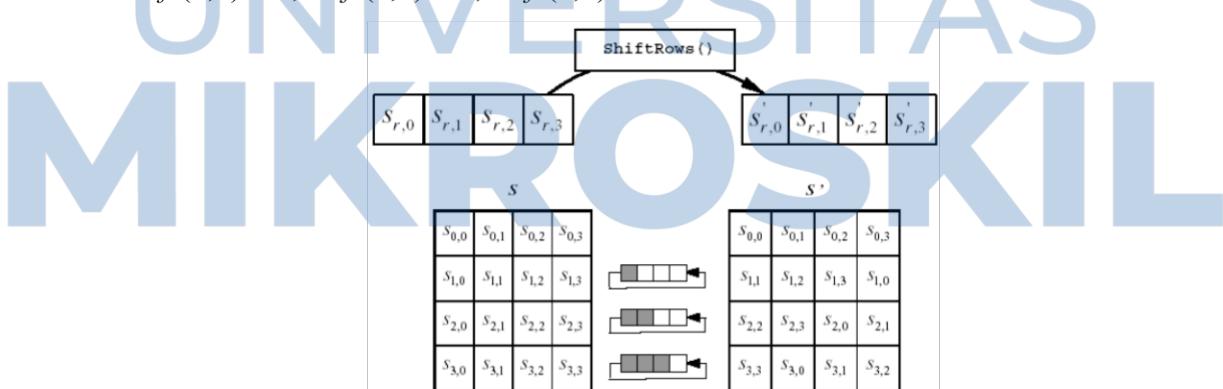
2. *ShiftRows()*

Transformasi *ShiftRows()* dilakukan pada 3 baris terakhir dengan melakukan geser (*shift*) memutar dengan nilai *shift* yang berbeda – beda tergantung kepada barisnya. Baris pertama ($r = 0$) tidak dilakukan operasi *ShiftRows()*. Secara spesifik, proses transformasi *ShiftRows()* adalah sebagai berikut:

$$S'^{r,c} = S_r, (c + \text{shift}(r, Nb)) \bmod Nb ; \text{ untuk } 0 < r < 4 \text{ dan } 0 \leq c < Nb$$

dimana nilai *shift*(r, Nb) tergantung pada nomor baris, untuk $Nb=4$ maka :

$$\text{shift}(1,4) = 1; \text{shift}(2,4) = 2; \text{shift}(3,4) = 3$$



Gambar 2.13 Transformasi *ShiftRows()* pada *State*

(FIPS 197, 2001)

3. *MixColumns()*

Transformasi *MixColumns()* dioperasikan pada *State* secara kolom per kolom, masing – masing kolom dikalikan dengan matriks yang sudah ditentukan.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{Untuk } 0 \leq c < \mathbf{Nb} \quad (10)$$

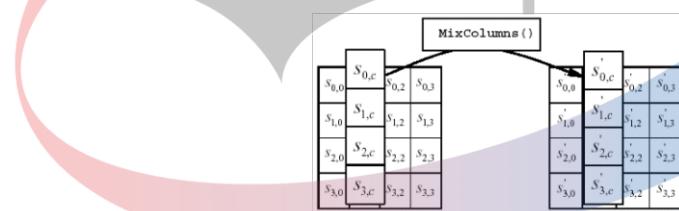
Hasil dari perkalian matriks diatas dapat dijabarkan sebagai berikut:

$$s'_{0,c} = (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c})$$

$$s'_{3,c} = (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{03\} \cdot s_{3,c})$$



Gambar 2.14 Transformasi *MixColumns()* pada *State*

(FIPS 197, 2001)

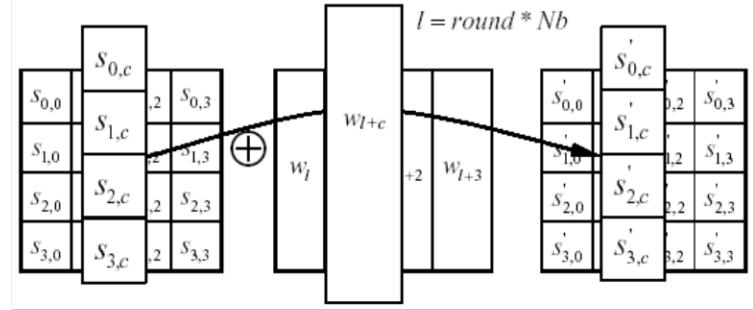
4. *AddRoundKey()*

Pada transformasi *AddRoundKey()*, sebuah subkunci ditambahkan pada *State* dengan operasi XOR. Setiap subkunci terdiri dari **Nb word** dari himpunan subkunci. Subkunci ditambahkan dengan *State* dengan cara sebagai berikut:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{\text{round}*Nb+c}] ; \text{untuk } 0 \leq c < \mathbf{Nb}$$

