

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Media Sosial

Media sosial mengacu pada berbagai platform yang tersedia di internet, yang memberikan kesempatan kepada penggunanya untuk membuat profil mereka dan berbagi serta mempromosikan konten. Platform media sosial ini semua dirancang untuk membantu orang dan perusahaan membangun kehadiran sosial dan membuat orang lain tahu tentang produk dan layanan mereka.

Popularitas media sosial, sebagai alat pemasaran, meningkat dalam beberapa dekade terakhir ketika semakin banyak perusahaan menyadari potensi media sosial yang sebenarnya dan mulai menggunakan untuk keuntungan mereka. Mereka mengerti bahwa mungkin bagi mereka untuk menjangkau jutaan orang, di seluruh dunia, dan meningkatkan pelanggan mereka hingga beberapa kali lipat. Media sosial sekarang menjadi bagian dari strategi pemasaran setiap perusahaan. Mulai dari toko kecil di Jepang hingga perusahaan bernilai jutaan dolar di AS, semua orang menggunakan kekuatan internet untuk mendapat perhatian dan meningkatkan penjualan produk mereka (Kennedy, 2016).

Van Dijk dalam Nasrullah (2015) menyatakan bahwa media sosial adalah platform media yang memfokuskan pada eksistensi pengguna yang memfasilitasi mereka dalam beraktivitas maupun berkolaborasi. Karena itu media sosial dapat dilihat sebagai medium (fasilitator) *online* yang menguatkan hubungan antar pengguna sekaligus sebuah ikatan sosial.

Meike dan Young dalam Nasrullah (2015) mengartikan kata media sosial sebagai konvergensi antara komunikasi personal dalam arti saling berbagi diantara individu (*to be share one-to-one*) dan media publik untuk berbagi kepada siapa saja tanpa ada kekhususan individu.

Menurut Boyd dalam Nasrullah (2015) media sosial sebagai kumpulan perangkat lunak yang memungkinkan individu maupun komunitas untuk berkumpul, berbagi, berkomunikasi, dan dalam kasus tertentu saling berkolaborasi atau bermain. Media sosial memiliki kekuatan pada *user-generated content* (UGC) dimana konten dihasilkan oleh pengguna, bukan oleh editor sebagaimana di instansi media massa (Setiadi, 2016).

Media Sosial (*Social media*) adalah media *online* yang mendukung interaksi sosial. Media sosial menggunakan teknologi berbasis web yang mengubah komunikasi menjadi dialog interaktif. Beberapa situs media sosial yang populer sekarang ini antara lain : Whatsapp, BBM, Facebook, Youtube, Twitter, Wikipedia, Blog, dan lain-lain. Definisi lain dari media sosial juga dijelaskan oleh Antony Mayfield (2008). Menurutnya media sosial adalah media dimana

penggunanya dengan mudah berpartisipasi di dalamnya, berbagi dan menciptakan pesan, termasuk *blog*, jejaring sosial, wiki/ensiklopedia *online*, forum-forum maya, termasuk *virtual worlds* (dengan *avatar* dan karakter 3D) (Doni, 2017).



Gambar 2.1 Logo Media Sosial

Sumber: Doni, 2017

Pada intinya, dengan sosial media dapat dilakukan berbagai aktivitas dua arah dalam berbagai bentuk pertukaran, kolaborasi, dan saling berkenalan dalam bentuk tulisan, visual maupun audiovisual. Media sosial diawali dari tiga hal, yaitu *Sharing*, *Collaborating* dan *Connecting* (Setiadi, 2016).

Media sosial memiliki beberapa fungsi sebagai berikut :

1. Media sosial adalah media yang didesain untuk memperluas interaksi sosial manusia menggunakan internet dan teknologi web.
2. Media sosial berhasil mentransformasi praktik komunikasi searah media siaran dari satu institusi media ke banyak *audience* (“one to many”) menjadi praktik komunikasi dialogis antar banyak *audience* (“many to many”).
3. Media sosial mendukung demokratisasi pengetahuan dan informasi. Mentransformasi manusia dari pengguna isi pesan menjadi pembuat pesan itu sendiri. (Doni, 2017)

Karakteristik media sosial tidak jauh berbeda dengan media *cyber* dikarenakan media sosial merupakan salah satu platform dari media *cyber*. Namun demikian, menurut Nasrullah (2015) media sosial memiliki karakter khusus, yaitu:

1. Jaringan (*Network*)

Jaringan adalah infrastruktur yang menghubungkan antara komputer dengan perangkat keras lainnya. Koneksi ini diperlukan karena komunikasi bisa terjadi jika antar komputer terhubung, termasuk di dalamnya perpindahan data.

2. Informasi (*Informations*)

Informasi menjadi entitas penting di media sosial karena pengguna media sosial mengkreasikan representasi identitasnya, memproduksi konten, dan melakukan interaksi berdasarkan informasi.

### 3. Arsip (*Archive*)

Bagi pengguna media sosial, arsip menjadi sebuah karakter yang menjelaskan bahwa informasi telah tersimpan dan bisa diakses kapanpun dan melalui perangkat apapun.

### 4. Interaksi (*Interactivity*)

Media sosial membentuk jaringan antar pengguna yang tidak hanya sekedar memperluas hubungan pertemanan atau pengikut (*follower*) semata, tetapi harus dibangun dengan interaksi antar pengguna tersebut.

### 5. Simulasi Sosial (*simulation of society*)

Media sosial memiliki karakter sebagai medium berlangsungnya masyarakat (*society*) di dunia virtual. Media sosial memiliki keunikan dan pola yang dalam banyak kasus berbeda dan tidak dijumpai dalam tatanan masyarakat yang nyata.

### 6. Konten oleh pengguna (*user-generated content*)

Di Media sosial konten sepenuhnya milik dan berdasarkan kontribusi pengguna atau pemilik akun. UGC merupakan relasi simbiosis dalam budaya media baru yang memberikan kesempatan dan keleluasaan pengguna untuk berpartisipasi. Hal ini berbeda dengan media lama (tradisional) dimana khalayaknya sebatas menjadi objek atau sasaran yang pasif dalam distribusi pesan. (Setiadi, 2016)

Menurut Jogiyanto (2007) Perilaku adalah tindakan atau kegiatan nyata yang dilakukan karena individual mempunyai keinginan untuk melakukan sesuatu tertentu. Minat perilaku akan menentukan perilakunya. Perilaku-perilaku yang di inginkan adalah perilaku-perilaku yang kejadiannya merupakan suatu hasil langsung dari usaha-usaha di bawah sadar yang dibuat oleh seseorang individual. Perilaku adalah tindakan yang dilakukan oleh seseorang. Dalam konteks penggunaan teknologi informasi, perilaku adalah penggunaan sesungguhnya dari teknologi (Doni, 2017).

Menurut Nasullah (2015) setidaknya ada enam kategori besar untuk melihat pembagian media sosial, yakni:

#### 1. Media Jejaring Sosial (*Social networking*)

Media jejaring sosial merupakan medium yang paling popular. Media ini merupakan sarana yang bias digunakan pengguna untuk melakukan hubungan sosial, termasuk konsekuensi atau efek dari hubungan sosial tersebut di dunia virtual. Karakter utama dari situs jejaring sosial adalah setiap pengguna membentuk jaringan pertemanan, baik terhadap pengguna yang sudah diketahuinya dan kemungkinan saling bertemu di dunia nyata (*offline*) maupun membentuk jaringan pertemanan baru. Contoh jejaring sosial yang banyak digunakan adalah Facebook dan LinkedIn.

## 2. Jurnal *online* (*blog*)

*Blog* merupakan media sosial yang memungkinkan penggunanya untuk mengunggah aktivitas keseharian, saling mengomentari dan berbagi, baik tautan *web* lain, informasi dan sebagainya. Pada awalnya *blog* merupakan suatu bentuk situs pribadi yang berisi kumpulan tautan ke situs lain yang dianggap menarik dan diperbarui setiap harinya. Pada perkembangan selanjutnya, *blog* banyak jurnal (tulisan keseharian pribadi) pemilik media dan terdapat kolom komentar yang bisa diisi oleh pengguna. Secara mekanis, jenis media sosial ini bias dibagi menjadi dua, yaitu kategori personal *homepage*, yaitu pemilik menggunakan nama domain sendiri seperti .com atau .net dan yang kedua dengan menggunakan fasilitas penyedia halaman *web blog* gratis, seperti wordpress atau blogspot.

## 3. Jurnal *online* sederhana atau *microblog* (*micro-blogging*)

Tidak berbeda dengan jurnal *online* (*blog*), *microblogging* merupakan jenis media sosial yang memfasilitasi pengguna untuk menulis dan mempublikasikan aktivitas serta atau pendapatnya. Contoh *microblogging* yang paling banyak digunakan adalah Twitter.

## 4. Media berbagi (*media sharing*)

Situs berbagi media merupakan jenis media sosial yang memfasilitasi penggunanya untuk berbagi media, mulai dari dokumen (*file*), video, audio, gambar, dan sebagainya. Contoh media ini adalah: Youtube, Flickr, Photo-bucket, atau snapfish.

## 5. Penanda sosial (*social bookmarking*)

Penanda sosial merupakan media sosial yang bekerja untuk mengorganisasi, menyimpan, mengelola, dan mencari informasi atau berita tertentu secara *online*. Beberapa situs sosial *bookmarking* yang popular adalah delicious.com, stumbleUpon.com, Digg.com, Reddit.com, dan untuk di Indonesia ada LintasMe.

## 6. Media konten bersama atau wiki.

Media sosial ini merupakan situs yang kontennya hasil kolaborasi dari para penggunanya. Mirip dengan kamus atau ensiklopedi, wiki menghadirkan kepada pengguna pengertian, sejarah hingga rujukan buku atau tautan tentang satu kata. Dalam praktiknya, penjelasan-penjelasan tersebut dikerjakan oleh pengunjung, artinya ada kolaborasi atau kerja sama dari semua pengunjung untuk mengisi konten dalam situs ini. (Setiadi, 2016)

## 2.2 Citra

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

Citra digital adalah citra yang dapat diolah oleh komputer. Pada gambar 2.1, sebuah citra *grayscale* berukuran 150 x 150 piksel (elemen terkecil dari sebuah citra) diambil sebagian (kotak kecil) berukuran 9 x 9 piksel. Maka, monitor akan menampilkan sebuah kotak kecil. Namun, yang disimpan dalam memori komputer hanyalah angka-angka yang menunjukkan besar intensitas pada masing-masing piksel tersebut.

Citra analog adalah citra yang bersifat kontinu, seperti gambar pada monitor televisi, foto sinar-X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasil CT *scan*, gambar-gambar yang terekam pada pita kaset dan lain sebagainya. Citra analog tidak dapat direpresentasikan dalam komputer sehingga tidak bisa diproses di komputer secara langsung. Oleh sebab itu, agar citra ini dapat diproses di komputer, proses konversi analog ke digital harus dilakukan terlebih dahulu. Citra analog dihasilkan dari alat-alat analog, seperti video kamera analog, kamera foto analog, *WebCam*, CT *scan*, sensor rontgen untuk foto *thorax*, sensor gelombang pendek pada sistem radar, sensor *ultrasound* pada sistem USG, dan lain-lain (Sutoyo, et al., 2009).

