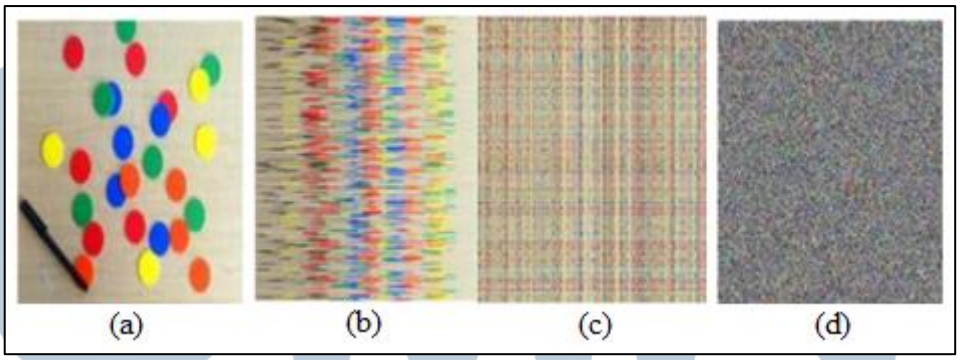
BAB II

LANDASAN TEORI

2.1 Enkripsi Chaotic

Xie dkk. (2016) memberi contoh enkripsi *chaotic* menggunakan sebuah persamaan untuk menghasilkan barisan angka yang seolah terlihat acak, kemudian data rahasia ditutupi dengan penambahan modulo. Setiap angka dalam baris tersebut ditambahkan ke data rahasia lalu dilakukan operasi modulo.

Enkripsi *chaotic* memanfaatkan barisan angka yang seolah terlihat acak. Namun enkripsi tidak hanya bisa dilakukan dengan penambahan modulo. Dalam enkripsi citra, barisan angka acak bisa digunakan untuk melakukan pengacakan piksel pada setiap baris dan/atau kolom. Şekertekin dan Atan (2016) memberi contoh hasil citra yang dienkripsi menggunakan kombinasi Ikeda *map* dan Hénon *map*.



Gambar 2.1 (a) Citra Asli; (b) Citra dengan Pengacakan pada Baris; (c) Citra dengan Pengacakan pada Kolom; (d) Citra Final yang Terenkripsi (Şekertekin dan Atan, 2016)

Gambar 2.1 (a) merupakan citra asli yang belum terenkripsi. Gambar 2.1 (b) merupakan citra yang posisi piksel-piksel pada setiap barisnya ditukar satu sama lain secara acak. Gambar 2.1 (c) merupakan hasil pada Gambar 2.1 (b) yang posisi

piksel-piksel pada setiap kolomnya ditukar satu sama lain secara acak. Gambar 2.1 (d) merupakan hasil pada Gambar 2.1 (c) yang warnanya diacak berdasarkan Hénon *map*. Gambar 2.1 yang dihasilkan dari percobaan Şekertekin dan Atan (2016) ini hanya merupakan salah satu contoh hasil teknik enkripsi menggunakan *chaotic map*.

2.2 Ikeda Map

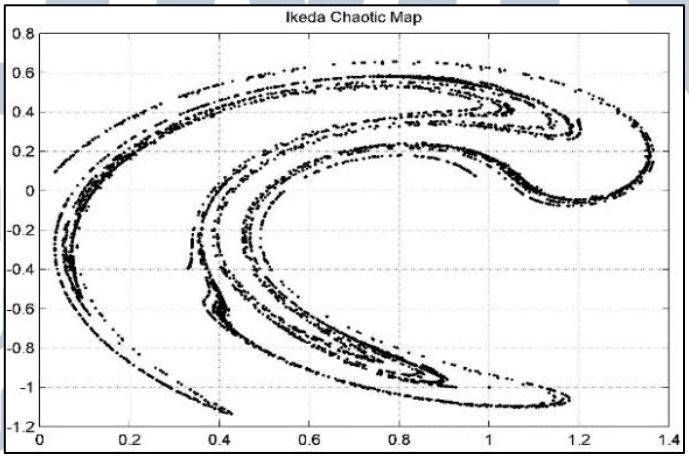
Untuk melakukan enkripsi *chaotic*, diperlukan *map* yang bersifat *chaotic*. Menurut Şekertekin dan Atan (2016), struktur yang kompleks dan sifat *chaotic* yang luar biasa menjadikan Ikeda *map* baik digunakan untuk enkripsi. Berikut ini adalah persamaan dari Ikeda *map*.