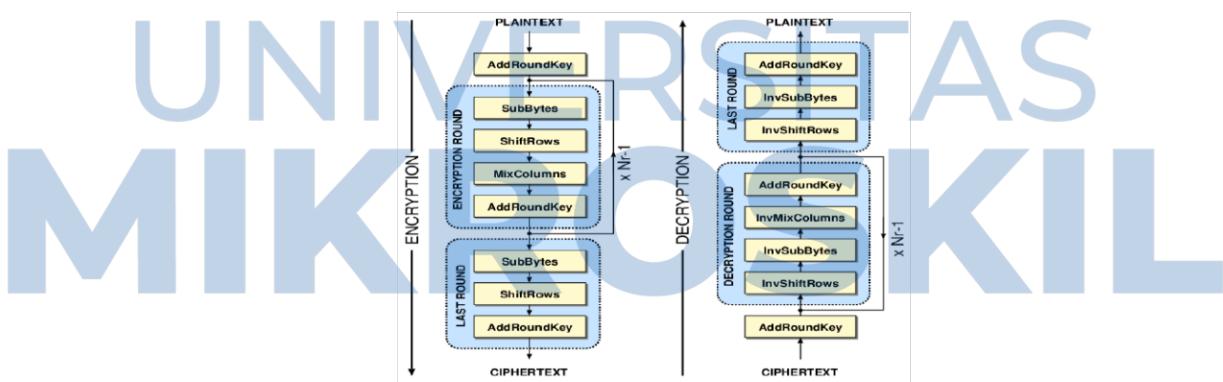
dimana: $[w_i] = \text{himpunan subkunci dalam satuan word}$

round = nilai yang berada pada range $0 \leq \text{round} \leq \mathbf{Nr}$



Gambar 2.15 Transformasi *AddRoundKey()* pada *State*
(FIPS 197, 2001)

Pada permulaan enkripsi, input yang berupa *plaintext* dimasukkan ke dalam *State*, pada *initial round* dilakukan transformasi *AddRoundKey(State, SubKunci(0))*, setelah *initial round* proses menuju pada *round function* sebanyak $\mathbf{Nr}-1$ putaran ($1 \leq \text{round} < \mathbf{Nr}$), dimana di dalam *round function* ini dilakukan transformasi berturut – turut yaitu *SubBytes()*, *ShiftRows()*, *MixColumns()*, dan *AddRoundKey()*. Setelah itu proses akan menuju pada putaran terakhir (*final round*) dimana pada putaran terakhir ini dilakukan transformasi *SubBytes()*, *ShiftRows()* dan *AddRoundKey()*, pada putaran terakhir ini setelah transformasi *AddRoundKey()* maka akan menghasilkan *final State* yang merupakan *output* yang disebut *ciphertext*. Proses enkripsi dan dekripsi dari metode AES dapat dilihat pada gambar 2.16:



Gambar 2.16 Proses Enkripsi dan Dekripsi Menggunakan AES
(Hameed, et. al, 2011)

2.5.6 Proses Dekripsi

Proses dekripsi pada algoritma AES merupakan kebalikan dari proses enkripsi. Transformasi yang digunakan pada proses dekripsi yaitu:

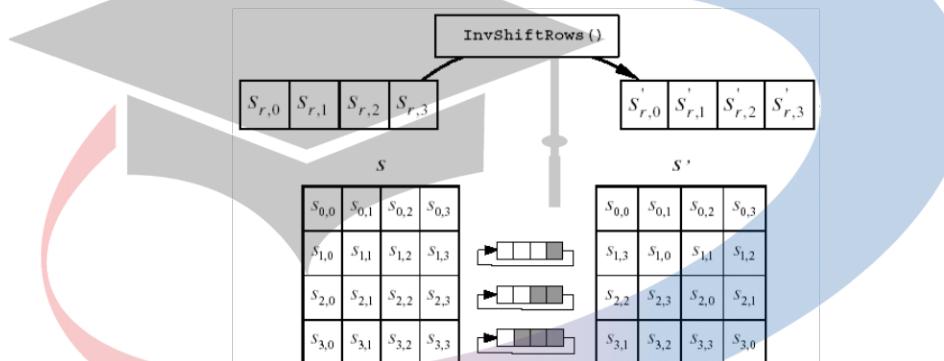
1. *InvShiftRows()*

Transformasi ini merupakan kebalikan dari transformasi *ShiftRows()* pada proses enkripsi. Transformasi *InvShiftRows()* dilakukan pada 3 baris terakhir dengan melakukan geser (*shift*) memutar dengan nilai *shift* yang berbeda-beda tergantung kepada barisnya. Baris pertama ($r = 0$) tidak dilakukan operasi *ShiftRows()*. Secara spesifik, proses transformasi *InvShiftRows()* adalah sebagai berikut:

$$S'_{r,c} = S_r, (c + (Nb - \text{shift}(r, Nb))) \bmod Nb ; \text{ untuk } 0 < r < 4 \text{ dan } 0 \leq c < Nb$$

dimana nilai *shift*(r, Nb) tergantung pada nomor baris, untuk $Nb=4$ maka :

$$\text{shift}(1,4) = 1; \text{ shift}(2,4) = 2; \text{ shift}(3,4) = 3$$



Gambar 2.17 Transformasi *InvShiftRows()* pada State
(FIPS 197, 2001)

