**BAB II**

**LANDASAN TEORI**

**2.1. Data**

Menurut berbagai kamus Bahasa Inggris-Indonesia data dapat diartikan sebagai istilah yang berasal dari kata datum yang berarti bahan-bahan keterangan. Sedangkan menurut Gordon B. Davis dalam bukunya yang berjudul Management Information System: Conceptual Fundations, Structures, and Development menyebut data sebagai lambing-lambang tidak acak yang menunjukan jumlah atau tindakan. Dari beberapa gambaran diatas dapat kita simpulkan bahwa data merupakan bahan baku informasi, yang didefenisikan sebagai kelompok teratur dari symbol-simbol yang mewakili kuantitas, tindakan, benda, dan sebagainya. Data terbentuk dari karakter, dapat berupa symbol khusus seperti \*, $ dan /. Data disusun untuk diolah dalam bentuk struktur data, struktur file, dan basis data (Zakaria, 2018).

**2.2. Autentikasi**

Autentikasi (*Authentification*) adalah suatu proses yang merupakan sebuah tindakan pembuktian (validasi) terhadap identitas seorang pengguna pada saat akan memasuki (mengakses) sebuah sistem. Dimana proses validasi tersebut biasanya menggunakan nama dan password dari seorang pengguna yang nantinya akan dijadikan sebagai penanda (verifikasi) apakah seseorang itu adalah orang yang berhak untuk masuk kedalam sistem tersebut.

Adapun metode-metode autentikasi adalah sebagai berikut

1. *Something you know*

Ini adalah metode autentikasi yang paling umum. Cara ini mengandalkan kerahasiaan informasi, contohnya adalah password dan PIN. Cara ini berasumsi bahwa tidak ada seorangpun yang mengetahui rahasia itu kecuali anda seorang.

1. *Something you have*

Cara ini biasanya merupakan faktor tambahan untuk membuat autentikasi menjadi lebih aman. Cara ini mengandalkan barang yang sifatnya unik, contohnya adalah kartu *magnetic/smartcard, hardware token, USB token* dan sebagainya. Cara ini berasumsi bahwa tidak ada seorangpun yang memiliki barang tersebut kecuali anda seorang.

1. *Something you are*

Ini adalah metode yang paling jarang dipakai karena faktor teknologi dan manusia juga. Cara ini menghandalkan keunikan bagian-bagian tubuh anda yang tidak mungkin ada pada orang lain seperti sidik jari, suara atau sidik retina. Cara ini berasumsi bahwa bagian tubuh anda seperti sidik jari dan sidik retina, tidak mungkin sama dengan orang lain.

1. *Something you do*

Melibatkan bahwa setiap user dalam melakukan sesuatu dengan cara yang berbeda. Contoh: Penggunaan analisis suara (*voice recognation*), dan analisis tulisan tangan (Jakfar, 2018).

**2.3. *QR Code (Quick Respone Code)***

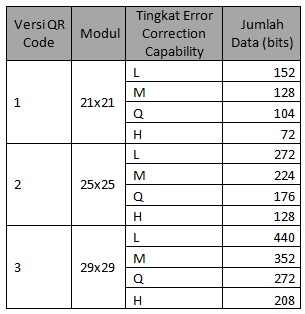
*QR Code (Quick Response Code)* adalah barcode dua dimensi yang dapat menyimpan data. *QR Code* dikembangkan oleh Denso Corporation, Jepang dan dapat digunakan secara gratis, bahkan untuk keperluan komersial. Berikut adalah contoh dari *QR Code*:



Gambar 2. 1 Contoh QR Code (Quick Respone Code)

Besaran data yang dapat disimpan bervariasi, tergantung pada versi *QR Code*, ukuran *QR Code* dan tingkat *Error Correction Capability*-nya.

Berikut tabel yang menunjukkan variasi besaran data yang dapat dimuat QR Code:



Gambar 2. 2 Besaran Data QR Code

*Error Correction Capability (ECC)* menunjukkan batasan *QR code* masih dapat terbaca ketika terjadi kerusakan pada *QR code* tersebut.*QR Code* dapat rusak ketika dicetak dan diletakkan pada kemasan produk dapat rusak.

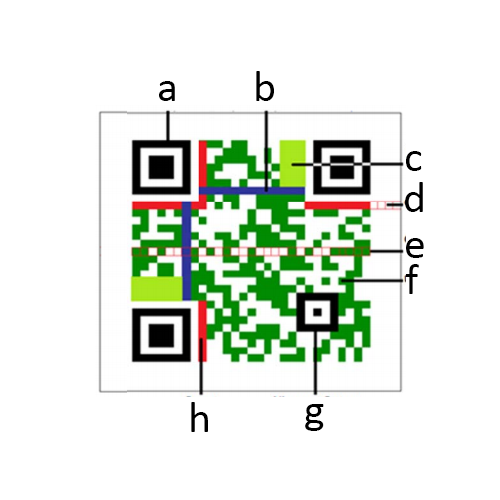
**2.3.1. Keuntungan dan Kerugian *QR Code***

Berikut adalah pertimbangan dalam implementasi QR Code. Keuntungan penggunaan *QR Code* antara lain:

1. Gratis dalam pembuatan dan penggunaannya
2. Tersedia QR Code scanner gratis
3. Menghemat kertas
4. Ukuran kecil
5. Tidak perlu membeli perangkat khusus scan QR Code
6. Sistem dapat cepat memberikan respon terkait hasil scan.