### 2.2.1 Representasi Citra Digital

Sebuah citra digital dapat diwakili oleh sebuah matriks yang terdiri dari M kolom dan N baris, di mana perpotongan antara kolom dan baris disebut piksel (*pixel = picture element*), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x, y) adalah  $f(x, y)$ , yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu, sebuah citra digital dapat dituliskan dalam bentuk matriks berikut:

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, M-1) \\ f(1, 0) & \dots & \dots & f(1, M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1, 0) & f(N-1, 1) & \dots & f(N-1, M-1) \end{bmatrix} \quad (1)$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas  $f(x, y)$ , di mana harga x (baris) dan y (kolom) merupakan koordinat posisi dan  $f(x, y)$  adalah nilai fungsi pada setiap titik (x, y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (*sampling* dan

kuantisasi) diperoleh besar baris M dan kolom N hingga citra membentuk matriks  $M \times N$  dan jumlah tingkat keabuan piksel G. Biasanya besar M, N dan G adalah perpangkatan dari dua,

$$M = 2^m \quad N = 2^n \quad \text{dan} \quad G = 2^k$$

yang dalam hal ini m, n dan k adalah bilangan bulat positif. Jika b menyatakan jumlah bit yang diperlukan untuk menyimpan citra digital dalam memori, maka:

$$b = M \times N \times k \quad (2)$$

(Sutoyo, et al., 2009)

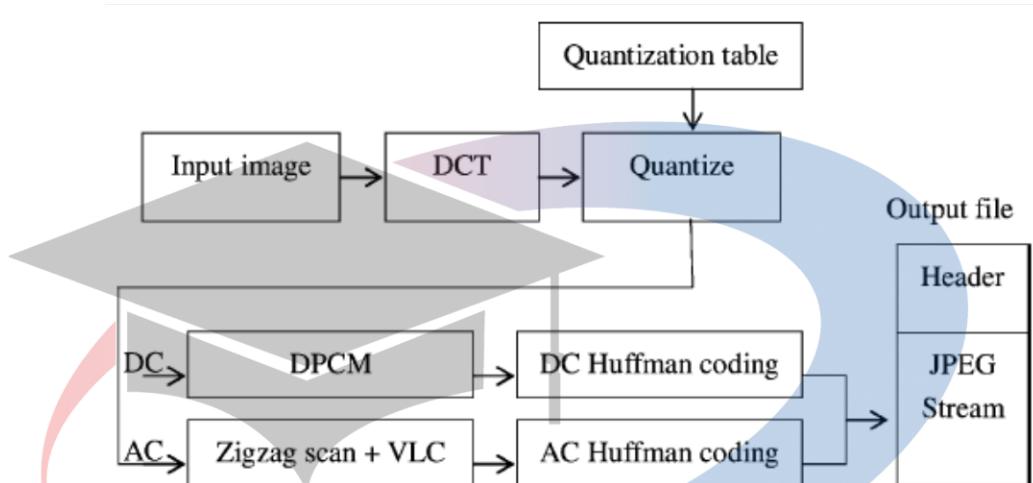
### 2.2.2 YCbCr Color Space

*Channel* yang dihasilkan dari 3 sensor, dikombinasi dari beberapa *frame* pada suatu sensor atau yang dihasilkan dari demonstrasi hasil akhirnya biasanya dinyatakan dalam ruang warna *Red-Green-Blue* (RGB). Dalam ruang warna RGB setiap piksel direpresentasikan sebagai kombinasi dari warna merah, hijau, dan biru. Meskipun ini tidak memuaskan secara intuitif, tidak ideal untuk sebagian besar jenis pemrosesan karena perubahan pada *channel* individual akan mengubah rasio warna, artinya semua *channel* perlu dimodifikasi secara merata untuk menjaga informasi warna. Daripada bekerja dengan tiga *channel* warna, RGB dapat diubah menjadi format YCbCr yang terbagi menjadi *channel luminance* dan dua *channel chrominance*. *Channel* pencahayaan Y, berisi informasi intensitas cahaya dan dapat digunakan sebagai representasi *grayscale* dari gambar. *Chrominance channel* yaitu Cb dan Cr adalah perbedaan komponen biru dan merah secara masing-masing. Konversi standar untuk koreksi *gamma* RGB ke YCbCr adalah:

Komponen *chrominance* dari gambar YCbCr sering disub-sampel untuk mengurangi jumlah data yang sedang diproses. Bentuk *sampling* YCbCr yang umumnya digunakan termasuk YCbCr 4:4:4 yang tidak melakukan pengambilan sub-sampel, YCbCr yang di sub-sampel dalam *chrominance* hanya berisi satu setengah dari resolusi dalam arah horizontal dan resolusi penuh dalam arah vertikal, YCbCr 4: 2: 0 yang mensub-sampel saluran *chrominance* hanya mengandung satu setengah resolusi di keduanya arah horizontal dan vertikal, dan YCbCr 4: 0: 0 yang hanya berisi *channel luminance*. Meskipun *chrominance sub-sampling* secara signifikan mengurangi resolusi warna, sistem visual manusia memiliki sensitivitas yang lebih rendah terhadap informasi warna frekuensi tinggi dibandingkan dengan intensitas informasi frekuensi tinggi dan oleh karena itu resolusi yang berbeda dapat diterima.

### 2.2.3 Standar Kompresi JPEG

JPEG merupakan sebuah standar kompresi citra diam, yang distandarisasi oleh *International Organization for Standardization and International Telecommunication Unit*. Karena rasio kompresi dan kualitas citranya yang bagus, metode ini dianggap sebagai standar kompresi citra. Langkah proses utama dari JPEG mencakup transformasi DCT, kuantisasi dan *entropy coding*. Gambar 2.4 berikut menunjukkan sebuah diagram blok sederhana dari standar kompresi JPEG.



Gambar 2.2 Gambaran Umum dari Diagram Blok Kompresi JPEG untuk Citra *Grayscale*

Sumber : Chin-Chen Chang, Jun-Chou Chuang dan Yih-Shin Hu, 2004

Asumsikan bahwa input berupa sebuah citra *grayscale*. Pertama, citra dipecahkan menjadi banyak blok yang tidak saling bertindihan dengan ukuran  $8 \times 8$ . Untuk setiap blok, bilangan 128 dikurangkan dari setiap nilai piksel. Kemudian, setiap blok ditransformasikan dengan DCT. Blok transformasi disebut sebagai blok DCT. Setelah langkah ini, setiap blok akan memiliki koefisien DCT. Koefisien pada lokasi  $(0, 0)$  pada setiap blok disebut sebagai Koefisien DC dan koefisien lainnya disebut sebagai Koefisien AC. Kemudian, sebuah tabel kuantisasi berat (*weighted quantization table*) yang disebut sebagai tabel Q, digunakan untuk mengkuantisasi koefisien DCT dari setiap blok.

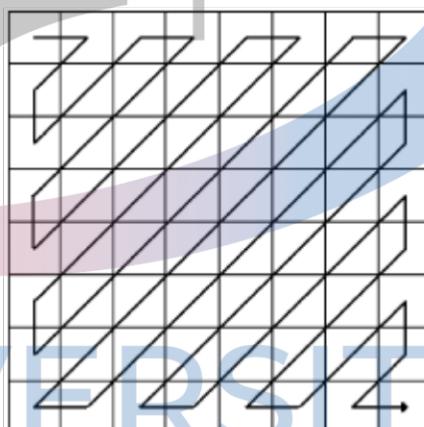
DCT  $G(u, v)$  dari sebuah *macroblock*  $B(i, j)$  dengan ukuran  $M \times M$  dapat ditentukan sebagai berikut:

$$G(u, v) = \alpha(u)\alpha(v) \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} B(i, j) \cos\left[\frac{(2i+1)u\pi}{2M}\right] \cos\left[\frac{(2j+1)v\pi}{2M}\right] \quad (3)$$

(4)

dimana  $i, j, u, v = 0, 1, \dots, M - 1$ . Citra transformasi DCT *macroblock*  $F(u, v)$  dari citra input  $P(i, j)$  dievaluasi dan kemudian dipecahkan menjadi blok tidak bertindihan dari koefisien DCT.

Berikutnya adalah prosedur untuk *entropy encoding*. Prosedur ini mencakup dua bagian. Pertama, untuk DC, koefisien DC kuantisasi setelah proses *differential pulse code modulation* (DPCM) akan *di-encode* menggunakan DC *Huffman coding*. DPCM digunakan untuk menghitung perbedaan antara dua koefisien DC kuantisasi pada blok DCT sekarang dan sebelumnya. Kedua, untuk AC, *array* dua dimensi dari koefisien AC harus ditransformasikan menjadi sebuah *array* satu dimensi menggunakan *zigzag scan*, seperti terlihat pada gambar 2.5.



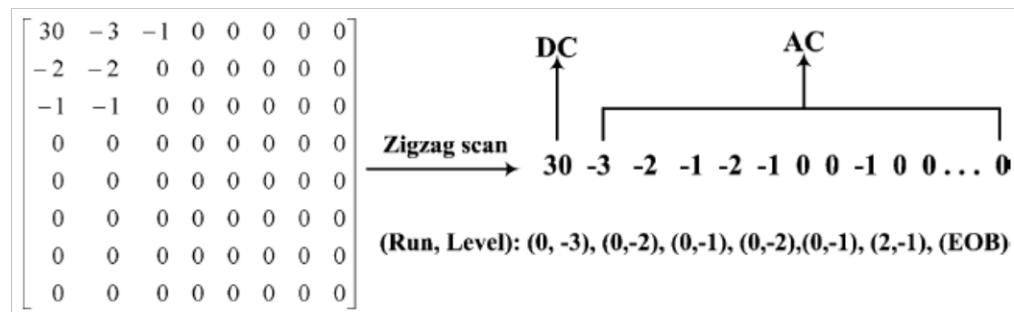
Gambar 2.3 Urutan Zigzag Scan

Sumber : Chin-Chen Chang, Jun-Chou Chuang dan Yih-Shin Hu, 2004

Koefisien AC kuantisasi pada *array* dimensi satu akan dikonversi menjadi bentuk (Run, Level) dengan menggunakan *variable length coding* (VLC). Run menunjukkan jumlah dari nol sebelum sebuah elemen *non-zero* pada *array* satu dimensi dan Level menunjukkan nilai dari sebuah elemen *non-zero*.

Gambar 2.6 adalah sebuah contoh dari *variable length coding*. Kemudian, hasil ini dimasukkan ke AC *Huffman coding*. Terakhir, *output* nya berupa DC dan AC *Huffman code* sebagai sebuah JPEG *stream*. Kemudian, sebuah *head* dikombinasikan dengan *stream* ini untuk membentuk sebuah *file* JPEG terkompresi. Hal yang sama juga berlaku untuk prosedur

*decoding* JPEG, yang merupakan kebalikan dari prosedur *encoding*. (Chang, et al., 2004)



Gambar 2.4 Sebuah Contoh dari *Variable Length Coding* (VLC)

Sumber : Chin-Chen Chang, Jun-Chou Chuang dan Yih-Shin Hu, 2004

Berikut adalah algoritme *discrete cosine transform* (DCT):

1. Gambar dibagi menjadi beberapa blok, dan masing-masing blok memiliki 8 piksel x 8 piksel.
  2. Nilai 128 mengurangi data Matriks *Original* karena algoritme DCT bekerja pada rentang -128 sampai 127 sesuai dengan ketentuan pengolahan citra digital.
  3. Dengan menggunakan persamaan *Discrete Cosine Transform*, cari matriks D dimana matriks D akan digunakan untuk kuantisasi lanjutan.

$$D = T \cdot M \cdot T^t \quad (5)$$

Dimana :

Matriks T =

	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
	0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
	0.4619	0.1919	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
T =	0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
	0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
	-0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	-0.4904	-0.2778
	0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
	0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2778	-0.0975

Matriks transpose dari  $T$  ( $T^t$ ) =

$$T^t = \begin{pmatrix} 0.3536 & 0.4904 & 0.4619 & 0.4157 & 0.3536 & 0.2778 & 0.1913 & 0.0975 \\ 0.3536 & 0.4157 & 0.1919 & -0.0975 & -0.3536 & -0.4904 & -0.4619 & -0.2778 \\ 0.3536 & 0.2778 & -0.1913 & -0.4904 & -0.3536 & 0.0975 & 0.4619 & 0.4157 \\ 0.3536 & 0.0975 & -0.4619 & -0.2778 & 0.3536 & 0.4157 & -0.1912 & -0.4904 \\ 0.3536 & -0.0975 & -0.4619 & 0.2778 & 0.3536 & -0.4157 & -0.1913 & 0.4904 \\ 0.3536 & -0.2778 & -0.1913 & 0.4904 & -0.3536 & -0.0975 & 0.4619 & -0.4157 \\ 0.3536 & -0.4157 & 0.1913 & 0.0975 & -0.3536 & -0.4904 & -0.4619 & 0.2778 \\ 0.3536 & -0.4904 & 0.4619 & -0.4157 & 0.3536 & -0.2778 & 0.1913 & -0.0975 \end{pmatrix}$$

Matriks M = matriks yang berisi nilai piksel dari citra input.

4. Matriks D sekarang berisi dengan koefisien DCT, dimana data yang terletak pada kiri atas merupakan korelasi dari frekuensi - frekuensi rendah dari data *original*. Sedangkan yang terletak pada kanan bawah merupakan korelasi dari frekuensi – frekuensi tinggi dari data *original*. Setelah itu lakukan proses kuantisasi dengan Quality level 50.

$$Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Persamaan matriks kuantisasi adalah sebagai berikut, dimana *round* berarti mendekatkan nilai hasil pembagian ke pembulatan bilangan integer terdekat.

$$C_{ij} = \text{round} \frac{D_{ij}}{Q_{ij}} \quad (6)$$

5. Susun bilangan menggunakan fungsi *zig-zag scanning* dimana ini merupakan langkah terakhir pada proses kompresi.

### 2.3 Fungsi Hash

Sebuah nilai *hash* dihasilkan oleh fungsi H dengan bentuk :

$$h = H(M) \quad (7)$$

dimana M adalah pesan dengan panjang bebas, dan H(M) atau h adalah nilai *hash* dengan panjang tetap. Nilai *hash* ini akan ditambahkan di awal pesan dan kemudian dikirimkan bersamaan. Penerima dari pesan itu akan melakukan autentikasi terhadap pesan itu dengan

mengkomputasi nilai *hash* dan kemudian membandingkannya dengan nilai *hash* yang ada di awal pesan.