2. *InvSubBytes()*

InvSubBytes() adalah kebalikan dari transformasi *SubBytes()*. Transformasi *InvSubBytes()* menggunakan S-box seperti pada tabel 2.6 :

Tabel 2.6 Tabel Inverse S-box

	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

(FIPS 197, 2001)

3. *InvMixColumns()*

Transformasi *InvMixColumns()* merupakan kebalikan dari transformasi *MixColumns()*, transformasi *InvMixColumns* dioperasikan pada *State* secara kolom per kolom, masing – masing kolom dikalikan dengan matriks yang sudah ditentukan.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \text{Untuk } 0 \leq c < \mathbf{Nb} \quad (11)$$

Hasil dari perkalian matriks diatas dapat dijabarkan sebagai berikut:

$$S'_{0,c} = (\{0e\} \cdot s_{0,c}) \oplus (\{0b\} \cdot s_{1,c}) \oplus (\{0d\} \cdot s_{2,c}) \oplus (\{09\} \cdot s_{3,c})$$

$$S'_{1,c} = (\{09\} \cdot s_{0,c}) \oplus (\{0e\} \cdot s_{1,c}) \oplus (\{0b\} \cdot s_{2,c}) \oplus (\{0d\} \cdot s_{3,c})$$

$$S'_{2,c} = (\{0d\} \cdot s_{0,c}) \oplus (\{09\} \cdot s_{1,c}) \oplus (\{0e\} \cdot s_{2,c}) \oplus (\{0b\} \cdot s_{3,c})$$

$$S'_{3,c} = (\{0b\} \cdot s_{0,c}) \oplus (\{0d\} \cdot s_{1,c}) \oplus (\{09\} \cdot s_{2,c}) \oplus (\{0e\} \cdot s_{3,c})$$

4. *AddRoundKey()*

Transformasi *AddRoundKey()* pada proses enkripsi dan dekripsi adalah sama saja karena hanya melibatkan operasi XOR.

Proses pendekripsi *ciphertext* menjadi *plaintext* dilakukan dengan cara yang tidak jauh berbeda dengan proses enkripsi, perbedaannya hanyalah pada urutan transformasi yang digunakan. Pada proses dekripsi, input yang berupa *ciphertext* dimasukkan ke dalam *State*, pada *initial round* dilakukan transformasi *AddRoundKey(State, SubKunci(Nr))*, setelah *initial round* proses menuju pada *round function* sebanyak *Nr-1* putaran ($1 \leq round < \mathbf{Nr}$), dimana di dalam *round function* ini dilakukan transformasi berturut – turut yaitu *InvShiftRows()*, *InvSubBytes()*, *AddRoundKey()*, dan *InvMixColumns()*. Setelah itu proses akan menuju pada putaran terakhir (*final round*) dimana pada putaran terakhir ini dilakukan transformasi *InvShiftRows()*, *InvSubBytes()*, dan *AddRoundKey()*, pada putaran terakhir ini setelah transformasi *AddRoundKey()* maka akan menghasilkan *final State* yang merupakan output yang disebut *plaintext*.

2.6 Algoritma Modified RK-MAES

Permasalahan utama pada ekspansi kunci dari algoritma AES adalah word w_1 dihasilkan dari kunci asli sehingga berhubungan satu sama lain. Jika salah satu *word* dapat dilacak, maka keseluruhan kunci dapat diperoleh dengan menggunakan metode differensial atau metode linier dari kriptanalisis. Untuk menyelesaikan permasalahan ini, maka modul ekspansi kunci dari AES akan dimodifikasi dengan *Symmetric Random Function Generator* (SRFG). SRFG menghasilkan *output* penyeimbang simetris dimana jumlah angka 1 dan 0 pada *string output* tidak berhubungan dengan *string input*. SRFG menghasilkan *output* yang merupakan hasil fungsi kombinasi dari *GATE* universal (*AND*, *OR*, *NOT* dan *XOR*). Ekspresi untuk generator fungsi kombinasi ini dapat dirumuskan sebagai berikut:

$$f_c = \otimes f_i^L \quad (12)$$

dimana:

f_c = fungsi gabungan generator ekspresi

i = 1, 2, ..., 4

L = panjang ekspresi (jumlah *term* pada fungsi kombinasi f_c)

\otimes = melambangkan kombinasi *random*

Saha, et. al. (2018) merekomendasikan $L = 5$.