Sedangkankekurangan QR code antara lain:

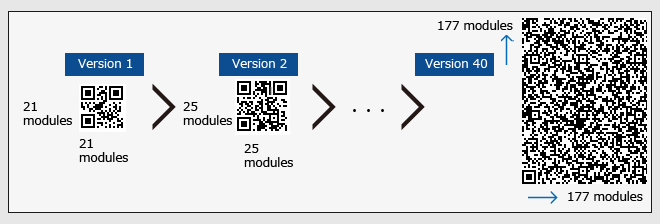
1. QR code hanya mudah diakses oleh pengguna smartphone.
2. Pengguna smartphone harus download aplikasi QR Code Scanner terlebih dahulu.
   * 1. **Anatomi *QR Code***



Gambar 2. 3 Anatomi QR Code (Sumber: qrcode.com)

Gambar di atas menyajikan struktur dari sebuah QR Code dan berikut istilah-istilah yang berkenaan dengan QR Code menurut Ariadi (2011):

1. Finding Pattern : Pola untuk mendeteksi posisi QR Code.
2. Timing Pattern : Pola yang digunakan untuk identifikasi koordinat pusat dari QR Code. Dibuat dalam bentuk modul hitam putih bergantian.
3. Version Information : Versi dari sebuah QR Code. Versi terkecil adalah 1 (21 x 21 modul) dan versi terbesar adalah 40 (177 x 177 modul).
4. Quiet Zone : Daerah kosong dibagian terluar QR Code yang mempermudah mengenali pengenal QR Code oleh sensor CCD.
5. QR Code Version : Versi QR Code. Pada contoh gambar versi yang digunakan adalah versi 3 (29 x 29 modul).
6. Data : Daerah tempat data tersimpan atau data dikodekan.
7. Alignment Pattern : Pola yang digunakan untuk memperbaiki penyimpanan QR Code terutama distorsi non linear.
8. Format Information : Informasi tentang *error correction level* dan *mark pattern.*
   * 1. **Versi *QR Code***



Gambar 2. 4 Versi QR Code (Sumber: qrcode.com)

Versi simbol QR Code terdiri dari versi 1 sampai versi 40. Setiap versi memiliki konfigurasi dan jumlah modul yang berbeda-beda. (Modul ini mengacu pada titik-titik hitam dan putih yang membentuk QR Code). Konfigurasi modul mengacu pada jumlah modul yang terkandung dalam simbol dari Versi 1 (21 × 21 modul) hingga Versi 40 (177 × 177 modul). Setiap nomor versi yang lebih tinggi berisi 4 modul tambahan di setiap sisi.

* + 1. **Macam-Macam QR Code**

Adapun macam-macam QR Code yaitu sebagai berikut:

1. *QR Code* model 1 dan model 2



Gambar 2. 5 QR Code Model 1 (Sumber: qrcode.com)

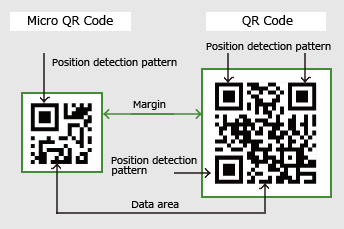
Model 1 adalah *Qr Code* asli, dapat menampung 1.167 angka dengan versi maksimum 14 (73 x 73 modul) (qrcode.com, 2013).



Gambar 2. 6 QR Code Model 2 (Sumber: qrcode.com)

Model 2 adalah penyempurnaan dari model 1 dengan versi terbesar 40 (177 x 177 modules), yang mampu menyimpan sampai 7.089 angka (qrcode.com, 2013).

1. *Micro QR Code*



Gambar 2. 7 Micro QR Code (Sumber: qrcode.com)

Versi terbesar dari kode ini adalah M4 (17x17 modul) yang dapat menyimpan hingga 35 angka. Fitur utama dari Micro Qr Code adalah hanya memiliki satu pola deteksi posisi, dibandingkan dengan regular QR Code yang memerlukan sejumlah tempat karena pola deteksi posisi yang terletak di tiga sudut simbol. Qr Code biasa membutuhkan setidaknya empat modul yang lebar di sekitar simbol, sedangkan Micro QR Code hanya membutuhkan cukup dua modul margin. Konfigurasi Micro Qr Code memungkinkan pencetakan di tempat lebih kecil dari Qr Code (qrcode.com,2013).

1. *iQr Code*



Gambar 2. 8 iQR Code (Sumber: qrcode.com)

Kode yang dapat dihasilkan dari salah satu modul, persegi atau persegi panjang. Dan dapat di cetak sebagai kode inversi hitam putih atau kode pola dot (bagian penanda). Versi terbesar dari kode ini dapat mencapai 61 (422x422 modul), yang dapat menyimpan 40.000 angka (qrcode.com, 2013).

1. *SQRC*



Gambar 2. 9 SQRC (Sumber: qrcode.com)

Jenis Qr Code ini dilengkapi dengan membaca fungsi pembatas. Ini dapat digunakan untuk menyimpan informasi pribadi untuk mengelola informasi internal perusahaan dan sejenisnya (qrcode.com, 2013).