Agar fungsi *hash* bisa bertindak sebagai autentikasi pesan, fungsi *hash* H harus memenuhi syarat – syarat di bawah ini (Stallings, 2011):

1. H dapat diterapkan pada blok data dengan ukuran berapa pun.
2. H menghasilkan keluaran dengan panjang yang tetap.
3. Relatif mudah untuk mengkomputasi  $H(x)$  untuk sembarang  $x$ .
4. Dengan mengetahui nilai *hash*  $y$ , tidaklah bisa secara komputasi untuk menemukan  $x$  yang memenuhi  $H(x) = y$ .
5. Dengan mengetahui  $x$ , tidaklah bisa secara komputasi untuk menemukan  $y \neq x$  dengan  $H(y) = H(x)$ .
6. Tidak bisa secara komputasi untuk menemukan pasangan  $(x,y)$  yang memenuhi  $H(x) = H(y)$ .

Tiga syarat pertama berfungsi sebagai syarat penerapan praktis dari fungsi *hash* untuk autentikasi pesan.

Syarat keempat adalah syarat “**satu-arah**” (*one-way*). Mudah untuk menghasilkan kode nilai *hash* dengan diberikan sebuah pesan tetapi tidak bisa untuk menghasilkan pesan dengan diberikan kode nilai *hash*.

Syarat kelima menjamin bahwa dengan diberikan pesan asli tidak akan ditemukan suatu pesan lain dengan nilai *hash* yang sama dengan nilai *hash* pesan aslinya. Syarat ini adalah untuk mencegah penipuan jika digunakan kode *hash* yang terenkripsi. Untuk kasus ini, pihak lawan bisa membaca pesan dan juga menghasilkan kode *hash*-nya. Tetapi karena pihak lawan tersebut tidak memiliki kunci untuk mendekripsi kode *hash* tersebut, maka ia tidak akan bisa mengubah pesan tanpa diketahui. Jika syarat ini tidak dipenuhi maka pihak lawan bisa melakukan hal – hal berikut.

1. Mengintersepsi suatu pesan dengan kode *hash* terenkripsinya.
2. Menghasilkan kode *hash* yang tidak terenkripsi dari pesan tersebut.
3. Membuat pesan baru yang lain dengan kode nilai *hash* yang sama dan kemudian mengirimkannya kembali.

Jika kelima syarat di atas dipenuhi, maka fungsi *hash* tersebut dinamakan fungsi *hash* yang lemah. Jika syarat keenam juga dipenuhi, maka fungsi *hash* itu dianggap fungsi *hash* yang kuat.

Selain untuk autentikasi pesan, fungsi *hash* juga dapat diterapkan untuk menghasilkan kunci berbasis *passphrase*. Cara kerjanya adalah sebagai berikut :

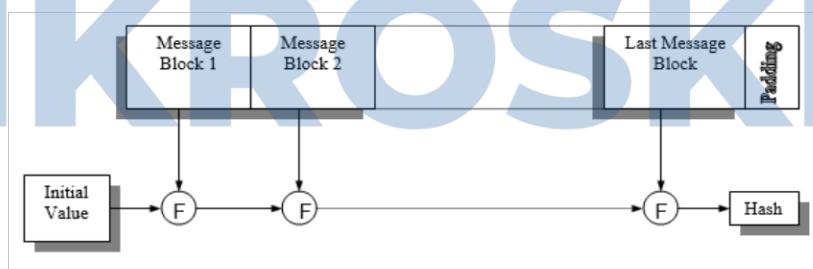
1. Dengan menggunakan fungsi *hash* dihasilkan nilai *hash* dari *passphrase*.

2. Nilai *hash* dari *passphrase* dijadikan sebagai kunci untuk melakukan enkripsi / dekripsi.

Nilai fungsi *hash* merepresentasikan pesan yang lebih pendek dari dokumen dari mana nilai tersebut dihitung, nilai ini sering disebut *message digest*. *Message digest* dapat dianggap sebagai satu *"digital fingerprint"* dari dokumen yang lebih panjang. Contoh fungsi *hash* yang terkenal adalah MD2, MD5, SHA dan HAVAL.

Peranan fungsi *hash* dalam kriptografi adalah dalam hal pengecekan kondisi terhadap integritas pesan dan tanda tangan digital. Suatu *digest* dapat dibuat publik tanpa menunjukkan isi dokumen dari mana *digest* tersebut diturunkan. Hal ini sangat penting dalam *digital timestamping* dimana dengan menggunakan fungsi *hash*, seseorang dapat memperoleh dokumen dengan stempel waktu (*document timestamped*) tanpa menunjukkan isi dari dokumen kepada penyedia layanan *timestamping*.

Damgard dan Merkle memberikan kontribusi yang signifikan dalam hal perancangan fungsi hash dalam kriptografi dengan mendefinisikan suatu fungsi *hash* dalam istilah khusus yang disebut *compression function*. Suatu fungsi kompresi menggunakan *input string* dengan panjang tertentu dan menghasilkan *string* yang lebih pendek tetapi dengan panjang tertentu juga. Berikan satu fungsi kompresi, satu fungsi *hash* dapat didefinisikan dengan penggunaan yang berulang dari fungsi kompresi hingga keseluruhan pesan selesai diproses. Dalam proses ini, sebuah pesan dengan panjang sembarang dipecah ke dalam beberapa blok dengan panjang tergantung pada fungsi kompresi dan *"padded"* (untuk alasan keamanan) sehingga ukuran pesan adalah perkalian dari ukuran blok. Blok-blok tersebut kemudian diproses secara sekuisial, dengan mengambil hasil dari *hash* sejauh ini sebagai *input* dan blok pesan saat ini, dengan hasil terakhir adalah nilai *hash* untuk pesan dimaksud.



Gambar 2.5 Struktur fungsi *hash* iteratif Damgard / Merkle; F = fungsi kompresi

Sebuah fungsi *hash* (*hash function* atau *hash algorithm*) adalah suatu cara untuk menghasilkan sebuah *digital "fingerprint"* kecil dari sembarang data. Fungsi ini memecahkan dan mencampurkan data untuk menghasilkan *fingerprint* yang sering disebut sebagai nilai *hash* (*hash value*). Nilai *hash* ini sering direpresentasikan dengan sebuah *string* pendek dari huruf-huruf dan angka-angka yang kelihatan acak (berbentuk heksadesimal). Sebuah fungsi *hash* yang

baik adalah suatu fungsi yang tidak (jarang) memiliki *output* nilai *hash* yang sama untuk *input* yang berbeda.

*Secure Hash Algorithm*, SHA-1 ini dikembangkan oleh NIST (*National Institute of Standard and Technology*). SHA-1 dapat diterapkan dalam penggunaan *Digital Signature Algorithm* (DSA) yang dispesifikasikan dalam *Digital Signature Standard* (DSS) dan SHA tersebut dapat diterapkan untuk aplikasi federal.

Untuk suatu pesan yang panjangnya  $< 2^{64}$ , SHA-1 akan menghasilkan keluaran sebanyak 160 bit dari pesan tersebut dan pesan keluaran itu disebut *message digest*. Panjang jarak *message digest* dapat berkisar antara 160 sampai 512 bit tergantung algoritmenya. Berdasarkan cirinya SHA-1 dapat digunakan dengan algoritme kriptografi lainnya seperti *Digital Signature Algorithms* atau dalam generasi angka yang acak (*bits*).

SHA-1 dikatakan aman karena proses SHA-1 dihitung secara *invisible* untuk mencari pesan yang sesuai untuk menghasilkan *message digest* atau dapat juga digunakan untuk mencari dua pesan yang berbeda yang akan menghasilkan *message digest* yang sama.

Menurut jenisnya SHA dapat dispesifikasikan menjadi 4 bagian yaitu: SHA-1, SHA-256, SHA-384, dan SHA-512. Berikut ini merupakan daftar-daftar properti dari keempat SHA.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security <sup>2</sup> (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

Gambar 2.6 Daftar-daftar properti dari keempat SHA

Untuk SHA-1 ukuran blok pesan -m bit- dapat ditentukan tergantung dari algoritmenya. Pada SHA-1 masing-masing blok pesan mempunyai 512 bit dimana dapat dilakukan dengan 16 urutan sebesar 32 bit.

SHA-1 digunakan untuk menghitung *message digest* pada pesan atau *file* data yang diberikan sebagai *input*. Tujuan pengisian pesan adalah untuk menghasilkan total dari pesan yang diisi menjadi perkalian dari 512 bits.

Beberapa hal yang dilakukan dalam pengisian pesan :

- Panjang dari pesan, M adalah k bits dimana panjang  $k < 2^{64}$ . Tambahkan bit “1” pada akhir pesan. Misalkan pesan yang asli adalah “01010000” maka setelah diisi menjadi “010100001”.

- b. Tambahkan bit “0”, angka bit “0” tergantung dari panjang pesan. Misalnya : Pesan asli yang merupakan bit *string* : abcde 01100001 01100010 01100011 01100100 01100101.

Setelah langkah (a) dilakukan

01100001 01100010 01100011 01100100 01100101 1.

Panjang  $k = 40$  dan angka bit di atas adalah 41 dan 407 ditambah bit “0” ( $448 - (40+1) = 407$ ). Kemudian diubah dalam *hex*:

61626364	65800000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000		

- c. Untuk memperoleh 2 kata dari  $k$ , angka bit dalam pesan asli yaitu jika  $k < 2^{32}$  maka kata pertama adalah semua bit ”0”. Maka gambaran dari 2 kata dari  $k = 40$  dalam *hex* adalah 00000000 00000028.

61626364	65800000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000028

SHA-1 menggunakan urutan fungsi logika yang dilambangkan dengan  $f_0, f_1, \dots, f_{79}$ .

Untuk masing-masing  $f_t$ , dimana  $0 \leq t < 79$  akan menghasilkan *output* sebanyak 32 bit.

Fungsinya adalah sebagai berikut:

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \leq t \leq 19 \\ B \oplus C \oplus D & 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leq t \leq 59 \\ B \oplus C \oplus D & 60 \leq t \leq 79 \end{cases}$$

$\oplus = \text{fungsi XOR}$

Konstanta kata yang digunakan pada SHA-1 yang disimbolkan secara berurutan dari  $K(0), K(1), \dots, K(79)$  dalam bentuk heksa adalah sebagai berikut :

$$K_t = \begin{cases} 5A827999 & 0 \leq t \leq 19 \\ 6ED9EBA1 & 20 \leq t \leq 39 \\ 8F1BBCDC & 40 \leq t \leq 59 \\ CA62C1D6 & 60 \leq t \leq 79 \end{cases}$$

Algoritme SHA-1 dapat diringkas sebagai berikut:

- Penghitungan menggunakan dua *buffer* dimana masing-masing *buffer* terdiri dari lima sebesar 32 bit kata dan urutan 80 juga sebesar 32 bit kata. Lima kata pertama pada *buffer* kata diberi nama A, B, C, D, E sedangkan lima kata kedua diberi nama H<sub>0</sub>, H<sub>1</sub>, H<sub>2</sub>, H<sub>3</sub>, dan H<sub>4</sub>. Kemudian pada 80 kata yang berurutan diberi nama W<sub>0</sub>, W<sub>1</sub>, ..., W<sub>79</sub> dan pada penghitungan ini juga memakai sebuah variabel sementara, TEMP.
- Lakukan pengisian pesan, M dan kemudian *parsingkan* pesan tersebut ke dalam N 512 bit blok pesan, M<sup>(1)</sup>, M<sup>(2)</sup>, ..., M<sup>(n)</sup>. Caranya : 32 bit pertama dari blok pesan ditunjukkan ke M<sub>0</sub><sup>(i)</sup>, lalu 32 bit berikutnya adalah M<sub>1</sub><sup>(i)</sup> dan selanjutnya berlaku hingga M<sub>15</sub><sup>(i)</sup>.
- Inisialisasi Nilai *Hash* (dalam bentuk hex) :

$$H_0 = 67452301$$

$$H_3 = 10325476$$

$$H_1 = EFCDAB89$$

$$H_4 = C3D2E1F0$$

$$H_2 = 98BADCFE$$

- Lakukan proses M<sub>1</sub>, M<sub>2</sub>, ..., M<sub>n</sub> dengan cara membagi M<sub>i</sub> ke dalam 16 kata W<sub>0</sub>, W<sub>1</sub>, ..., W<sub>15</sub> dimana W<sub>0</sub> merupakan *left most*.
- Hitung : For t = 16 to 79

$$W_t = S^1(W_{t-3} \square W_{t-8} \square W_{t-14} \square W_{t-16})$$

- Inisialisasi 5 variabel A, B, C, D, dan E dengan nilai *Hash* :

$$A = H_0 ; B = H_1 ; C = H_2 ; D = H_3 ; E = H_4 .$$

- Hitung : For t = 0 to 79

$$TEMP = S^5(A) + f_t(B,C,D) + E + W_t + K_t$$

$$E = D ; D = C ; C = S^{30}(B) ; B = A ; A = TEMP .$$

- Hitung Nilai *Hash* :

$$H_0 = H_0 + A ; H_1 = H_1 + B ;$$

$$H_2 = H_2 + C ; H_3 = H_3 + D ;$$

$$H_4 = H_4 + E .$$

Hasil dari *message digest* sebesar 160 bit dari pesan, M adalah : H<sub>0</sub> H<sub>1</sub> H<sub>2</sub> H<sub>3</sub> H<sub>4</sub>.