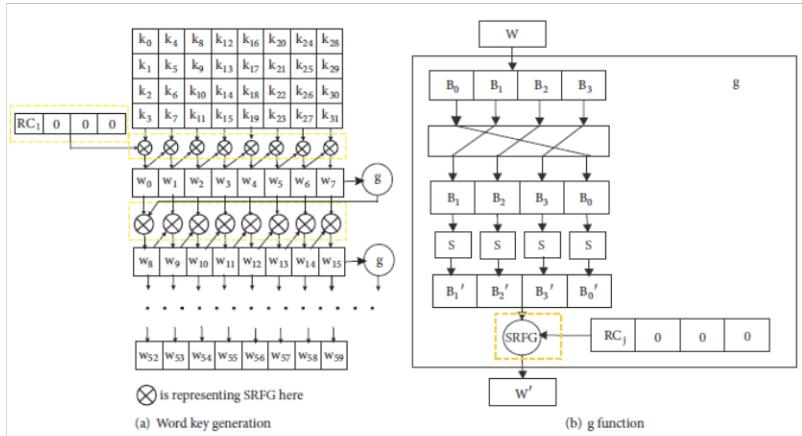
Untuk menekankan pengacakan pada generator fungsi kombinasi ini, persamaan diatas dapat diekspresikan dalam bentuk N buah *input* variabel pengacakan pada penyeleksian seperti terlihat pada persamaan berikut:

$$f_c(V_1, V_2, \dots, V_N) = \otimes f_i^L [\text{rand}(V_1, V_2, \dots, V_N)] \quad (13)$$

Saha, et. al. (2018) menggunakan persamaan berikut dalam percobaannya:

$$f_c(V_1, V_2) = \otimes f_i^5 [\text{rand}(V_1, V_2)] \quad (14)$$

Tujuan utama dari penambahan SRFG pada AES adalah menambahkan sifat pengacakan pada modul ekspansi kunci. Hal ini untuk menjaga agar *word* dari kunci tidak dapat dilacak walaupun telah diperoleh kunci parsial. Modifikasi modul ekspansi kunci dapat dilihat pada gambar 2.18 berikut:



Gambar 2.18 Modifikasi Ekspansi Kunci untuk 14 Putaran AES

(Geetha, et. al, 2018)

Untuk mengatasi masalah keamanan terhadap kunci pada metode AES yaitu terhadap beberapa serangan seperti *biased keys* dan *fault injection attacks*, yang dapat mengungkapkan sisi *random* dari metode AES sehingga membuat kunci rahasia bisa diketahui. Untuk mengatasi hal ini diciptakanlah metode yang memberikan keacakan dan fitur simetris yang seimbang dalam fungsi khusunya pada keamanan kunci (Geetha, et. al, 2018).

Geetha, et. al. telah menekankan modul pembangkitan kunci pada putaran AES-14, sehingga efek input bias dalam pemindaiannya *byte* kunci dapat dihapus dari deduksi keseluruhan *byte* kunci. Kunci dapat disimpulkan jika proses analisis kripto mampu menyimpulkan persamaan linear atau diferensial dari kata kunci yang dihasilkan dari modul ekspansi kunci. Untuk proses kriptonalisasi, tidak selalu perlu memiliki seluruh kunci di tangan tetapi jika ada satu saja bagian kunci yang berhubungan antara kata sudah cukup dalam mengungkapkan keseluruhan *keyspace* (Geetha, et. al, 2018). Sehingga untuk menentukan seberapa *random* metode RK-AES maka tidak boleh ada bagian kunci yang berhubungan antara kata yang digunakan.

Untuk mengatasi permasalahan dari masalah waktu, maka algoritma AES akan dimodifikasi untuk mengurangi kalkulasi dari algoritma dan untuk meningkatkan performansi dari proses enkripsi. Sasaran dari modifikasi AES adalah untuk menyediakan komputasi yang sedikit dan keamanan yang lebih baik dari data. Algoritma modifikasi AES menyediakan kecepatan proses enkripsi yang lebih baik. Pada modifikasi AES, panjang blok dan kunci dispesifikasikan

berdasarkan pada spesifikasi AES: tiga alternatif panjang kunci 128, 192 atau 256 bit dan panjang blok sebesar 128 bit (Hameed, et. al, 2011).

Modified AES memerlukan *input* sebesar 64 bit dan akan menggeser bit satu per satu berdasarkan pada tabel, seperti terlihat pada gambar 2.19 dan gambar 2.20 berikut:

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

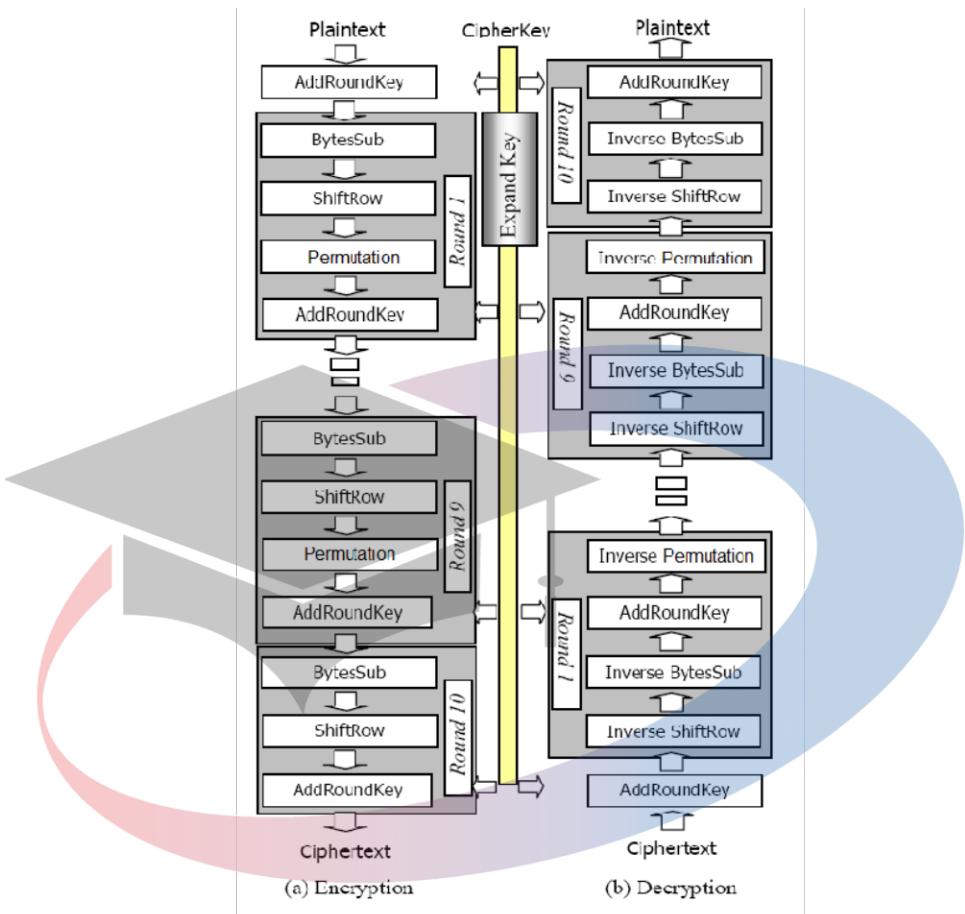
Gambar 2.19 Algoritma *Modified AES* : Tabel Bit Permutation
(Hameed, et. al, 2011)

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

Gambar 2.20 Algoritma *Modified AES* : Tabel Inverse Bit Permutation
(Hameed, et. al, 2011)

Pada metode modifikasi AES, proses enkripsi dan dekripsi memiliki spesifikasi yang sama dengan metode AES, dalam hal jumlah putaran, ukuran data dan kunci. Fungsi putaran terdiri dari empat tahapan. Untuk menyelesaikan permasalahan kalkulasi yang besar, tahapan *Mixcolumn* dibuang dan diganti dengan permutasi. *Mixcolumn* memberikan keamanan yang bagus, tetapi memerlukan kalkulasi yang besar, sehingga membuat proses enkripsi menjadi lambat. Sementara itu, tiga tahapan lainnya sama dengan metode AES. Sebuah blok tunggal sebesar 128 bit merupakan *input* dari algoritma enkripsi dan dekripsi. Blok ini merupakan sebuah matriks persegi berukuran 4 x 4 yang terdiri dari kumpulan *byte*. Blok ini akan diduplikasi ke *state array*. *State array* ini akan dimodifikasi pada setiap

tahapan enkripsi dan dekripsi. Proses kerja dari metode modifikasi AES ini dapat dilihat pada gambar berikut (Hameed, et. al, 2011):



Gambar 2.21 Algoritma *Modified AES* : Struktur Enkripsi dan Dekripsi
(Hameed, et. al, 2011)

2.7 Steganografi

Steganografi adalah seni menyembunyikan pesan rahasia dalam pesan lain sehingga keberadaan pesan rahasia tidak dapat diketahui orang lain (Munir, 2006). Kata steganografi berasal dari Bahasa Yunani yang berarti “tulisan tersembunyi” (*covered writing*). Steganografi membutuhkan dua properti: wadah penampung dan data rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara, teks, dan video. Data rahasia yang disembunyikan juga dapat berupa citra, suara, teks, atau video (Munir, 2006).

Steganografi dapat dipandang sebagai kelanjutan kriptografi. Jika pada kriptografi, data yang telah disandikan (*ciphertext*) tetap tersedia, maka dengan steganografi cipherteks dapat disembunyikan sehingga pihak ketiga tidak mengetahui keberadaannya. Di negara-negara yang melakukan penyensoran informasi, steganografi sering digunakan untuk menyembunyikan pesan-pesan melalui gambar (*images*), video, atau suara (*audio*).