1. *Frame QR / LogoQ*



Gambar 2. 10 Frame QR (Sumber: qrcode.com)

Jenis Qr Code yang dapat menggabungkan fitur desain tingkat tinggi seperti ilustrasi, huruf dan logo. Qr Code ini menggunakan Logika Since proprietary (qrcode.com,2013).

* 1. **Kriptografi**

Kriptografi adalah ilmu mengenai teknik enkripsi dimana “naskah asli” (plaintext) diacak menggunakan suatu kunci enkripsi menjadi “naskah acak yang sulit dibaca” (ciphertext) oleh seseorang yang tidak memiliki kunci dekripsi. Dekripsi menggunakan kunci dekripsi bisa mendapatkan kembali data asli. Probabilitas mendapat kembali naskah asli oleh seseorang yang tidak mempunyai kunci dekripsi dalam waktu yang tidak terlalu lama adalah sangat kecil.

Teknik enkripsi yang digunakan dalam kriptografi klasik adalah enkripsi simetris dimana kunci dekripsi sama dengan kunci enkripsi. Untuk public key cryptography, diperlukan teknik enkripsi asimetris dimana kunci dekripsi tidak sama dengan kunci enkripsi. Enkripsi, dekripsi dan pembuatan kunci untuk teknik enkripsi asimetris memerlukan komputasi yang lebih intensif dibandingkan enkripsi simetris, karena enkripsi asimetris menggunakan bilangan – bilangan yang sangat besar. (Kromodimoeljo, 2010).

**2.4.1. Sejarah Kriptografi**

Kriptografi sudah digunakan jauh sejak zaman dahulu. Penerapan kriptografi yang pertama kali (yang sudah ditemukan) adalah *hieroglyphics* yang diterapkan oleh bangsa mesir kuno sejak 3000 tahun sebelum masehi. Selanjutnya pada 400 SM bangsa spartan di Yunani juga menerapkan kriptografi di bidang militer. Mereka menggunakan alat yang disebut dengan *scytale*, yakni pita panjang terbuat dari bahan *papyrus*, cara membaca pesannya yaitu dengan menggulungkan pita tadi pada batang silinder. Bangsa Cina dan Jepang mulai mengenal kriptografi pada abad ke 15 M

Peradaban islam juga menggunakan ilmu kriptografi ini. Ilmuwan muslim yang bernama lengkap Abu Yusuf Ya’qub ibn ‘Ishaq as-Shabbah al Kindi ini memaparkan tentang kriptoanalisis dengan bukunya yang berjudul Risalah fi Istikhraj al-Mu’amma (Manuskrip untuk memecahkan pesan-pesan Kriptografi). Beliau terinspirasi dari keindahan Al-Qur’an sehingga beliaulah yang menemukan teknik analisis frekuensi dalam ilmu kriptografi, yakni teknik untuk memecahkan ciphertext berdasarkan frekuensi kemunculan karakter pada sebuah pesan.

* + 1. **Tujuan Kriptografi**

Ada empat tujuan dasar dalam menggunakan kriptografi yaitu:

1. Menjaga kerahasiaan agar informasi yang tersedia tidak dapat di akses atau tidak dapat diketahui oleh pihak yang tidak memiliki otoritas untuk mengetahui informasi yang telah di isikan sandi.
2. Menjaga keutuhan informasi agar pada saat informasi di transmisikan tidak terjadi perubahan oleh pihak yang tidak memiliki otoritas untuk mengubahnya. Contohnya untuk menghindari pinyisipan, penghapusan dan pensubsitusian datalain kedalam data aslinya.
3. Memasitikan identitas antar dua pihak yang saling berkomunikasi dan memastikan keaslian dari isi data, waktu pengirimanm, dll.
4. Mencegah penyangkalan bahwa data yang dikirimkan memang berasal dari pengirim yang benar yang telah sesuai dengan perjanjian.
   * 1. **Pembagian Algoritma Kriptografi**

Berdasarkan kunci pemecahannya algoritma kriptografi juga dibagi dalam dua jenis

1. Kriptografi Simetri

Algoritma kriptografi ini menggunakan kunci pemecahan yang sama saat enkripsi maupun dekripsi. Sehingga algoritma ini sering disebut dengan algoritma kunci tunggal.

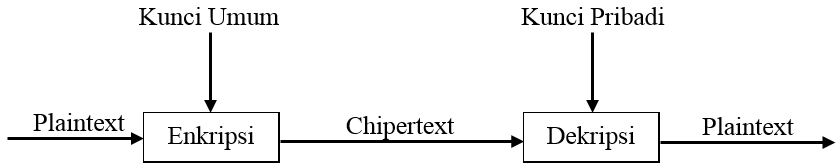


Gambar 2. 11. Proses Enkripsi Deskripsi Menggunakan Kriptografi Simetri

Kunci simetri meliputi enkripsi yang menggunakan algoritma Tiny Encryption Algorithm (TEA), OTP, Data Encryption Standard DES, Rivest Code 4 (RC4), Twofish, Rinjdael, AES dan Blowfish.

1. Kriptografi Asimetri

Algoritma ini menerapkan kunci yang berbeda saat proses enkripsi maupun dekripsi. Misal saat proses enkripsi menggunakan kunci A, namun saat proses dekripsinya kita diharuskan untuk menggunakan kunci B. Kunci A dalam algoritma ini sering disebut dengan kunci publik. Sedangkan kunci B disebut kunci rahasia (private key).