## 2.4 Digital Signature

Sejak berabad-abad lamanya, tanda tangan digunakan untuk membuktikan autentikasi dokumen kertas (misalnya surat, piagam, ijazah, buku, karya seni, dan sebagainya). Tanda tangan mempunyai karakteristik sebagai berikut (Schneier, 1996) :

1. Tanda tangan adalah bukti yang autentik.
2. Tanda tangan tidak dapat dilupakan.
3. Tanda tangan tidak dapat dipindah untuk digunakan ulang.
4. Dokumen yang telah ditandatangani tidak dapat diubah.
5. Tanda tangan tidak dapat disangkal (*repudiation*).

Fungsi tanda tangan pada dokumen kertas juga diterapkan untuk autentikasi pada data digital seperti pesan yang dikirim melalui saluran komunikasi dan dokumen elektronik yang disimpan di dalam memori komputer. Tanda tangan pada data digital ini dinamakan tanda tangan digital (*digital signature*). Yang dimaksud dengan tanda tangan digital bukanlah tanda tangan yang di-digitalisasi dengan alat *scanner*, tetapi suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan. Dengan tanda tangan digital, maka integritas data dapat dijamin, di samping itu ia juga digunakan untuk membuktikan asal pesan (keabsahan pengirim).

Menandatangani pesan dapat dilakukan dengan salah satu dari dua cara:

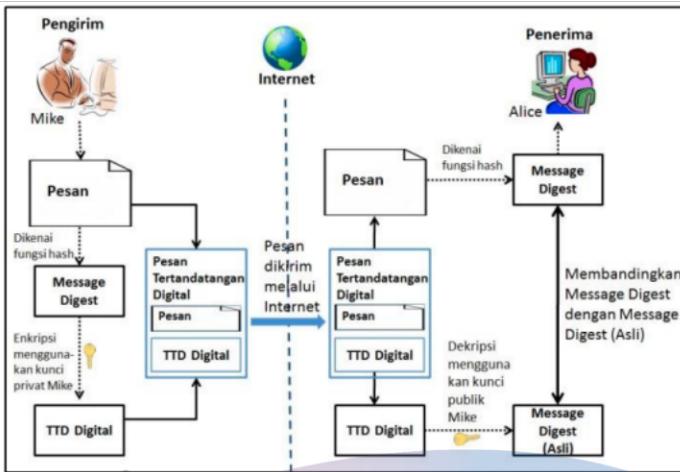
1. Enkripsi pesan

Mengenkripsi pesan dengan sendirinya juga menyediakan ukuran autentikasi. Pesan yang terenkripsi sudah menyatakan bahwa pesan tersebut telah ditandatangani.

2. Tanda tangan digital dengan fungsi *hash* (*hash function*)

Tanda tangan digital dibangkitkan dari *hash* terhadap pesan. Nilai *hash* adalah kode ringkas dari pesan. Tanda tangan digital berlaku seperti tanda tangan pada dokumen kertas. Tanda tangan digital ditambahkan (*append*) pada pesan. (Munir, 2011)

Gambar 2.13 memperlihatkan cara kerja tanda tangan digital. Mike menuliskan sebuah dokumen dan hendak mengirimkannya kepada Alice. Dengan menggunakan fungsi *hash*, Mike mengubah dokumen yang telah ditulisnya menjadi sebuah *message digest* dan kemudian mengenkripsi *message digest* tersebut menggunakan kunci privat miliknya. Hasil enkripsi *message digest* inilah yang disebut dengan tanda tangan digital. Tanda tangan digital ini kemudian dapat diattach ke dokumen asli atau dikirim secara terpisah tetapi pada waktu yang bersamaan, lalu dikirimkan kepada Alice melalui internet.



Gambar 2.7 Contoh Tanda Tangan Digital

Sumber: Azdy, 2016

Alice menerima pesan yang dikirim oleh Mike beserta tanda tangan digitalnya. Alice membagi kedua pesan tersebut menjadi dokumen asli dan tanda tangan digitalnya. Alice mengenakan fungsi *hash* dari dokumen asli dan memperoleh *message digest* dari dokumen tersebut. Alice melakukan dekripsi terhadap tanda tangan digital dengan menggunakan kunci publik milik Mike dan memperoleh *plaintext*-nya, kemudian membandingkan *plaintext* tersebut dengan *message digest* yang diperolehnya. Jika hasilnya sama, maka dapat diperoleh kesimpulan berupa:

1. Pesan tersebut benar ditulis oleh Mike karena hanya Mike yang mengetahui kunci privat miliknya.
2. Pesan tersebut tidak dimodifikasi oleh orang lain. Segala aktivitas yang dilakukan Mike disebut dengan proses *signing* atau penandatanganan, dan aktivitas yang dilakukan Alice adalah proses verifikasi.

Tanda tangan digital dapat menjamin keamanan dokumen yang ditandatanganinya dalam aspek *integrity*, *authentication*, serta *non-repudiation*.

1. *Integrity*: Tanda tangan digital dapat memastikan bahwa pesan yang dikirim adalah pesan yang sebenarnya dan tidak dimodifikasi pada saat transmisi. Hal ini dikarenakan pada dokumen asli tidak diberlakukan proses enkripsi, sehingga dokumen dapat dibaca oleh banyak pihak. Keaslian pesan dapat diperoleh dengan membandingkan *message digest* dari dokumen asli dengan *plaintext* hasil verifikasi tandatangan digital. Jika hasilnya sama, maka dokumen telah terjamin keasliannya.
2. *Authenticity*: Tanda tangan digital dibuat dengan mengenkripsi *message digest* dari pesan asli menggunakan kunci privat pengirim. *Ciphertext* ini hanya bisa didekripsi menggunakan pasangan kunci publik dari kunci privat tersebut, sehingga jika hasil verifikasi

membuktikan bahwa *message digest* dokumen sama dengan hasil dekripsi dari *ciphertext*, maka dapat dipastikan bahwa pengirim adalah benar orang yang memiliki kunci privat tersebut.

3. *Non-repudiation*: Jika tanda tangan digital telah terbukti ditandatangani menggunakan kunci privat tertentu, maka tidak dapat disangkal bahwa pemilik kunci privatlah yang telah menulis dokumen tersebut. (Azdy, 2016)

## 2.5 Metode Autentikasi Citra

Citra yang akan diamankan akan dikonversi terlebih dahulu dari ruang warna RGB ke ruang warna YCbCr. Komponen yang akan digunakan dalam proses selanjutnya hanya elemen Y (*luminance*) saja. Ciri yang akan membentuk *signature* citra akan diekstrak dari informasi yang terdapat pada koefisien DCT (*Discrete Cosine Transform*). DCT dari keseluruhan citra akan tidak efisien jika ditinjau dari dekorelasi informasi, tetapi juga karena tidak ada hubungan antara posisi piksel dan posisi koefisien DCT, sehingga area yang berubah tidak dapat ditentukan. Selain itu, metode autentikasi citra juga harus mampu mengenali citra autentik secara tepat walaupun citra tersebut telah dikompresi JPEG.

Oleh karena itu, elemen *luminance* dipecahkan menjadi blok berukuran 8 x 8 (ukuran yang sama dengan yang digunakan pada *coding* JPEG). Koefisien DCT dari setiap blok akan dipecahkan dengan kuantisasi matriks  $Q(q)$  dan hasilnya akan dibulatkan dengan operasi pembulatan (*round*).  $Q(q)$  untuk sebuah nilai faktor kualitas  $q$  tertentu dapat dirumuskan sebagai berikut:

$$Q(q) = \begin{cases} \text{round}\left(\frac{50 \cdot Q(50)}{q}\right), & 1 \leq q \leq 50 \\ \text{round}\left[50 \cdot Q(50) \cdot (2 - 0.02 \cdot q)\right], & 50 < q \leq 100 \end{cases} \quad (12)$$

Dimana  $Q(50)$  adalah matriks kuantisasi *luminance* JPEG standar untuk sebuah faktor kualitas sebesar 50.

Setelah operasi ini, diperoleh koefisien yang disebut sebagai koefisien DCT ternormalisasi. Dalam proses autentikasi citra ini, diinginkan agar dapat mengekstrak *signature* dengan benar walaupun citra telah dikompresi dengan menggunakan kompresi JPEG dengan sebuah faktor kualitas yang lebih besar daripada nilai minimum  $q_{\min}$ . Jika dipilih  $Q(q_{\text{low}})$  sebagai matriks kuantisasi JPEG untuk kompresi kasar (faktor kualitas rendah) dan ekstrak *signature* berdasarkan pada hasil ini, maka *signature* juga dapat diekstrak jika citra dikompresi dengan menggunakan sebuah faktor kualitas yang lebih besar daripada  $q_{\text{low}}$ . Perlu dicatat bahwa metode

ekstraksi *signature* tidak mengubah citra asli. Komputasi dilakukan hanya untuk mengekstrak *signature*, sementara citra tidak diotak-atik.

Algoritme ekstraksi ciri dapat dirincikan sebagai berikut.

Anggap  $C_{norm}$  adalah blok citra yang berisi elemen matriks koefisien DCT ternormalisasi. Elemennya adalah  $C_{norm}(x, y)$ , dimana nilai x dan y memiliki *range* nilai dari 0 sampai 7. Nilai  $C_{norm}(0, 0)$  dikenal sebagai koefisien DC ternormalisasi dan yang lainnya disebut sebagai koefisien AC ternormalisasi. Delapan bit diekstrak dari setiap blok menggunakan nilai *threshold* = 5 yang disimbolkan sebagai  $t_i$ , dengan  $i = 1 \dots 5$ , dengan perincian: tiga bit pertama diperoleh dengan membandingkan komponen DC ternormalisasi dengan tiga buah nilai *threshold* pertama. Asumsikan nilai maksimum untuk koefisien DC ternormalisasi disimbolkan sebagai  $C_{norm}(0,0)_{max}$ , nilai *threshold*-nya adalah  $t_1 = C_{norm}(0,0)_{max}/4$ ,  $t_2 = C_{norm}(0,0)_{max}/2$  dan  $t_3 = 3 * C_{norm}(0,0)_{max}/4$ . Untuk tiga bit berikutnya, nilai ternormalisasi dari tiga koefisien AC pertama dengan urutan pengambilan/pembacaan secara zig-zag JPEG  $C_{norm}(0, 1)$ ,  $C_{norm}(1, 0)$ ,  $C_{norm}(2, 0)$  akan dibandingkan dengan  $t_4 = 0$ . Untuk menghasilkan dua bit terakhir, nilai absolut dari dua koefisien AC pertama akan dibandingkan dengan nilai *threshold* yang ditentukan secara empiris, yaitu  $t_5 = 3$ . Hasil perbandingan berupa nilai bit 1 jika nilai koefisien lebih besar atau sama dengan nilai *threshold* dan menghasilkan nilai bit 0 jika sebaliknya. Ukuran dari *signature* adalah sebesar 0.52% dari ukuran citra tidak terkompresi.