2.7.1 Sejarah Steganografi

Steganografi sudah dikenal oleh bangsa Yunani. Herodatus, penguasa Yunani, mengirim pesan rahasia dengan menggunakan kepala budak atau prajurit sebagai media. Dalam hal ini, rambut budak dibotaki, lalu pesan rahasia ditulis pada kulit kepala budak. Ketika rambut budak tumbuh, budak tersebut diutus untuk membawa pesan rahasia di balik rambutnya. Bangsa Romawi mengenal steganografi dengan menggunakan tinta tak-tampak (*invisible ink*) untuk menuliskan pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan di atas kertas dapat dibaca dengan cara memanaskan kertas tersebut (Munir, 2006).

2.7.2 Kriteria Steganografi yang Bagus

Steganografi yang dibahas di sini adalah penyembunyian data di dalam citra digital saja. Meskipun demikian, penyembunyian data dapat juga dilakukan pada wadah berupa suara digital, teks, ataupun video. Penyembunyian data rahasia ke dalam citra digital akan mengubah kualitas citra tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data adalah:

1. *Fidelity*

Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.

2. *Robustness*

Data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan (*cropping*), enkripsi, dan sebagainya). Bila

pada citra dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.

3. Recovery

Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut. (Munir, 2006)

2.7.3 Metode Steganografi

Kebanyakan algoritma steganografi menggunakan sebuah kombinasi dari bidang jenis teknik untuk melakukan sebuah tugas dalam penyelubungan pesan rahasia dalam sebuah selubung berkas. Sebuah program steganografi dibutuhkan untuk melakukan hal-hal berikut (baik implisit melalui suatu perkiraan maupun eksplisit melalui sebuah perhitungan), menemukan kelebihan *bits* dalam selubung *file* yang dapat digunakan untuk menyelubungi pesan rahasia didalamnya, memilih beberapa diantaranya untuk digunakan dalam menyelubungi data dan penyelubungan data dalam bits dipilih sebelumnya.

Ada empat jenis metode Steganografi, yaitu:

1. Least Significant Bit Insertion (LSB Insertion)

Metode yang digunakan untuk menyembunyikan pesan pada media digital tersebut berbeda-beda. Contohnya, pada berkas *image* pesan dapat disembunyikan dengan menggunakan cara menyisipkannya pada *bit* rendah atau *bit* yang paling kanan (LSB) pada data *pixel* yang menyusun *file* tersebut. Pada berkas bitmap 24 *bit*, setiap *pixel* (titik) pada gambar tersebut terdiri dari susunan tiga warna merah, hijau dan biru (RGB) yang masing-masing disusun oleh bilangan 8 *bit* (*byte*) dari 0 sampai 255 atau dengan format biner 00000000 sampai 11111111. Dengan demikian, pada setiap *pixel* berkas bitmap 24 *bit* kita dapat menyisipkan 3 *bit* data.

Kekurangan dari LSB *Insertion*: Dapat diambil kesimpulan dari contoh 8 *bit pixel*, menggunakan LSB *Insertion* dapat secara drastis mengubah unsur pokok warna dari *pixel*. Ini dapat menunjukkan perbedaan yang nyata dari *cover image* menjadi *stego image*, sehingga tanda tersebut menunjukkan keadaan dari steganografi. Variasi warna kurang jelas dengan 24 *bit image*, bagaimanapun *file*

tersebut sangatlah besar. Antara 8 bit dan 24 bit *image* mudah diserang dalam pemrosesan *image*, seperti *cropping* (kegagalan) dan *compression* (pemampatan).

Keuntungan dari LSB Insertion : Keuntungan yang paling besar dari algoritma LSB ini adalah cepat dan mudah. Dan juga algoritma tersebut memiliki *software* steganografi yang mendukung dengan bekerja di antara unsur pokok warna LSB melalui manipulasi *pallete* (lukisan).

2. Algorithms and Transformation

Algoritma *compression* adalah metode steganografi dengan menyembunyikan data dalam fungsi matematika. Dua fungsi tersebut adalah *Discrete Cosine Transformation* (DCT) dan *Wavelet Transformation*. Fungsi DCT dan *Wavelet* yaitu mentransformasi data dari satu tempat (*domain*) ke tempat (*domain*) yang lain. Fungsi DCT yaitu mentransformasi data dari tempat spatial (*spatial domain*) ke tempat frekuensi (*frequency domain*).

3. Redundant Pattern Encoding

Redundant Pattern Encoding adalah menggambar pesan kecil pada kebanyakan gambar. Keuntungan dari metode ini adalah dapat bertahan dari *cropping* (kegagalan). Kerugiannya yaitu tidak dapat menggambar pesan yang lebih besar.