Gambar 2. 12. Proses Enkripsi Deskripsi Menggunakan Kriptografi Asimetri

Algoritma kriptografi asimetri meliputi algoritma ECC, LUC, RSA, El Gamal, DH, DSA dan lain sebagainya.

* 1. **Algoritma RC4**

RC4 merupakan jenis aliran kode yang berarti operasi enkripsinya dilakukan per karakter 1byte untuk sekali operasi. Algoritma kriptografi *Rivest Code 4 (RC4)* merupakan salah satu algoritma kunci simetris dibuat oleh RSA DataSecurity Inc (RSADSI) yang berbentuk stream cipher. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA (merupakan singkatan dari tiga nama penemu yaitu Rivest, Shamir, dan Adleman) (Ariyus, 2008).

Secara garis besar algoritma dari metode RC4 StreamCipherini terbagi menjadi dua bagian, yaitu Key Setup atau Key Schedulling Algorithm (KSA) dan Stream Generation atau Pseudo Random Generation Algorithm (PRGA) dan proses XOR dengan stream data.

RC4 menghasilkan *pseudorandom stream bit*. Seperti halnya stream cipher lainnya, algoritma RC4 ini dapat digunakan untuk mengenkripsi dengan mengombinasikannya dengan plainteks dengan menggunakan *bit-wise* XOR (Exclusive-or). Proses dekripsinya dilakukan dengan cara yang sama (karena XOR merupakan fungsi simetrik). Untuk menghasilkan *key-stream*, cipher menggunakan *state internal* yang meliputi dua bagian :

1. Sebuah permutasi dari 256 kemungkinan *byte*.

2. 2 Indeks-pointer 8-bit.

Permutasi diinisialisasi dengan sebuah variabel panjang kunci, biasanya antara 40 sampai 256 bit dengan menggunakan algoritma *key-scheduling* (KSA). Setelah proses ini selesai, *stream* yang terdiri dari sekumpulan bit tersebut terbentuk dengan menggunakan Pseudo-Random Generation Algorithm (PRGA). Berikut ini akan dijelaskan tentang kedua algoritma tersebut.

1. ***Key-Scheduling Algorithm* (KSA)**

*Algoritma key scheduling* digunakan untuk menginisialisasi permutasi di *array* “S”. panjang kunci didefinisikan sebagai jumlah *byte* di kunci dan mempunyai rentang panjang kunci dari 1 sampai 256, khususnya antara 5-16 tergantung dari panjang kunci 40-128 bit. Pertama-tama *array* “S” diinisialisasi untuk identitas permutasi. S kemudian diproses ke 256 iterasi dengan cara yang sama dengan PRGA utama, tapi juga dikombinasikan dalam byte dari kunci dalam waktu yang bersamaan. Berikut adalah algoritma KSA :

**for** i **from** 0 **to** 255

S[i] := i

**endfor**

j := 0

**for** i **from** 0 to 255

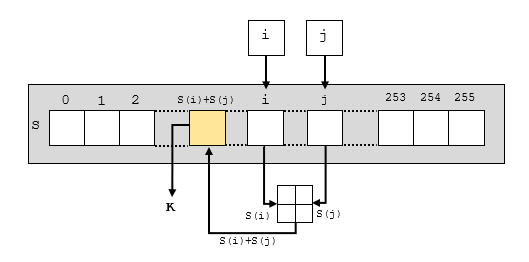
j := (j + S[i] + key[i mod keylength]) mod 256

*swap* values of S[i] and S[j]

**endfor**

1. **Pseudo-Random Generation Algorithm (PRGA)**

PRGA (Pseudo-Random Generation Algortihm) memodifikasi *state* dan *output* sebuah *byte* dari *key-stream*. Hal ini penting karena banyaknya dibutuhkan iterasi. Dalam setiap iterasi, PRGA menginkremen i, menambahkan nilai S yang ditunjuk oleh i sampai j, kemudian menukar nilai S[i] dan S[j], lalu mengembalikan elemen dari S di lokasi S[i] + S[j] (modulo 256). Setiap elemen S ditukar dengan elemen lainnya paling tidak satu kali setiap 256 iterasi.



Gambar 2. 13. Proses Pseudo Random Pada Algoritma RC4

Realisasi dari algoritma PRGA yaitu :

i := 0

j := 0

**while** GeneratingOutput:

i := (i + 1) mod 256

j := (j + S[i]) mod 256

*swap* values of S[i] and S[j]

K := S[(S[i] + S[j]) mod 256]

output K

**endwhile**

**2.5.1 Cara Kerja Algoritma RC4 Stream Chiper**

Cara kerja algoritma RC4 yaitu inisialisasi *S-Box* pertama, S[0], S[1],…,S[255], dengan bilangan 0 sampai 255. Pertama isi secara berurutan S[0]=0, S[1]=1,…,S[255]=255. Kemudian inisialisasi *array* lain (S-Box lain), misal *array* K dengan panjang 256. Isi *array* K dengan kunci diulangi sampai seluruh *array* K[0], K[1],….., K[255] terisi seluruhnya. Setelah itu menyimpan key dalam *Key* *Byte Array*, Permutasi pada *S-Box*. Pada *Stream Generation* akan menghasilkan nilai *pseudorandom* yang akan dikenakan operasi XOR untuk menghasilkan *ciphertext* ataupun sebaliknya yaitu untuk menghasilkan *plaintext*. Berikut cara kerja dari algoritma RC4 dengan menggunakan 4-bit kunci.