*Signature* autentikasi untuk setiap blok berukuran 8 x 8 akan digabungkan, sehingga akan diperoleh sebuah vektor biner dengan ukuran  $s = MN/8$  bit, dimana M = lebar citra dan N = tinggi citra. Setiap vektor biner akan dienkripsi dengan menggunakan algoritme enkripsi AES-128. Setelah itu, *signature* terenkripsi yang dihasilkan akan ditempelkan pada *blockchain*.

Untuk menguji autentikasi dari sebuah citra, *user* harus memiliki kunci privat untuk mendekripsi *signature* terenkripsi pada *blockchain*. Kemudian, mencari data tersebut pada *blockchain*, mengambil *signature* terenkripsi pada *blockchain*, mendekripsinya dan membandingkannya dengan *signature* yang terekstrak dari citra untuk diuji dengan menggunakan algoritme yang sama. Jika dua *signature* sama, maka citra tersebut autentik. Jika tidak, maka citra tidak autentik dan dapat mengindikasi area yang rusak (berubah). Hal ini dapat dilakukan karena *signature* merupakan penggabungan dari data yang terekstrak dari setiap blok. Hal ini berarti bahwa jika citra berubah maka beberapa blok dapat ditandai sebagai autentik sementara blok lainnya tidak autentik. Dengan mengetahui indeks dari blok, maka posisi pada citra dapat diketahui (Dobre, et al., 2018).

## 2.6 Confusion Matrix

*Confusion matrix* adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi. Rumus ini melakukan perhitungan dengan 4 keluaran, yaitu: *recall*, *precision*, *accuracy* dan *error rate*.

Tabel 2.1 Tabel *Confusion Matrix*

Correct Classification	Classified as	
	Predicted "+"	Predicted "-"
Actual "+"	True Positives	False Negatives
Actual "-"	False Positives	True Negatives

Sumber: Rahman, et. al., 2017

Berdasarkan tabel *Confusion Matrix* di atas:

- a. *True Positives* (TP) adalah jumlah *record* data positif yang diklasifikasikan sebagai nilai positif
- b. *False Positives* (FP) adalah jumlah *record* data negatif yang diklasifikasikan sebagai nilai positif
- c. *False Negatives* (FN) adalah jumlah *record* data positif yang diklasifikasikan sebagai nilai positif
- d. *True Negatives* (TN) adalah jumlah *record* data negatif yang diklasifikasikan sebagai nilai *negative*

Nilai yang dihasilkan melalui metode *Confusion Matrix* adalah berupa evaluasi sebagai berikut:

- a. *Accuracy*, yaitu persentase jumlah *record* data yang diklasifikasikan (prediksi) secara benar oleh algoritme.

$$\text{Accuracy} = (TP + TN) / \text{Total data} \quad (13)$$

- b. *Misclassification (Error) Rate*, yaitu persentase jumlah *record* data yang diklasifikasikan (prediksi secara salah oleh algoritme).

$$\text{Misclassification Rate} = (FP + FN) / \text{Total data} \quad (14)$$

(Rahman, et al., 2017)

## 2.7 Blockchain

*Blockchain* adalah blok yang berisi informasi digital yang terus tumbuh, blok tersebut terhubung satu sama lain dan diamankan oleh kriptografi. Tiap blok dalam *blockchain* berisi tanda waktu, hash dari blok sebelumnya dan informasi yang disimpan, contohnya data transaksi.

*Blockchain* awalnya dibuat untuk jaringan yang terdesentralisasi untuk pertukaran uang secara digital. Teknologi *blockchain* konsisten dalam pendistribusian data tanpa memerlukan entitas yang dipercaya untuk melakukan transaksi. Untuk mencapai hal ini, *database* berisi semua blok-blok yang terhubung dan memiliki salinan yang bebas disimpan siapapun. Keamanannya tetap terjaga karena untuk modifikasi data pada blok harus melibatkan blok-blok lain yang terhubung. Proses modifikasi akan sangat sulit.

*Blockchain* merupakan basis data terdistribusi yang digunakan untuk memelihara daftar *record* yang terus berkembang, yang disebut dengan blok. Setiap blok mengandung penanda waktu (*timestamp*) dan tautan (*link*) ke blok sebelumnya. Pada umumnya, *Blockchain* dikelola oleh jaringan *peer-to-peer* yang secara kolektif mematuhi protokol tertentu untuk memvalidasi blok baru. Teknologi Ledger terdistribusi ini dapat menjadi suatu *framework* yang memungkinkan inovasi radikal dalam banyak bidang.

*Blockchain* merupakan struktur data terdistribusi yang direplikasi dan dibagi di antara anggota jaringannya. *Blockchain*, pada mulanya diperkenalkan sebagai solusi untuk pengeluaran ganda (*double spending*) pada *Bitcoin*. Sebagai hasil dari cara *node* pada jaringan *Bitcoin* (disebut *miner*) divalidasi, dan menyepakati transaksi yang terjadi di atasnya. *Blockchain* pada *Bitcoin* menyediakan wadah untuk ledger transaksi otoritatif yang menentukan siapa yang memiliki transaksi tersebut (Arief & Sundara, 2017).

Meskipun demikian, *Blockchain* dapat berdiri dengan sendirinya, tanpa perlu terkait dengan *cryptocurrency*. *Blockchain* dapat digambarkan sebagai suatu log yang *record*-nya dibatch dengan *block* yang diberikan tanda waktu (*timestamp*). Setiap blok ditandai dengan *hash* kriptografi. Setiap blok merujuk pada *hash* blok yang sebelumnya. Ini menimbulkan tautan (*link*) di antara blok-blok tersebut, sehingga membuat suatu rantai blok (*chain*).

Setiap *node* yang terhubung pada daftar blok yang terhubung dan tertaut dengan blok sebelumnya (*backlinked*) dapat membaca dan mendapatkan gambaran mengenai keadaan dunia data terkini yang sedang dipertukarkan di dalam jaringan. Gambaran mengenai cara kerja *Blockchain* dapat diperoleh dengan memahami bagaimana jaringan *Blockchain* (*Blockchain network*) bekerja. Ini merupakan himpunan *node* (*client*) yang beroperasi pada *Blockchain* yang sama melalui salinan yang dimiliki oleh setiap *node*.

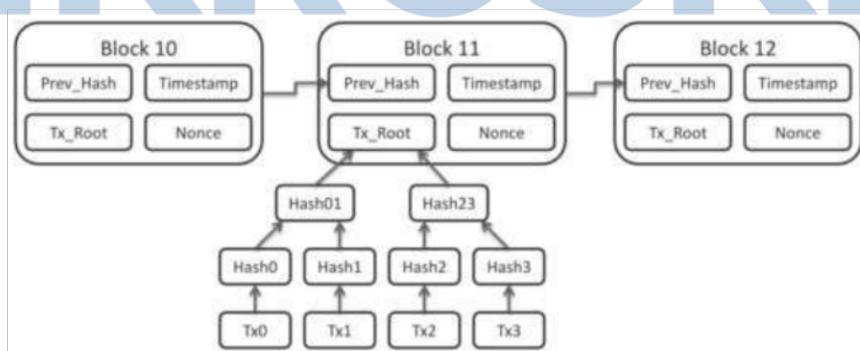
Suatu *node*, secara umum, dapat berperan sebagai titik masuk (*entry point*) untuk pengguna yang berbeda-beda pada *Blockchain*, tetapi untuk mempermudah, setiap pengguna dianggap bertransaksi pada *Blockchain* melalui *node* mereka sendiri.

1. Pengguna berinteraksi dengan *Blockchain* melalui sepasang *public* dan *private key*. Mereka menggunakan kunci privat (*private key*) untuk menandai (*sign*) transaksi mereka sendiri,

dan alamat mereka dapat ditelusuri melalui kunci publik (*public key*) mereka yang tersedia di jaringan. Penggunaan kriptografi asimetris membawa integritas, autentikasi, dan *non-repudiation* ke dalam jaringan. Setiap transaksi yang ditandatangani disiarkan melalui *node pengguna* ke *peer* satu loncatan.

2. *Peer* yang bertetangga memastikan bahwa transaksi ini valid sebelum me-*relay* lebih jauh. Transaksi yang tidak valid akan diabaikan. Pada akhirnya, transaksi akan disebarluaskan ke seluruh jaringan.
3. Transaksi yang telah dihimpun dan divalidasi oleh jaringan menggunakan proses di atas dalam rentang waktu yang disepakati, diurut dan dipaketkan pada kandidat *block* yang diberi *timestamp*. Proses ini disebut dengan *mining*. *Node mining* akan menyebarkan kembali blok ini ke dalam jaringan.
4. *Node* lain akan memverifikasi bahwa blok yang disarankan (1) mengandung transaksi yang valid, dan (2) merujuk lewat *hash* blok sebelumnya dari rantai yang tepat. Apabila terjadi demikian, blok tersebut akan ditambahkan ke dalam rantai. Apabila sebaliknya, blok tersebut akan diabaikan. Ini menandai akhir dari suatu siklus.

Cara kerja *Blockchain* terlihat pada Gambar 2.8. Proses ini berlangsung terus menerus. Pada dasarnya, *Blockchain* merupakan himpunan dari penulis yang tidak saling mempercayai yang berbagi *database* tanpa adanya pihak perantara yang dipercaya. Dalam rangka mencegah terjadinya kekacauan dalam lingkungan terdistribusi ini, dan untuk mencapai suatu *global consensus*, setiap jaringan *Blockchain* perlu menerapkan sekumpulan aturan yang harus dipatuhi oleh setiap transaksi *database*. Aturan ini diprogram pada setiap *client Blockchain*, yang kemudian akan menggunakan aturan tersebut untuk memeriksa apakah suatu transaksi valid atau tidak dan, sebagai konsekuensinya, apakah transaksi tersebut akan diteruskan (*relay*) ke jaringan atau tidak.

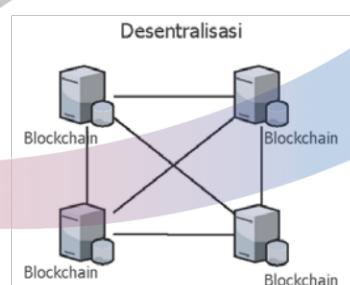


Gambar 2.8 Cara Kerja *Blockchain*

Sumber: (Arief & Sundara, 2017)

Dengan demikian, *Blockchain* memiliki ciri-ciri berikut:

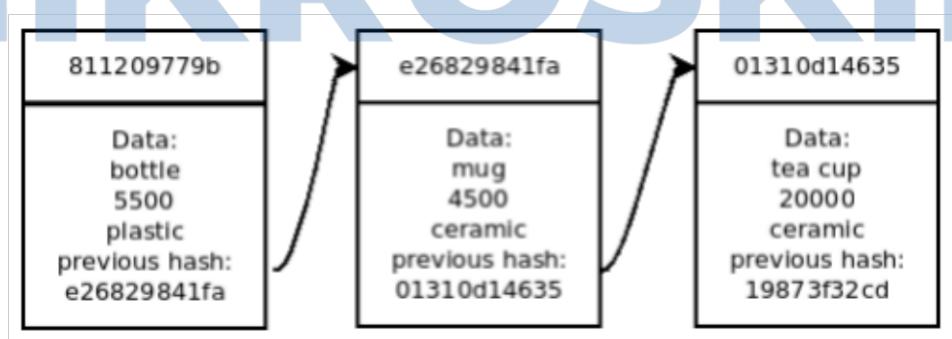
1. Terdesentralisasi, *Blockchain* menggunakan model jaringan terdistribusi dengan algoritme konsensus untuk memelihara konsistensi data dalam jaringannya.
2. Persisten, Hampir tidak mungkin menghapus atau me-*rollback* transaksi yang pernah terjadi dalam jaringan *Blockchain*. Blok yang mengandung transaksi yang tidak valid dapat ditemukan segera.
3. Anonim. Setiap pengguna dapat berinteraksi dalam *Blockchain* dengan alamat yang *disguise*, yang tidak mengungkapkan identitas riil pengguna. Ini merupakan karakteristik *Blockchain*, meskipun anonimitas pengguna tidak selalu dapat dijaga karena adanya batasan-batasan intrinsik.
4. Dapat diaudit, Transaksi dalam *Blockchain* dapat dengan mudah diverifikasi dan dilacak. *Bitcoin*, sebagai contoh penggunaan *Blockchain*, menyimpan data pengguna pada model *Unspent Transaction Output* (UTXO). Setiap transaksi harus merujuk pada transaksi sebelumnya yang belum dipergunakan. *Blockchain* dapat terbagi ke dalam: (1). Publik, (2). Privat, dan (3). Konsorsium (Arief & Sundara, 2017).