$$X_{n+1} = 1 + U(X_n \cos t_n - Y_n sin\ t_n) \quad (2.1)$$

$$Y_{n+1} = U(X_n \sin t_n + Y_n cos\ t_n) \quad (2.2)$$

$$t_n = 0.4 - 6/(1 + {X_n}^2 + {Y_n}^2) \quad (2.3)$$

Ikeda *map* akan *chaotic* pada saat kondisi $0{,}5 \leq U \leq 0{,}95$ terpenuhi. Ikeda *map* akan menghasilkan nilai X dan Y yang terlihat acak. Angka-angka tersebut akan digunakan untuk mengubah nilai warna awal dari setiap piksel yang ada pada citra rahasia. U akan digunakan sebagai kunci. Berikut ini adalah *map* yang dihasilkan dalam iterasi sebanyak 5000 kali dengan $X_0 = 1$, $Y_0 = -1$, dan $U = 0.8$.

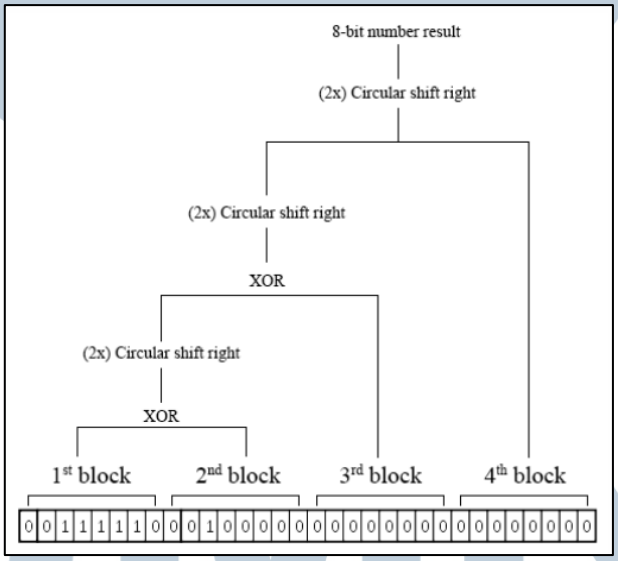


Gambar 2.2 Ikeda *Map* (Şekertekin dan Atan, 2016)

2.3 SCAC-MAT

SCAC-MAT (*Stream Cipher Algorithm with Ikeda Map Trajectories*) merupakan algoritma *stream cipher* baru yang mengunakan Ikeda *map* sebagai generator angka (Susanto, 2016). *Stream cipher* merupakan metode enkripsi yang kunci serta algoritmanya diterapkan ke setiap digit biner data (Rouse, 2005). Susanto (2016) menunjukkan bahwa angka-angka yang dihasilkan oleh Ikeda *map* dapat digunakan untuk mengubah data.

Nilai X dan Y yang didapat dari persamaan Ikeda *map* diubah menjadi *floating-point* 32 bit kemudian dikompres dengan algoritma seperti pada Gambar 2.3 berikut.