Array S : 0 1 2 3

Array K : 2 5 7 3

Inisiasi i dan j dengan 0, kemudian dilakukan KSA agar tercipta *state-array* yang acak. Penjelasan lebih lanjut adalah sebagai berikut.

**Iterasi 1**

i = 0

j = (0 + S[0] + K [0 mod 4]) mod 4 = (0 + 0 +2) mod 4 = 2

*swap* (S[0], S[2])

hasil *array* S : 2 1 0 3

**Iterasi 2**

i = 1

j = (2 + S[1] + K [1 mod 4]) mod 4 = (2 + 1 + 5) mod 4 = 0

*swap* (S[1], S[0])

hasil *array* S : 1 2 0 3

**Iterasi 3**

i = 2

j = (0 + S[21] + K [2 mod 4]) mod 4 = (0 + 0 + 7) mod 4 = 3

*swap* (S[2], S[3])

hasil *array* S : 1 2 3 0

**Iterasi 4**

i = 3

j = (3 + S[3] + K [3 mod 4]) mod 4 = (3 + 0 + 3) mod 4 = 2

*swap* (S[3], S[2])

hasil *array* S : 1 2 0 3

Setelah melakukan KSA, akan dilakukan PRGA. PRGA akan dilakukan sebanyak 4 kali dikarenakan plainteks yang akan dienkripsi berjumlah 4 karakter. Hal ini disebakan karena dibutuhkan 1 kunci dan 1 kali pengoperasian XOR untuk tiapiap karakter pada plainteks. Berikut adalah tahapan penghasilan kunci enkripsi dengan PRGA.

*Araay* S : 1 2 0 3

Inisialisasi

i = 0

j = 0

**Iterasi 1**

i = (0 + 1) mod 4 = 1

j = (0 + S[1]) mod 4 = (0 + 2) mod 4 = 2

*swap* (S[1], S[2])

1 0 2 3

K1 = S[(S[1] + S[2]) mod 4] = S[2 mod 4] = 2

K1 = 00000010

**Iterasi 2**

i = (1 + 1) mod 4 = 2

j = (2 + S[2]) mod 4 = (2 + 2) mod 4 = 0

*swap* (S[2], S[0])

2 0 1 3

K2 = S[(S[2] + S[0]) mod 4] = S[3 mod 4] = 3

K2 = 00000011

**Iterasi 3**

i = (2 + 1) mod 4 = 3

j = (0 + S[3]) mod 4 = (0+ 3) mod 4 = 3

*swap* (S[3], S[3])

1 0 2 3

K3 = S[(S[3] + S[3]) mod 4] = S[6 mod 4] = 2

K3 = 00000010

**Iterasi 4**

i = (3 + 1) mod 4 = 0

j = (3 + S[0]) mod 4 = (3+ 1) mod 4 = 0

*swap* (S[0], S[0])

1 0 2 3

K1 = S[(S[0] + S[0]) mod 4] = S[2 mod 4] = 2

K1 = 00000010

Berikut adalah tahapan penghasilan kunci enkripsi dengan PRGA.

Array S :1 2 0 3

Inisialisasi

i = 0

j = 0

**Iterasi 1**

i = (0 + 1) mod 4 = 1

j = (0 + S[1]) mod 4 = (0 + 2) mod 4 = 2

*swap* (S[1], S[2])

1 0 2 3

K1 = S[(S[1] + S[2]) mod 4] = S[2 mod 4] = 2

K1 = 00000010

**Iterasi 2**

i = (1 + 1) mod 4 = 2

j = (2 + S[2]) mod 4 = (2 + 2) mod 4 = 0

*swap* (S[2], S[0])

2 0 1 3

K2 = S[(S[2] +S[0]) mod 4] = S[3 mod 4] = 3

K2 = 00000011

**Iterasi 3**

i = (2 + 1) mod 4 = 3

j = (0 + S[3]) mod 4 = (0 + 3) mod 4 = 3

*swap* (S[3], S[3])

2 0 1 3

K3 = S[(S[3] + S[3]) mod 4] = S[6 mod 4] = 1

K3 = 00000001

**Iterasi 4**

i = (3 + 1) mod 4 = 0

j = (3 + S[0]) mod 4 = (3 + 2) mod 4 = 1

*swap* (S[0],S[1])

0 2 1 3

K4 = S[(S[0] + S[1]) mod 4] = S[2 mod 4] = 1

K4 = 00000001

Proses XOR kunci enkripsi dengan plainteks

H A L O : 01001000 01000001 01001100 01001111

Key : 00000010 00000011 00000001 00000001

Chiperteks : 01001010 01000010 01001101 01001110

(J) (B) (M) (N)