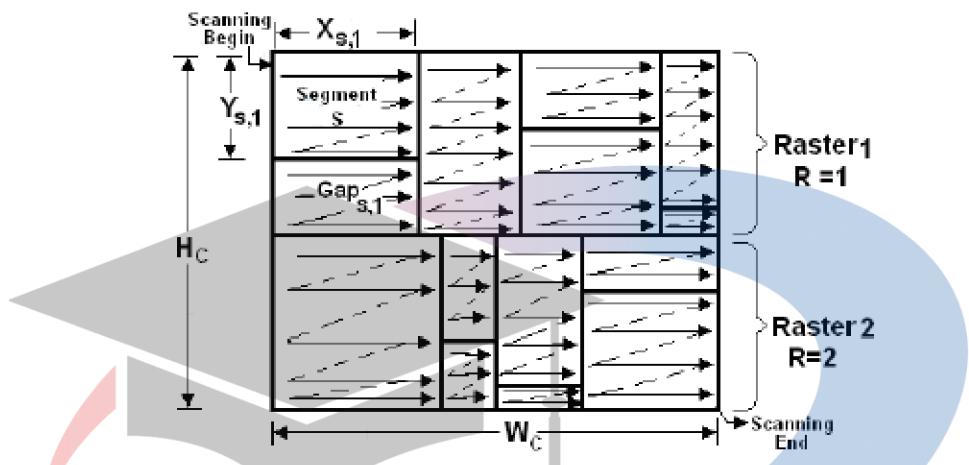
4. Spread Spectrum method

Spread Spectrum steganografi terpencar-pencar sebagai pesan yang diacak (*encrypted*) melalui gambar (tidak seperti dalam LSB). Untuk membaca suatu pesan, penerima memerlukan algoritma yaitu *crypto-key* dan *stego-key*. Metode ini juga masih mudah diserang yaitu penghancuran atau pengrusakan dari kompresi dan proses *image* (gambar) (Munir, 2006).

2.7.4 MLSB dengan Random Pixel Selection

Algoritma steganografi baru ini mengikuti sekumpulan aturan untuk menghasilkan sebuah citra *stego*. Algoritma steganografi (bagian pengirim) terdiri dari dua tahapan. Tujuan dari tahapan pertama adalah untuk menghasilkan *different size image segmentation* (DSIS) dari citra sampul untuk menyimpan data rahasia secara acak, sedangkan tahapan kedua bertujuan untuk membangun sebuah pendekatan efektif untuk menyisipkan data rahasia ke dalam citra sampul.

Segmentasi citra adalah proses yang digunakan untuk memecahkan citra sampul menjadi sekumpulan sub citra dengan berdasarkan pada sebuah hipotesis baru. Hipotesis ini dihasilkan berdasarkan pada kunci *cipher* dengan tiga buah operasi untuk membuat *attacker* sulit dalam mendeteksi sisi segmen. Proses segmentasi citra sampul dapat dilihat pada gambar 2.22 berikut:



Gambar 2.22 Segmentasi Citra Berdasarkan Algoritma DSIS

(Nameer, et. al, 2015)

Algoritma segmentasi DSIS dapat dijabarkan sebagai berikut:

Algorithm1: Image segmentation DSIS

Step1: Input K, Cover image C and \bar{A} ;

Step2: For each color in C; // color $\in \{R, G, B\}$.

Step3: For each raster R in the C;

Step 3-1: For each segment s in the raster R;

Step 3-1-1: Compute M_s ;

$$M_s = Val(K_s) \quad \forall s = 1, \dots, |SS| \quad (15)$$

Step 3-1-2: Compute F;

$$F = K - Val(St(M_s)^r) \quad (16)$$

dimana: $K = 300$

$(M_s)^r = \text{reverse dari } M_s$

Step 3-1-3: Compute A;

$$A = \left\lfloor \frac{F}{100} \right\rfloor \quad (17)$$

Step 3-1-4: Compute B;

$$B = \left\lfloor \frac{F - A \times 100}{10} \right\rfloor \quad (18)$$

Step 3-1-5: Compute C;

$$C = (F - (A \times 100) - (B \times 10)) \quad (19)$$

Step 3-1-6: Compute $X_{s,R}$;

$$X_{s,R} = \left\lceil \sqrt{\frac{W_c}{h}} \right\rceil + h \quad (20)$$

Step 3-1-7: Compute $Y_{s,R}$;

$$Y_{s,R} = \left\lceil \frac{\sqrt{H_c + A + B}}{\lambda} \right\rceil + \lambda \quad (21)$$

Step 3-1-8: Compute $\text{Gap}_{s,R}$;

$$\text{Gap}_{s,R} = X_{s,R} \times \left(\max_{\substack{\forall \text{ segment} \\ \text{index } (i) \text{ in } R}} (Y_{i,R}) - Y_{s,R} \right) \quad (22)$$

End; //foreach segment s.

End; //foreach raster R.