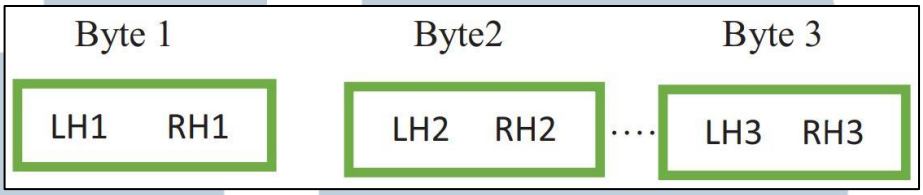


Gambar 2.3 Kompresi *Floating-Point* 32 Bit

Setelah algoritma pada Gambar 2.3 berjalan, bit-bit yang semula berjumlah 32 telah berubah menjadi berjumlah 8. Bit-bit tersebut akan mengalami operasi XOR terhadap warna piksel citra rahasia. Jika operasi XOR dilakukan terhadap warna piksel citra rahasia yang tidak terenkripsi, maka hasilnya adalah warna piksel citra yang terenkripsi. Jika operasi XOR dilakukan terhadap warna piksel citra rahasia yang terenkripsi, maka hasilnya dalah warna piksel citra yang terdekripsi.

2.4 Pemisahan Bit

Untuk setiap satu *byte* data, bit bisa dipecah menjadi dua sampai delapan bagian. Al-Bayati dan Al-Jarrah (2016) memberikan contoh pemisahan bit dengan memecah satu *byte* menjadi dua bagian, yaitu *Most Significant Bits* (MSB) dan *Least Significant Bits* (LSB).

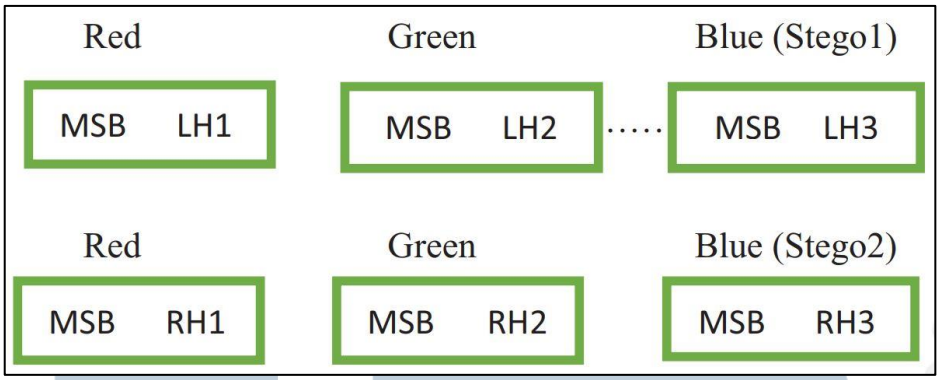


Gambar 2.4 Pemisahan Bit (Al-Bayati dan Al-Jarrah, 2016)

Pada Gambar 2.4, MSB dinamakan *Left Half-Byte* (LH) dan LSB dinamakan *Right Half-Byte* (RH). Dalam setiap *byte*, LH merupakan 4 bit di sebelah kiri dan RH merupakan 4 bit di sebelah kanan. Untuk setiap *byte* yang ada, LH dan RH dipisah.

2.5 Steganografi Berbasis LSB

Steganografi berbasis LSB merupakan teknik menyembunyikan data dengan mengganti LSB dari sebuah citra. Dengan memodifikasi bit paling kanan dari sebuah citra, data rahasia bisa dimasukkan tanpa menimbulkan perubahan yang terlihat jelas. Namun, jika data yang dimasukkan terlalu besar sehingga perlu memodifikasi lebih dari satu bit di sebelah kanan, maka perubahan citra akan semakin terlihat jelas (Izhar, 2018). Al-Bayati dan Al-Jarrah (2016) memberikan contoh pemasukkan bit citra rahasia dengan cara mengganti empat LSB citra pelindung dengan empat bit citra rahasia seperti pada gambar di bawah ini.



Gambar 2.5 Pemasukkan Bit Rahasia ke LSB Citra Pelindung (Al-Bayati dan Al-Jarrah, 2016)

Pada Gambar 2.5, setiap LH dan RH yang didapatkan berdasarkan Gambar 2.4 digunakan untuk mengganti setiap 4 bit terakhir dari citra pelindung.