* 1. **PHP**

PHP Kepanjangan dari PHP adalah "*Hypertext Preprocessor*" (ini merupakan singkatan rekursif).PHP adalah bahasa scriptingweb *HTML-embedded*. Ini berarti kode PHP dapat disisipkan ke dalam HTML halaman Web. Ketika sebuah halaman PHP diakses, kode PHP dibaca atau"diurai" oleh *server*. *Output* dari fungsi PHP pada halaman biasanya dikembalikan sebagai kode HTML, yang dapat dibaca oleh *browser*. Karena kode PHP diubah menjadi HTML sebelum halaman dibuka, pengguna tidak dapat melihat kode PHP pada halaman. Ini membuat halaman PHP cukup aman untuk mengakses database dan informasi aman lainnya. (Ferdianto, 2013)

Banyak sintaks PHP yang hasil adaptasi dari bahasa lain seperti bahasa C, Java dan Perl. Namun, PHP memiliki sejumlah fitur unik dan fungsi tertentu juga. Tujuan dari bahasa pemrograman PHP adalah untuk memungkinkan pengemban web untuk menulis halaman yang dihasilkan secara dinamis dengan cepat dan mudah. PHP juga bagus untuk menciptakan situs Web *database-driven*. Jika Anda ingin mempelajari lebih lanjut tentang PHP, situs resminya yaitu PHP.net. (Ferdianto, 2013)

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukansebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulaiapache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudahkarena memiliki referensi yang banyak.
5. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara *runtime*melalui *console* serta juga dapat menjalankan perintah-perintah system. (Ferdianto, 2013).
   1. ***Hypertext Markup Language* (HTML)**

HTML adalah sebuah bahasa pemrograman yang berbentuk skrip-skrip yang berguna untuk membuat sebuah halaman *web*. HTML dapat dibaca oleh berbagai *platform* seperti: Windows, Linux, Macintosh. Kata *Markup Language* pada HTML menunjukkan fasilitas yang berupa tanda tertentu dalam skrip HTML dimana pengguna bisa mengatur judul, garis, tabel, gambar, dan lain-lain dengan perintah yang telah ditentukan pada elemen HTML. HTML sendiri dikeluarkan oleh W3C (*Word Wide Web Consortium*), setiap terjadi perkembangan level HTML harus dievakuasi ketat dan disetujui oleh W3C. Contoh tag HTML antara lain: <head>, <body> dan <table> (Kadir, 2002).

* 1. ***Cascading Style Sheet* (CSS)**

*Cascading Style Sheets* (CSS) adalah salah satu bahasa pemrograman desain *web* (*style sheetlanguage*) yang mengontrol format tampilan sebuah halaman *web* yang ditulis dengan menggunakan bahasa penanda (*markup language*). Biasanya CSS digunakan untuk mendesain sebuah malam HTML dan XHTML, tetapi sekarang bahasa pemrograman CSS bisa diaplikasikan untuk segala dokumen *XML*,termasuk SVG dan XUL. CSS dibuat untuk memisahkan kontek utama(biasanya dibuat dengan menggunakan bahasa HTMLdan sejenisnya) dengan tampilan dokumen yangmeliputi *layout*, warna dan *font*. Pemisahan ini dapat meningkatkan daya akses konten pada *web*, menyediakan lebih banyak fleksibilitas dan control dalam spesifikasi dari sebuah karakteristik dari sebuah tampilan, memungkinkan untuk membagi banyak halaman untuk sebuah formating dan mengurangi kerumitan dalam penulisan kode dan struktur dari konten, contohnya teknik tables pada layout desain *web* (*layout* tanpa tabel)

*Style Sheet* adalah sebuah *text file*yang sederhana (dimana berekstensi \*.css), ditulis menurut aturan bahasa yang dipaparkan pada rekomendasi CSS1 atau CSS2. Cara kerja CSS dengan menggunakan dua buah elemen penting untuk pemformatan tampilannya, diantaranya selektor dan deklarator. Dua buah elemen ini berfungsi sebagai penentu format tampilan dan lainnya menempatkan format tampilan tersebut. Deklarator berisi beberapa perintah-perintah CSS untuk menentukan format dari sebuah elemen pada halaman *web*. Sedangkan selektor adalah sebuah perintah lanjut dari deklarator dan berfungsi menempatkan format tampilan dari deklarator.

Dalam *Cascading Style Sheets* ada dua cara menghubungkan sebuah dokumen HTML dengan CSS

*1. Selector class*

*2. Selector ID*

*3. Selector Descendant* (turunan)

*4. Selector Link Pseudo Class*

*5. Selector Pseudo elemen*

*6. Selector dynamic pseudo class*

*7. Selector languange*

*8. Selector child*

*9. Selector first-child*

10. *Selector adjacent* (berdekatan)

* 1. **Android**

*Android* adalah sistem operasi yang dikeluarkan oleh Google. Android dibuat khusus untuk smartphone dan tablet. Berbagai macam produsen telah menggunakan Android sebagai sistem operasi untuk peranti (device) yang mereka produksi. Android juga mempunyai store dengan lebih dari 2 miliar pengguna aktif, per Januari 2018 ketika tulisan ini dibuat.

Sejak dirilis tahun 2008, Google telah mengeluarkan beberapa versi, dengan "Pie" sebagai versi yang terbaru.

Pada tahun 2013, Android menjadi operation system (OS) terlaris pada tablet dan smartphone. Tercatat pada tahun 2016, store Android memiliki lebih dari 2.8 juta aplikasi.