Gambar 2.9 Konsep Desentralisasi pada *Blockchain*

Sumber: Putra, et. al., 2018

*Blockchain* selain sebagai sebuah kumpulan blok yang terhubung dengan mencatat tanda digital / *hash* dari blok sebelumnya dapat digambarkan seperti gambar 2.10.



Gambar 2.10 Ilustrasi dari *Blockchain*

Sumber: Yulianton, et. al., 2018

Dari gambar 2.10 dapat dikatakan bahwa blok dalam *blockchain* memiliki 3 komponen penting, yaitu data, *previous hash* dari blok sebelumnya, dan *hash* beserta waktu saat blok

dibuat untuk blok yang bersangkutan. Mekanisme yang digunakan di dalam *blockchain* untuk memastikan keamanan dari *blockchain* terjaga ada beberapa hal, yaitu (Yulianton, et al., 2018) :

1. *Mekanisme tanda digital / hash.*

Tanda digital/*hash* merupakan sebuah hasil dari pemanfaatan teknik *hash*. Teknik *hash* sendiri merupakan salah satu teknik kriptografi. *Hash* di dalam kriptografi merupakan sebuah algoritme matematika yang memetakan data secara acak ke dalam bentuk *string* berukuran tetap (*hash*) dan dirancang untuk menjadi sebuah fungsi satu arah, yaitu fungsi yang hasilnya tidak dapat dibalikkan ke bentuk asalnya. Jadi dapat dikatakan bahwa bila sebuah informasi, katakanlah sebuah baris informasi yang dibuat nilai *hash*nya melalui fungsi *hash* kriptografi, maka tanda digital yang muncul adalah benar-benar unik milik baris informasi tersebut dan tidak dapat dibalikkan ke bentuk asalnya. Bila baris informasi yang sudah dibuat tanda digitalnya mengalami perubahan informasi, maka nilai *hash* kriptografinya juga akan berubah.

2. *Mekanisme proof-of-work.*

Mekanisme ini adalah sebuah mekanisme untuk memperlambat pembuatan blok baru dengan tujuan agar proses penghitungan *proof-of-work* seluruh blok dapat terjaga. Sebagai contoh, *blockchain* yang digunakan oleh *bitcoin* memerlukan waktu 10 menit untuk melakukan mekanisme *proof-of-work* saat sebuah blok baru ditambahkan.

3. *Mekanisme peer-to-peer.*

Merupakan sebuah mekanisme untuk mendistribusikan catatan transaksi kepada orang-orang atau *user* yang tergabung dalam jaringan *peer-to-peer blockchain*. Pada intinya, mekanisme ini akan bekerja seperti berikut ini; seseorang yang baru bergabung dengan jaringan *peer-to-peer blockchain* akan memperoleh salinan lengkap transaksi yang terjadi di dalam jaringan tersebut. Dan bila suatu saat terjadi sebuah transaksi atau penambahan blok baru maka informasi mengenai blok baru tersebut akan dikirimkan ke semua *node/user* yang ada dan diverifikasi. Setelah proses verifikasi selesai dan blok baru tersebut dinyatakan valid, maka blok baru tersebut akan dicatatkan kedalam *blockchain* masing-masing *user*.

### 2.7.1 Permissioned Blockchain

*Permissioned Blockchain* atau *private blockchain* adalah jenis *blockchain* yang hanya mengizinkan pembaca dan penulis secara terbatas. Di sini, entitas sentral memutuskan dan menghubungkan hak dengan *peers* individu untuk berpartisipasi dalam operasi tulis atau baca *blockchain*. Untuk menyediakan enkapsulasi dan privasi, pembaca dan penulis juga dapat berjalan di *blockchain* paralel yang terpisah yang saling berhubungan. Contoh yang paling

dikenal luas dari *permissioned blockchain* adalah Hyperledger Fabric dan R3 Cord. *Permissioned blockchain* berjalan pada sekumpulan partisipan atau *node* yang sudah diketahui dan teridentifikasi (Prabandari , et al., 2019).

*Permissioned blockchain* menggunakan *layer* pengaturan akses untuk mengendalikan siapa yang memiliki akses ke dalam jaringan. Berbeda dengan jaringan *blockchain* publik, *validator* pada jaringan *blockchain private* diperiksa oleh pemilik jaringan. Mereka tidak bergantung pada *node* anonim dalam memvalidasi transaksi atau mengambil manfaat dari *network effect*. *Permissioned blockchain* juga bisa menggunakan nama *blockchain* "konsorsium" atau "hybrid". New York Times mencatat pada tahun 2016 dan 2017 banyak perusahaan yang memanfaatkan jaringan *blockchain* dengan *private blockchain*, yang independen dari sistem publik.

Nikolai Hampton mengungkapkan dalam Computerworld bahwa Tidak perlu '51 persen' serangan pada *private blockchain*, karena *private blockchain* (kemungkinan besar) sudah mengendalikan 100 persen dari semua sumber daya dalam penyusunan blok. Jika *User* mampu menyerang atau merusak perangkat penyusunan *blockchain* pada peladen perusahaan *private*, *User* dapat secara efektif mengendalikan 100 persen jaringan mereka dan mengubah transaksi sesuai keinginan *User*. Hal ini memiliki serangkaian implikasi yang buruk selama krisis finansial atau krisis utang seperti krisis finansial 2007–08, di mana aktor yang kuat secara politik dapat mengambil keputusan yang mendukung beberapa kelompok dengan mengorbankan orang lain, dan "*blockchain bitcoin* dilindungi oleh upaya penambangan (*mining*) kelompok besar. Mustahil bagi *private blockchain* dapat melindungi datanya menggunakan gigawatt daya komputasi — ini memakan waktu dan tidak murah." Dia juga mengatakan, "Dalam *private blockchain* juga tidak ada 'kompetisi'; tidak ada insentif jika menggunakan lebih banyak daya atau menemukan blok lebih cepat daripada yang lain. Ini artinya bahwa banyaknya solusi *blockchain internal* tidak lebih dari sekadar basis data yang rumit."

### 2.7.2 Permissionless Blockchain

*Permissionless Blockchain* adalah jenis *blockchain* yang terbuka dan terdesentralisasi. Setiap *peer* dapat bergabung dan meninggalkan jaringan sebagai pembaca atau penulis di waktu kapanpun. Lebih menariknya, tidak ada pusat yang mengatur keanggotaan, atau tidak ada yang dapat mencegah pembaca atau penulis. Keterbukaan ini membuat konten si penulis dapat dibaca oleh *peer* siapa saja.

*Permissionless Blockchain* (*blockchain* tanpa izin) adalah *blockchain* yang tidak memerlukan izin untuk bergabung dan berinteraksi dengannya. Mereka juga dikenal sebagai

*blockchain* publik. *Permissionless blockchain* sangat ideal untuk menjalankan dan mengelola mata uang digital.

Dalam *permissionless blockchain*, pengguna dapat membuat alamat pribadi dan kemudian berinteraksi dengan jaringan dengan membantu jaringan untuk memvalidasi transaksi atau hanya mengirim transaksi ke pengguna lain di jaringan. Jenis pertama dari *permissionless blockchain* adalah *Bitcoin*. Ini memungkinkan pengguna untuk mentransfer mata uang digital di antara mereka sendiri. Selain itu, pengguna dapat berinteraksi dengan jaringan dengan berpartisipasi dalam proses penambangan. Ini adalah proses memecahkan persamaan matematika yang kompleks dan kemudian menggunakan untuk memvalidasi transaksi. Algoritme konsensus yang digunakan oleh *bitcoin* adalah *Proof-of-Work* (PoW).

Ada juga *blockchain* lain yang tidak memiliki izin. *Ethereum* (ETH) adalah jenis tanpa izin publik populer lainnya yang menggunakan metode konsensus lain *Proof-of-Stake* (PoS). Ini juga memperkenalkan konsep lain seperti kontrak pintar.

*Blockchain* tanpa izin memiliki beberapa karakteristik yang menarik, yaitu:

1. Benar-benar terdesentralisasi

Teknologi *blockchain* tanpa izin benar-benar terdesentralisasi.

2. Anonimitas

*Blockchain* tanpa izin terbuka untuk semua orang. Namun, itu tidak berarti bahwa ini tidak anonim. Siapa pun yang bergabung dengan jaringan dapat tetap anonim karena *User* tidak memerlukan KYC untuk bergabung dan menjelajahi jaringan.

3. Transparansi

*Node* publik dapat melihat transaksi, membuat jaringan transparan.

4. Kepercayaan

*User* dapat melacak atau membaca transaksi. Dengan demikian, *User* dapat mempercayai *blockchain* tanpa izin ini lebih dari sekadar *blockchain* tertutup atau diizinkan.

5. Tidak dapat diubah

Setiap data di platform tidak dapat diubah, itu artinya *User* tidak dapat mengubahnya kapan saja.

6. Keamanan yang ditingkatkan

Kriptografi dan parameter keamanan lainnya membuat *blockchain* tanpa izin menjadi lebih aman.

Selain itu, karakteristik besar lainnya dari *blockchain* tanpa izin adalah bahwa siapa pun dapat bergabung dengan jaringan jika mereka mau. *Blockchain* tanpa izin juga sangat bagus

dalam hal memberi insentif kepada pengguna di jaringan. Mereka dapat digunakan untuk kemajuan peserta karena membawa transparansi dan kepercayaan ke seluruh jaringan.

## 2.8 Kriptografi

Kriptografi berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu *kripto* dan *graphia*. *Kripto* artinya menyembunyikan, sedangkan *graphia* artinya tulisan. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentifikasi data. Tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi.

Kriptografi dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan dikirim dari suatu tempat ke tempat lain, isi pesan tersebut mungkin dapat disadap oleh pihak lain yang tidak berhak untuk mengetahui isi pesan tersebut. Untuk menjaga pesan, maka pesan tersebut dapat diubah menjadi suatu kode yang tidak dapat dimengerti oleh pihak lain. Enkripsi adalah sebuah proses penyandian yang melakukan perubahan sebuah kode atau pesan dari yang bisa dimengerti, disebut *plaintext*, menjadi sebuah kode yang tidak bisa dimengerti, disebut dengan *ciphertext*. Sedangkan proses kebalikannya untuk mengubah *ciphertext* menjadi *plaintext* disebut dekripsi. Proses enkripsi dan dekripsi memerlukan suatu mekanisme dan kunci tertentu, dan kesatuan sistem ini sering disebut dengan *cipher*.

Berdasarkan sifat kuncinya, kriptografi dibagi menjadi dua, yaitu kriptografi simetris dan kriptografi asimetris. Pada kriptografi simetris, proses enkripsi dan dekripsi dilakukan menggunakan kunci rahasia yang sama. Sedangkan pada kriptografi asimetris, proses enkripsi dan dekripsinya menggunakan kunci yang berbeda, yaitu kunci publik untuk enkripsi, dan kunci rahasia yang digunakan untuk dekripsi.

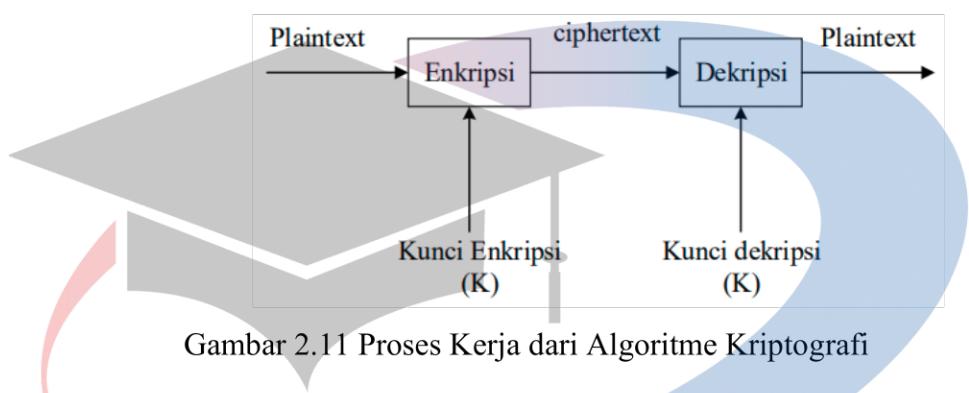
Berdasarkan waktu kemunculannya, kriptografi dibedakan menjadi dua, yaitu kriptografi klasik dan kriptografi modern. Pada kriptografi klasik, proses enkripsi menggunakan perhitungan yang sederhana dan dapat dilakukan secara manual. Sedangkan pada kriptografi modern, proses enkripsi menggunakan perhitungan yang rumit dan melibatkan bilangan yang besar, sehingga diperlukan bantuan komputer (Hasugian, 2017).