2.6 Mean Square Error

Mean Square Error (MSE) adalah rata-rata perbedaan kuadrat setiap piksel antara citra pelindung yang tidak mengandung citra rahasia dan citra pelindung yang mengandung citra rahasia (Bhardwaj dan Khanna, 2015). MSE dapat digunakan untuk menguji kesamaan citra rahasia asli dengan citra rahasia yang dikembalikan dari citra-citra pelindung. Pradhan dkk. (2016) menyatakan bahwa dua buah citra yang sama akan memiliki nilai MSE 0. Berikut ini adalah persamaan MSE.

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m} \sum_{j=0}^{n} (p_{ij} - q_{ij})^2 \quad (2.4)$$

2.7 Peak Signal-to-Noise Ratio

Pradhan dkk. (2016) menyatakan bahwa *Peak Signal-to-Noise Ratio* (PSNR) adalah ukuran distorsi dari suatu citra. Semakin besar PSNR, semakin kecil distorsi. PSNR yang baik bernilai di atas 40 dB. PSNR yang bernilai di antara 30 dB sampai 40 dB masih dapat diterima. Jika sudah di bawah 30 dB, berarti distorsi sangat besar. Untuk mendapatkan PSNR, nilai MSE harus dihitung terlebih dahulu.

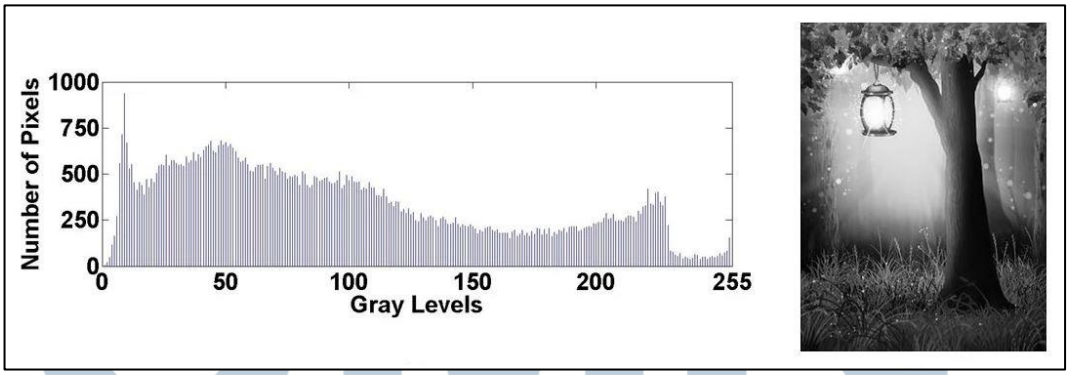
Berikut ini adalah persamaan PSNR yang digunakan oleh Pradhan dkk. (2016).

$$PSNR = 10 \times log_{10} \frac{255^2}{MSE} \quad (2.5)$$

m dan *n* merupakan tinggi dan lebar citra. p_{ij} merupakan piksel citra pelindung asli pada baris piksel i dan kolom piksel j, sedangkan q_{ij} merupakan piksel citra pelindung yang sudah mengandung citra rahasia. Angka 255 yang digunakan pada persamaan PSNR merupakan nilai warna maksimal dari sebuah piksel. Robinson (2018) mengatakan warna piksel yang tersusun atas 8 bit akan memiliki nilai maksimal 255.

2.8 Analisis Histogram Citra

Histogram citra adalah grafik dari intensitas piksel terhadap jumlah piksel. Sumbu x menandakan tingkat keabuan dan sumbu y menandakan jumlah piksel yang memiliki tingkat keabuan tersebut (Sneha, 2017).



Gambar 2.6 Contoh Histogram

Analisis histogram dapat digunakan untuk menguji enkripsi. Dengan histogram, penyerang dapat melakukan analisis frekuensi kemunculan intensitas piksel untuk menyimpulkan kunci rahasia atau piksel citra yang sebenarnya (Munir, 2012). Histogram yang ideal dari citra terenkripsi adalah datar. Histogram yang datar menandakan distribusi piksel yang seragam (Jia, 2010). Distribusi piksel yang

seragam tidak dapat memberikan petunjuk terhadap serangan analisis statistik dari citra terenkripsi (Li dkk., 2018).