Android menarik bagi perusahaan teknologi yang membutuhkan barang siap jadi, biaya rendah dan kustomisasi OS untuk perangkat teknologi tinggi mereka. Hal ini menjadi daya tarik bagi banyak perusahaan, sehingga mereka memilih Android.

Source code dari Android bersifat open source. Ini adalah hal menarik bagi komunitas developer, karena lisensi open source sangat mendukung untuk mengembangkan produknya dengan aman.

Versi-versi android antara lain :

* 1. Android versi 1.1
  2. Android versi 1.5 (Cupcake)
  3. Android versi 1.6 (Donut)
  4. Android versi 2.0 / 2.1 (Eclair)
  5. Android versi 2.2 Froyo (Frozen Yoghurt)
  6. Android versi 2.3 (Gingerbread)
  7. Android versi 3.0/3.1 (Honeycomb)
  8. Android versi 4.0 ICS (Ice Cream Sandwich)
  9. Android versi 4.1 – 4.3 (Jelly Bean)
  10. **Android versi 4.4 (Kitkat)**
  11. **Android versi 5.0 – 5.1 (Lollipop)**
  12. **Android versi 6.0 (Marshmallow)**
  13. **Android versi 7.0 (Nougat)**
  14. **Android versi 8.0 / 8.1 (Oreo)**
  15. **Android versi 9.0 (Pie)**
  16. ***MySQL***

*MySQL* adalah aplikasi *SQL database server* yang multi *user*. Oleh karena itu, *MySQL* digunakan dalam pembuatan aplikasi ini yang digunakan sebagai *database server* untuk menyimpan data lokasi yang dikirim oleh masing-masing telepon seluler. Dalam *MYSQL* ada beberapa operasi dasar yang lebih dikenal dengan operasi CRUD yaitu :

1. *Create*

*Create* adalah operasi penambahan data baru ke dalam table. Terdapat 2 *Query* untuk menambah data, yang pertama adalah penambahan yang tidak menspesifikasikan nama kolom yang akan ditambahkan dan hanya memberikan isi dari tabelnya. *Query*nya adalah sebagai berikut:

INSERT INTO *table\_name*

VALUES (*value1*, *value2*, *value3*,...);

Kedua adalah menginputkan data dengan mencantumkan nama kolom yang akan diisikan dengan isi di dalamnya. *Query*nya adalah sebagai berikut:

INSERT INTO *table\_name* (*column1*,*column2*,*column3*,...)

VALUES (*value1*, *value2*, *value3*,...);

1. *Read*

*Read* adalah operasi untuk menampilkan semua atau sebagian data yang berada di dalam *database*. *Query*nya adalah sebagai berikut:

SELECT *column\_name*, *column\_name*

FROM *table\_name*;

1. *Update*

*Update* adalah operasi untuk mengubah data yang ada di dalam *database*. *Query*nya adalah sebagai berikut:

UPDATE *table\_name*

SET *column1*=*value1*, *column2*=*value2*,...

WHERE *some\_column*=*some\_value*;

1. *Delete*

*Delete* adalah operasi untuk menghapus data yang ada di dalam table. *Query*nya adalah sebagai berikut:

DELETE FROM *table\_name*

WHERE *some\_column* = *some\_value*;

**Tabel 2.1 Tipe Data *Database* *MySQL***

|  |  |  |
| --- | --- | --- |
| **No.** | **Type** | **Keterangan** |
| 1. | *Varchar* | Tipe data karakter yang panjangnya tidak tetap |
| 2. | TINYINT | Adalah tipe data bilangan bulat yang rentangnya  -128 sampai 127 atau menggunakan atribut unsigned dari 0 hingga 255 |
| *3.* | *Text* | Tipe data yang dapat menampung semua tipe data. |
| *4.* | *Date* | Tipe data yang digunakan untuk mendiskripsikan tanggal. |
| *5.* | *Smallint* | Adalah tipe data bilangan bulat yang rentangnya  –32768 sampai 32767. The unsigned range is 0 to 65535 |
| *6.* | *Mediumint* | Adalah tipe data bilangan bulat yang rentangnya  –8388608 to 8388607. unsigned range-nya  0 sampai 16777215 |
| *7.* | *Int* | Tipe data yang bernilai integer/bilangan bulat. |
| *8.* | *Time* | Tipe data waktu. Jangkauannya adalah '-838:59:59' hingga '838:59:59'. MySQL menampilkan TIME dalam format 'HH:MM:SS'. |
| *9.* | *Char* | Tipe data untuk menampung data yang bertipe karakter |
| 10. | *Primary Key* | Kunci primer adalah suatu atribut atau satu set minimal atribut yang tidak hanya mendefinisikan secara unik suatu kejadian spesifik tetapi juga dapat mewakili setiap kejadian dari suatu kejadian |

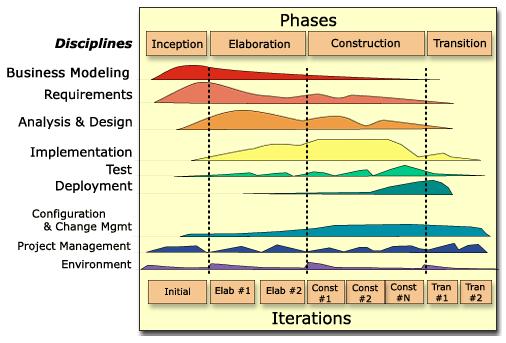
* 1. ***Rational Unified Process* (RUP)**

*Rational Unified Process* (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perankat lunak. Gambar dibawah menunjukkan secara keseluruhan arsitektur yang dimiliki RUP.