### 2.8.1 Terminologi

Hingga zaman modern seperti saat ini, kriptografi semata-mata dianggap sebagai enkripsi, yaitu proses mengubah informasi yang tidak biasa dan tidak dapat dibaca menjadi suatu informasi yang jelas dan dapat dibaca. Sedangkan dekripsi adalah proses sebaliknya.

*Ciphertext* tersebut adalah suatu pasangan algoritme yang melakukan enkripsi dan

membalikkan dekripsi. Informasi detail dari *ciphertext* dikontrol oleh algoritme tersebut, dengan kata lain dengan suatu kunci. Hal tersebut merupakan parameter rahasia untuk membaca pesan rahasia tersebut, dan biasanya hanya pengirim dan yang dikirim yang mengetahui kunci tersebut. Kunci tersebut amatlah penting karena tanpa kunci itu, pesan tersebut akan mudah terbongkar dan menjadi tidak berarti lagi. Berdasarkan sejarahnya, *ciphertext* kadang kala digunakan langsung untuk mengenkripsi atau deskripsi tanpa prosedur tambahan seperti pengesahan dan pengecekan kepribadian (Hasugian, 2017). Gambaran proses kerja dari algoritme kriptografi secara umum dapat dilihat pada gambar 2.1.



Gambar 2.11 Proses Kerja dari Algoritme Kriptografi

Sumber: Meko, 2018

Dalam bahasa sehari-hari, kode biasanya digunakan untuk mengartikan suatu metode enkripsi atau penyembunyian suatu makna. Tetapi, dalam kriptografi, kode memiliki arti lebih spesifik; berarti suatu pergantian dari suatu unit dari suatu informasi dengan kata kode (sebagai contoh, apple pie diganti dengan attack at dawn). Kode tidak digunakan lagi dalam kriptografi yang sesungguhnya kecuali tidak sengaja seperti proses desain suatu unit (contoh ‘Bronco Flight’ atau Operation Overlord)- sejak ciphertext yang dipilih lebih praktis dan lebih aman dari biasanya, serta lebih mudah disesuaikan dengan komputer. Beberapa penggunaan kriptografi dan kriptologi dapat saling bertukar tempat dalam bahasa Inggris, ketika penggunaan kriptografi yang lain mengarah ke penggunaan dan praktik dari teknik (Hasugian, 2017).

### 2.8.2 Kriptografi Modern

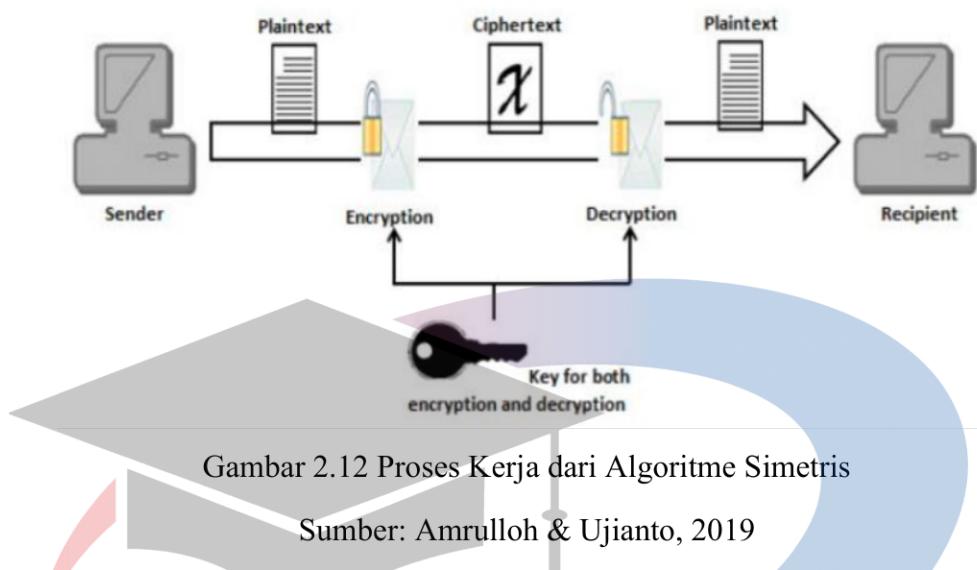
Bidang kriptografi modern dapat dibagi menjadi beberapa area studi, sebagai berikut:

a. Kriptografi Kunci-Simetris

Kriptografi kunci-simetris mengarah kepada metode enkripsi yang mana baik pengirim maupun yang dikirim saling memiliki kunci yang sama (walaupun kebanyakan kunci yang ada sedikit berbeda namun masih berhubungan dalam hal kemudahan perhitungan).

Algoritme ini disebut simetris karena memiliki *key* atau kunci yang sama dalam proses enkripsi dan dekripsi sehingga algoritme ini juga sering disebut algoritme kunci tunggal

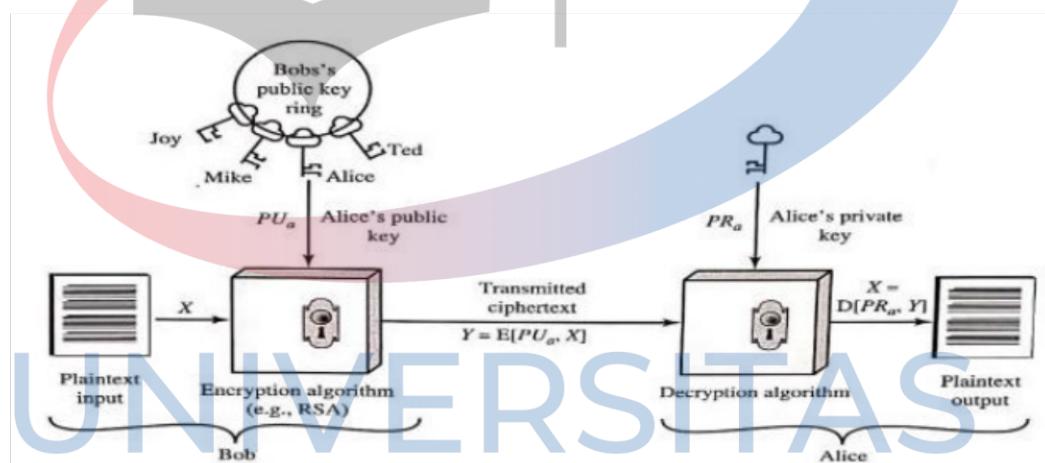
atau satu kunci. *Key* dalam algoritme ini bersifat rahasia atau *private key* sehingga algoritme ini juga disebut dengan algoritme kunci rahasia (Meko, 2018). Gambaran proses kerja dari algoritme kriptografi kunci-simetris dapat dilihat pada gambar 2.2.



*Secret-key cryptography* kadang disebut sebagai *symmetric cryptography* merupakan bentuk kriptografi yang lebih tradisional, dimana sebuah kunci tunggal dapat digunakan untuk mengenkripsi dan mendekripsi pesan. *Secret-key cryptography* tidak hanya berkaitan dengan enkripsi tetapi juga berkaitan dengan autentikasi. Salah satu teknik semacam ini disebut *message authentication codes*. *Data Encryption Standard* (DES) dan *Advanced Encryption Standard* (AES) adalah salah satu desain sandi balok yang sudah didesain standar kriptografi oleh pemerintah AS. Meskipun terdapat bantahan dari standar resminya, DES masih cukup terkenal dan digunakan sebagai aplikasi yang sudah luas penggunaannya, dari enkripsi ATM sampai privasi *email* dan akses keamanan. Banyak sandi balok lain yang telah didesain dan diluncurkan ke publik dengan mempertimbangkan kualitas dalam berbagai variasi. Tetapi banyak pula yang sudah terbongkar. Sandi gelombang berlawanan dengan sandi balok, membuat material gelombang panjang yang berubah-ubah yang dikombinasikan dengan kode tulisan bit demi bit atau karakter demi karakter. Masalah utama yang dihadapi *secret-key cryptosystems* adalah membuat pengirim dan penerima menyetujui kunci rahasia tanpa ada orang lain yang mengetahuinya. Ini membutuhkan metode dimana dua pihak dapat berkomunikasi tanpa takut akan disadap. Kelebihan *secret-key cryptography* dari *public-key cryptography* adalah lebih cepat.. Teknik yang paling umum dalam *secret-key cryptography* adalah *block ciphers*, *stream ciphers*, dan *message authentication codes*.

## b. Kriptografi Kunci-Publik/Asimetris

Seperti yang telah disebutkan dalam artikel sebelumnya, algoritme sandi dapat dikelompokkan menjadi 3 kategori yaitu : sistem sandi simetris, sistem sandi asimetris dan sistem sandi *hashing*. Masing-masing sistem sandi ini memiliki cara yang berbeda dalam metode penyandiannya. Sistem sandi asimetris atau dikenal juga sebagai sistem sandi kunci publik adalah sistem sandi yang metode menyandi dan membuka sandinya menggunakan kunci yang berbeda. Tidak seperti sistem sandi simetris, sistem sandi ini relatif masih baru. Algoritme sandi jenis ini yang telah terkenal diantaranya RSA (Rivest-Shamir-Adleman), ElGamal, dan Diffie-Hellman. Sistem ini memiliki sepasang kunci yang disebut kunci publik yaitu kunci yang didistribusikan secara umum dan kunci privat yaitu kunci yang dirahasiakan yang hanya dimiliki oleh pihak yang berhak. Umumnya kunci publik digunakan untuk menyandi dan kunci privat digunakan untuk membuka sandi. Gambaran proses kerja dari algoritme asimetris dapat dilihat pada gambar 2.3.



Gambar 2.13 Proses Kerja dari Algoritme Asimetris

Sumber: Saputro, et. al., 2020

Sistem sandi asimetris bekerja lebih lambat dari sistem sandi simetris, sehingga sistem sandi ini lebih sering digunakan untuk menyandi data dengan ukuran bit yang kecil. Sistem sandi ini sering pula digunakan untuk mendistribusikan kunci sistem sandi simetris. Penggunaan lain sistem sandi asimetris adalah dalam tandatangan digital. Tandatangan digital seperti halnya tandatangan biasa digunakan untuk membuktikan keaslian dari suatu dokumen yang dikirimkan. Kunci privat digunakan untuk menandatangani, sedangkan kunci publik digunakan untuk membuktikan keaslian tandatangan itu (Fahmi and Faidah, 2010). Untuk lebih memudahkan pengertian tandatangan digital dapat diilustrasikan sebagai berikut : Untuk menandai pesannya, si Pengirim menyandi pesan tersebut dengan kunci privatnya. Setiap orang yang memiliki pasangan kunci publiknya dapat membuka pesan tersandi itu

dan mengetahui dengan pasti si Pengirim adalah orang yang tepat. Cara ini tidak melindungi kerahasiaan datanya, mengingat setiap orang dapat saja memiliki pasangan kunci publik dari si Pengirim. Tujuan dari tandatangan digital hanyalah membuktikan bahwa pesan tersebut memang dari si Pengirim. Karena kunci publik didistribusikan secara umum, kita mempunyai permasalahan yang berbeda dengan sistem sandi simetris. Permasalahan utamanya adalah apakah kunci publiknya berada ditangan yang tepat? Untuk mengatasi masalah tersebut maka Infrastruktur Kunci Publik (PKI) mencoba memberikan pemecahannya. Namun karena masih dalam tahap pengembangan, PKI tidak memberikan jaminan. Masih membutuhkan waktu lama untuk dapat menerima solusi PKI ini. Sistem kriptografi kunci-simetris secara tipikal menggunakan enkripsi dan dekripsi yang sama meskipun pesan ini memiliki kunci berbeda satu sama lain. Secara signifikan, ketidakuntungan dari sistem ini adalah manajemen kunci yang diperlukan untuk keamanan. Setiap pasang komunikasi yang berjarak jauh harus memiliki kunci yang berbeda. Setiap kunci yang bertambah akan menambahkan jarak dari anggota jaringan yang mana akan membutuhkan manajemen kunci yang lebih teliti lagi agar terjamin keamanannya. Hal yang membuat sulit adalah kesulitan dalam menempatkan kunci rahasia diantara kelompok yang berkomunikasi. Algoritme kunci publik ini biasanya berdasarkan kompleksitas komputasional dari masalah yang “sulit”, biasanya dari teori angka. Sebagai contoh, kekerasan dari RSA biasanya berhubungan dengan masalah faktorisasi *integer*, ketika Diffie Hellman dan DSA berkaitan dengan masalah logaritma diskrit. Lebih jauh lagi, kriptografi kurva cekung telah berkembang dari masalah keamanan yang ada. Karena kesulitan dari masalah tersebut, algoritme kunci-publik termasuk operasi modular seperti perkalian dan eksponensial, yang mana hal tersebut secara komputasi lebih mahal daripada teknik lain yang digunakan oleh *ciphertext*, terutama yang menggunakan kunci spesifik. Hasilnya, sistem kriptografi kunci-publik merupakan kriptosistem hibrida secara umum, yang mana algoritme kunci-simetris kualitas tinggi digunakan sebagai pesan tersebut. Persamaannya, skema tanda tangan hibrida lebih sering digunakan, yang mana fungsi kriptografi diperhitungkan dan hasilnya akan berlaku secara digital (Hasugian, 2017)

## 2.9 Algoritme AES (Advanced Encryption Standard)

AES merupakan nama untuk *Federal Information Processing Standards Publication* 197. AES menjelaskan mengenai algoritme kriptografi yang digunakan untuk melindungi data sebagai pengganti DES. Algoritme AES adalah sebuah *symmetric block cipher* yang dapat

melakukan enkripsi dan dekripsi pada data. Algoritme 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 algoritme ini sering disebut dengan “AES-128”, “AES-192”, atau “AES-256” sesuai dengan kunci yang dipakai, sedangkan ukuran blok yang digunakan adalah 128 bit (Federal Information Processing Standards Publication, 197, FIPS, 2001).