End.// End Algorithm1

UNIVERSITAS MIKROSKIL

Setelah dilakukan proses segmentasi menggunakan DSIS, selanjutnya akan dilakukan penyisipan yang diawali dengan perhitungan *byte value reduction* (BVR) yang berfungsi memperkecil intensitas *byte* dari 0 - 255 menjadi 0 – 16, dengan adanya fungsi ini maka klasifikasi dan perhitungan setiap byte menjadi lebih mudah dan cepat. Tahap berikutnya dilanjutkan dengan perhitungan nilai *number of bits to be hide* (NBTH) dengan rumus sebagai berikut :

$$\sigma^2 = \frac{\sum(x_i - \bar{x})^2}{n - 1} \quad (23)$$

Dimana Varian (σ^2) digunakan menentukan jumlah bit yang dapat disisipkan ke dalam setiap piksel

Algoritma penyisipan data pada citra sampul dapat dirincikan pada gambar 2.23 berikut:

Algorithm2: Embedding algorithm.

Step 1: Input Cover image (C), cipher key (K) and Secret image (S).

Step 2: Apply Comp(S, \hat{U}_m , \hat{U}_s , \check{I}_m) function to produce C_s ;

Step 3: Apply IEncry(C_s , ℓ_{C_s}) function to produce CE_s ;

Step4: For each color in C;

Step4-1: Apply $C_{DSIS}(C, K)$ function to produce non uniform segments Stg by calculating $X_{s,R}$ and $Y_{s,R}$;

Step4-2: Call Byte_Characteristic(.) to find NBTH;

Step4-3: Perform embedding function EM(C, SS, CE_s) of the secret image's compression and encryption (CE_s);

Step 5: Send Stg image to insecure channel;

End. //End Algorithm2

Sub-Algorithm2 // Set the intensity of each byte in the range (1, 16) and then find $NBTH_{i,j}$ for each byte at each color;

Byte_Characteristic(NBTH) {
 // scanning all bytes for all segment at each color in a cover image

 For each color in C

 For each segment Seg_s

 For each byte B

 Calculate BVR_B ;

 Calculate $NBTH_B$;

End. //for each segment set.

End. // for each segment set.

End. //for each color.

}// end sub-algorithm2.

Gambar 2.23 Algoritma Penyisipan Data pada Citra Sampul

(Nameer, et. al, 2015)

Sementara itu, algoritma ekstraksi dari citra stego dapat dirincikan sebagai berikut:

Algorithm3- Extraction algorithm
Step1: Input Stg image and the K that are received from the insecure channel;
Step2: For each color in Stg;
Step2-1: Applying $C_{DSIS}(Stg, K)$ to find the edges of each segment Seg _s by calculating X _{s,R} and Y _{s,R} ;
Step2-2: Scan all bytes in Stg image and calculate NBTH;
Step2-3: Apply EX function EX(Stg , SS) for all bytes depending on byte characteristics to produce CE _s ;
Step3: Apply IDcry(CE _s × ℓ _{C_s}) function to produce C _s ;
Step4: Apply DEC(C _s) function to find a secret image S;
End. // End Algorithm3

Gambar 2.24 Algoritma Ekstraksi Data Rahasia

(Nameer, et. al, 2015)

2.8 Penilaian Kualitas Citra

Metode yang paling sederhana dan banyak digunakan dalam mengukur kualitas citra adalah *Mean Squared Error* (MSE), dihitung dengan rata-rata perbedaan intensitas kuadrat piksel gambar terdistorsi dan referensi, dengan kuantitas terkait *Peak Signal to Noise Ratio* (PSNR). Metriks tersebut menarik karena sederhana untuk menghitung, memiliki arti fisik yang jelas dan matematis dalam konteks optimasi. Tapi kedua metriks tersebut sangat tidak cocok untuk dilihat kualitasnya secara visual (Wang, Bovik, Fellow, Sheikh, & Simoncelli, 2004).

2.8.1 Mean Square Error

Pengukuran yang paling sederhana dalam pengukuran kualitas gambar adalah *Mean Square Error* (MSE). Nilai besar MSE berarti bahwa gambar berkualitas buruk. MSE didefinisikan sebagai berikut (Youssif, Darwish, & Madbouly, 2010):

$$\text{MSE} = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N (x(m, n) - \hat{x}(m, n))^2 . \quad (24)$$

Keterangan :

MSE : nilai *Mean Square Error*

f(x,y) : intensitas citra asli

$f(x,y)$: intensitas citra hasil filter

2.8.2 Peak Signal to Noise Ratio (PSNR)

Sebuah gambar berkualitas tinggi memiliki nilai kecil *Peak Signal to Noise Ratio* (PSNR). PSNR didefinisikan sebagai berikut (Youssif, Darwish, & Madbouly, 2010):

$$\text{PSNR} = \left[10 \log \frac{255^2}{MSE} \right]. \quad (25)$$

Keterangan :

PSNR: nilai *Peak Signal to Noise Ratio*

MSE : nilai *Mean Squared Error*

255 : nilai skala keabuan citra

UNIVERSITAS
MIKROSKIL