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui Gambar 2.5 dapat dilihat bahwa RUP memiliki 2 dimensi yaitu:

1.Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari fase selanjutnya. Setiap fase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.

2. Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing*, *what*, *how* dan *when*. Dimensi ini terdiri atas *Business Modeling, Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration* dan *Change Manegement, Project Management, Environtment.*



Gambar 2. 14. Arsitektur Rational Unified Process

Dalam metode ini, terdapat empat tahap pengembangan perangkat lunak yaitu:

1. *Inception*

Pada tahap ini pengembang mendefinisikan batasan kegiatan, melakukan analisis kebutuhan user, dan melakukan perancangan awal perangkat lunak (perancangan arsitektural dan *use case*). Pada akhir fase ini, prototipe perangkat lunak versi *Alpha* harus sudah dirilis

1. *Elaboration*

Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikasikan fitur perangkat lunak hingga perilisan prototipe versi *Betha* dari perangkat lunak.

1. *Construction*

Pengimplementasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta  dokumentasi perangkat lunak.

1. *Transition*

Instalasi *deployment* dan sosialisasi perangkat lunak dilakukan pada tahap ini. Pada tahapan ini dilakukan tahap evaluasi dari sistem yang telah direncanakan, dianalisa dan dibangun. Dari tahap ini bisa dilihat kekurangan dari sistem yang telah dibangun.

* 1. ***Flowchart***

*Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung. Jadi, setiap simbol *flowchart* melambangkan pekerjaan dan instruksinya. Simbol-simbol *flowchart* adalah standar yang ditentukan oleh *Amerika National Standard Institute Inc*.

Simbol–simbol yang digunakan dalam *flowchart* ditampilkan pada Tabel 2.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabel 2.2. Simbol-simbol *Flowchart*** | | | | |
| **NO.** | **SIMBOL** | **NAMA** | **KETERANGAN** |
| 1. |  | Proses | Mempresentasikan operasi. |
| 2. |  | *Input / Output* | Mempresentasikan *Input* data atau *Output* data yang diproses atau informasi. |
| 3. |  | Keputusan | Keputusan dalam program. |
| 4. |  | Dokumen | Dokument I / O dalam format cetak. |
| 5. |  | *Terminal points* | Awal / akhir *flowchart*. |
| 6. |  | *Preparation* | Pemberian harga awal. |

* 1. ***Unified Modeling Language* (UML)**

*Unified Modeling Language* (UML) merupakan pengganti dari metode analisis berorientasi objek dan desain berorientasi objek (OOA&D) yang dimunculkan sepenggunar akhir tahun 80-an dan awal tahun 90-an. UML merupakan gabungan dari metode Booch, Rumbaugh (OMT) dan Jacobson. Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. UML merupakan bahasa standar untuk penulisan *blue print software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak (Shalahudin, 2011).

* 1. **Diagram Dalam UML**

Dalam UML terdapat diagram-diagram yang bisa menggambarkan bagian atau aspek tertentu dari sebuah sistem. Sehingga dapat terlihat jelas alur dan gambaran umum dari perangkat lunak yang dibangun. Ada beberapa jenis diagram dalam UML yaitu:

1. *Use case* *Diagram*

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapatjuga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem (Shalahudin, 2011).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.3 Simbol *Use Case Diagram*** | | | |
| **NO** | **SIMBOL** | **NAMA** | **KETERANGAN** |
| 1. |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case* |
| 2. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (dependent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent) |
| 3. |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*) |
| 4. |  | *Include* | Menspesifikasikan bahwa *use case* sumber secara eksplisit |
| 5. |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan |
| 6. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7. |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas |
| 8. |  | *Use case* | Deskripsi dari uraian aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu actor |
| 9. |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi) |
| 10. |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

1. *Class Diagram*

Menggambarkan struktur statis *class* di dalam sistem. *Clas*s merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram* (Rosa dan Shalahudin, 2011).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.4 Simbol *Class Diagram*** | | | |
| **NO** | **SIMBOL** | **NAMA** | **KETERANGAN** |
| 1. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |
| 2. |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek |
| 3. |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4. |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu actor |
| 5. |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek |
| 6. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri |

1. *Sequence Diagram*

Menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksiantara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem (Shalahudin, 2011).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.5 Simbol *Sequence Diagram*** | | | |
| **NO** | **SIMBOL** | **NAMA** | **KETERANGAN** |
| 1 |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |
| 4 |  | *Message* | Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy* |

1. *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi (Shalahudin, 2011).

**Tabel 2.6 Simbol *Activity Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| **1** |  | *Actifity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| **2** |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi |
| **3** |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| **4** |  | *Actifity Final Node* | Bagaimana objek dibentuk dan dihancurkan |
| **5** |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran |
| **6** |  | *Line Conector* | Digunakan untuk menghubungkan satu simbol dengan simbol yang lain |
| **7** |  | *Decision* | Menunjukan suatu keputusan yang mempunyai satu atau lebih transisi sesuai dengan kondisi |