### 2.9.1 Notasi

Notasi yang digunakan dalam algoritme AES 128 dapat dirincikan sebagai berikut:

#### 1. *Input* dan *Output*

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

#### 2. *Byte*

Unit dasar yang digunakan untuk proses di dalam algoritme AES adalah *byte*, yaitu deretan yang terdiri dari 8 bit. Setiap nilai *byte* pada algoritme 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 2.14 sebagai berikut :

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_7 \\ \vdots \\ a_{15} \end{bmatrix} = \begin{bmatrix} \text{input}0, \text{input}1, \dots, \text{input}7 \\ \text{input}8, \text{input}9, \dots, \text{input}15 \\ \text{input}16, \text{input}17, \dots, \text{input}23 \\ \vdots \\ \vdots \\ \text{input}120, \text{input}121, \dots, \text{input}127 \end{bmatrix}$$

Gambar 2.14 *Input Array of Byte 128 bit*

Sumber: (Bruce Schneier, 1996)

#### 4. State

Operasi pada algoritme 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  $0 \leq c < \text{Nb}$ . *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 2.15.

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

Gambar 2.15 *State Array* terdiri atas 4 word

Sumber: (Bruce Schneier, 1996)

### 2.9.2 Konsep Matematika pada Algoritme AES

Setiap byte dalam algoritme 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 algoritme AES dilakukan dengan menggunakan operasi XOR yang dilambangkan dengan “⊕”. Contoh berikut memiliki nilai yang sama antara satu dengan lainnya :

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

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

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

## 2. Perkalian (*Multiplication*)

Dalam representasi *polynomial*, perkalian dalam  $\text{GF}(2^8)$  (dinotasikan dengan “•”) berhubungan dengan perkalian *polynomial* modulo *irreducible polynomial* berpangkat 8. *Polynomial* dikatakan *irreducible* jika hanya dapat dibagi dengan 1 dan dengan dirinya sendiri. Untuk algoritme 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:

$$\begin{array}{r} 57 = 01010111 \\ 83 = 10000011 \\ \hline 01010111 \\ 01010111 \\ \hline 01010111 \oplus \\ 1010110111001 \\ \hline 100011011 \oplus \\ 100000011001 \\ \hline 100011011 \oplus \\ 11000001 \end{array} \Rightarrow c1$$

=> (jumlah bit lebih dari 8 maka di-xor-kan dengan 100011011)

=> (jumlah bit lebih dari 8 maka di-xor-kan dengan 100011011)

## 3. *Polynomial* dengan koefisien pada $\text{GF}(2^8)$

*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 *polynomial*  $c(x) = a(x) \cdot b(x)$  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:

$$c_0 = a_0 \cdot b_0$$

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

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

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

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

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

$$c_6 = a_3 \cdot b_3$$

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

### 2.9.3 Spesifikasi Algoritme

Pada algoritme 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, atau 8 yang menunjukkan jumlah *word* pada kunci. Jumlah putaran pada algoritme 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 2.1.

Tabel 2.2 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 algoritme 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 :  $\text{RotWord}(09cf4f3c) \Rightarrow (cf4f3c09)$

#### 3. Rcon[i]

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

$$\begin{aligned} \text{Contoh : } \quad \text{Rcon}[1] &= [02^{1-1}, \{00\}, \{00\}, \{00\}] \\ &= [02^0, \{00\}, \{00\}, \{00\}] \\ &= [01000000] \end{aligned}$$

Tabel 2.3 Rcon dengan I dimulai dari 1 sampai 10 dalam bentuk heksadesimal

<i>i</i>	<i>rc<sub>i</sub></i>
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  $Nb(Nr+1) - 1$ .

#### 2.9.4 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  $Nk \leq i < Nb(Nr+1)$ , carilah  $w[i]$  dengan ketentuan sebagai berikut :
  - jika  $(i \bmod Nk) \neq 0$  dan  $(i \bmod Nk) \neq 4$  maka  $w[i] = w[i-Nk] \oplus w[i-1]$ .
  - jika  $(i \bmod Nk) = 0$  maka  $w[i] = w[i-Nk] \square (\text{SubWord}(\text{RotWord}(w[i-1]))) \square Rcon[i/Nk]$
3.  $w[i] = w[i-Nk] \oplus (\text{SubWord}(\text{RotWord}(w[i-1])) \oplus Rcon[i/Nk])$ 
  - jika  $(i \bmod Nk) = 4$  maka  $w[i] = w[i-Nk] \square \text{Subword}(w[i-1])$   
Subkunci (*round key*) diperoleh dengan ketentuan:
  - Subkunci( $i$ ) = ( $w[i*4], w[i*4+1], w[i*4+2], w[i*4+3]$ ) ; untuk  $0 \leq i \leq Nr$

#### 2.9.5 Proses Enkripsi

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

1. *SubBytes()*

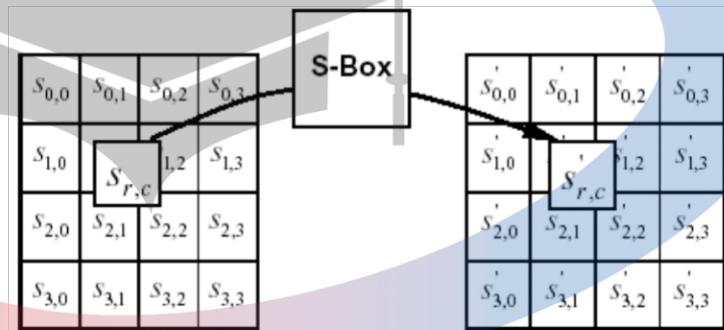
Transformasi *SubBytes()* merupakan substitusi *byte* yang beroperasi terhadap setiap *byte* pada *State* dengan menggunakan tabel substitusi (S-box).

Tabel 2.4 Tabel S-box

	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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 2.16.



Gambar 2.16 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}.

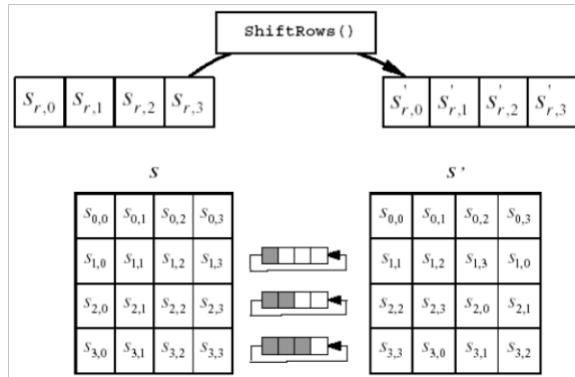
## 2. *ShiftRows()*

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()* seperti yang terlihat pada Gambar 2.17 berikut ini :

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

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 2.17 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.

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

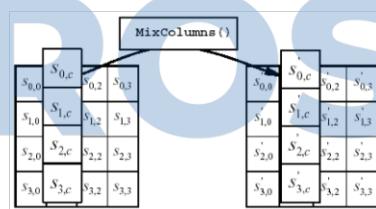
Hasil dari perkalian matriks di atas dapat dijabarkan sebagai berikut:

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

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

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

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



Gambar 2.18 Transformasi *MixColumns()* pada *State*

(FIPS 197, 2001)

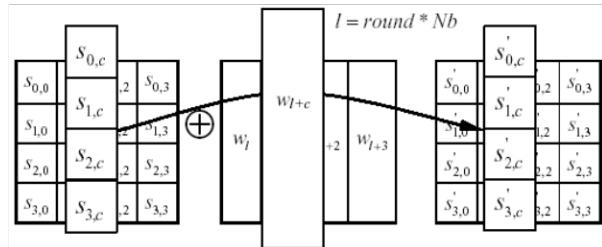
### 4. *AddRoundKey()*

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

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

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

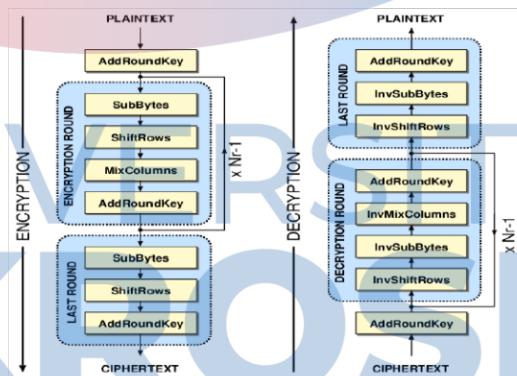
*round* = nilai yang berada pada range  $0 \leq round \leq Nr$



Gambar 2.19 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  $Nr-1$  putaran ( $1 \leq round < Nr$ ), 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 2.20.



Gambar 2.20 Proses Enkripsi dan Dekripsi Menggunakan AES

(Hameed, et al., 2011)

## 2.9.6 Proses Dekripsi

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

1. *InvShiftRows()*

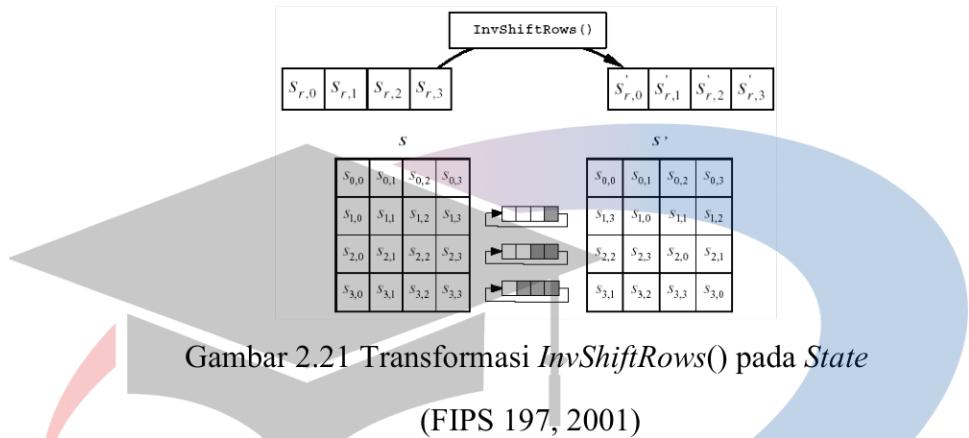
Transformasi ini merupakan kebalikan dari transformasi *ShiftRows()* pada proses enkripsi. Transformasi *InvShiftRows()* dilakukan pada 3 baris terakhir dengan melakukan geser

(*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:

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

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

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



## 2. *InvSubBytes()*

*InvSubBytes()* adalah kebalikan dari transformasi *SubBytes()*. Transformasi *InvSubBytes()* menggunakan S-box seperti pada tabel 2.4 :

Tabel 2.5 Tabel *Inverse S-box*

		y															
		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.

$$\begin{bmatrix} 0, c \\ 0, c \end{bmatrix} \begin{bmatrix} 0e & 0b \\ 0d & 09 \\ 09 & 0e \end{bmatrix} \begin{bmatrix} 0, c \\ 0, c \end{bmatrix}$$

Untuk  $0 \leq c < \mathbf{Nb}$

(11)

Hasil dari perkalian matriks di atas 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  $\mathbf{Nr}-1$  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*.