**BAB II**

**LANDASAN TEORI**

* 1. **Tinjauan Studi**

Berikut ini adalah beberapa penelitian yang menjadi referensi penulis dalam menyusun Usulan Penelitian ini

1. Nunu Kustian, Fakultas Tehnik dan MIPA, Universitas Indraprasta PGRI, Jakarta. Judul yang diangkat “Sistem Informasi Pengamanan Basis Data Menggunakan Teknik Enkripsi Bagian Tata Usaha Lembaga Sandi Negara”Tahun 2014**.**
2. Permasalahan pada penelitian ini adalah *user* sering mengalami gangguan dalam mengatasi administrasi dan tidak amannya basis data karena sering terjadinya kebocoran data.
3. Solusinya adalah dengan membuat sistem keamanan login aplikasi program menggunakan enkripsi berbasis program PHP dan MySQL dengan implementasi MD5 pada menu login.
4. Hasil akhirnya adalah aplikasi menampilkan tentang hal-hal yang mengenai berbagai macam kerusakan data.
5. Aditya eka arifyanto, Mahasiswa Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang. Judul yang diangkat adalah, “Implementasi Enkripsi Basis Data Berbasis Web dengan Algoritma *Steram Cipher* RC4”, Tahun 2013.
6. Masalah yang dihadapi adalah penyalahgunaan terhadap hak akses basis data, sehingga dapat terjadi pembacaan data, manipulasi data dan perusakan data oleh pihak yang tidak berwenang sehingga integritas data kurang terjaga.
7. Solusinya adalah dengan menerapkan Enkripsi Basis Data Berbasis Web Dengan Algoritma *Stream Cipher* RC4.
8. Dari hasil akhir pengujian, Panjang data kunci yang digunakan mempengaruhi kekuatan dari sistem enkripsi data. Semakin panjang kunci yang digunakan, maka semakin kuat sistem keamanan tersebut dari serangan. Setelah dilakukan enkripsi pada table *password*, *password* yang biasanya dapat dibaca kini sulit untuk dimengerti oleh orang yang tidak memiliki key untuk mendekripsikannya.
9. Peneliti Egar Dika Santoso, Mahasiswa Program Studi Teknik Informatika, Universitas Dian Nuswantoro, Semarang. Judul yang diangkat adalah “Implementasi Algoritma *Caesar Cipher* Dan *Hill Cipher* Pada *Database* Sistem Inventori TB Mita Jepara” Tahun 2014.
10. Permasalahan yang dihadapi tidak amannya *database* karena bisa diaksesnya mysql secara langsung tanpa harus memasuki sistem inventori TB Mita.
11. Solusinya adalah merancan keamanan sistem inventori TB Mita menggunakan enkripsi *Caesar Cipher* dan *Hill Cipher.*
12. Hasil dari penelitian dengan mengunakan algoritma Caesar Cipher dan Hill Cipher adalah dengan melihat hasil enkripsi database sistem inventori TB Mita Jepara. Hasil output yang didapat kolom (*field*) persediaan barang pada invetori TB Mita Jepara dapat terenkripsi.
    1. **Tinjauan Teori**
       1. **Basis Data**

Dari awal penggunaan komputer, penyimpanan dan manipulasi data merupakan fokus utama aplikasi. Pada awal tahun 1960, Charles Bachman diperusahaan General Electric mendesain generasi pertama DBMS yang disebut Penyimpanan Data Terintegrasi (*Integrated Data Store*). Dasar untuk [model data](http://informatika.web.id/model-data.htm) jaringan dibentuk lalu distandardisasi oleh *Conference on Data System Language* (CODASYL). Kemudian, Bachman menerima ACM *Turing Award* (Penghargaan semacam nobel pada ilmu komputer) di tahun 1973.

Pada akhir tahun 1960-an, IBM mengembangkan sistem manajemen informasi (*Information Manajemen System*) IMS. IMS dibentuk dari representasi data pada kerangka kerja yang disebut model data hierarki. Dalam waktu yang sama, hasil kerja sama antara IBM dengan perusahaan penerbangan Amerika mengembangkan system SABRE. Sistem SABRE memungkinkan *user* mengakses data yang sama pada jaringan komputer.

Pada tahun 1970, Edgar Codd di laboratorium penelitian di San Jose mengusulkan suatu representasi data baru yang disebut model data relational. Pada tahun 1980, model relasional menjadi paradigm DBMS paling dominan. Bahasa *query* SQL dikembangkan untuk basisdata relasional sebagai bagian proyek Sistem R dari IBM. SQL di standardisasi di akhir tahun 1980 dan SQL-92 diadopsi oleh American National Standards Institute (ANSI) dan International Standards Organization (ISO). Program yang digunakan untuk eksekusi bersamaan dalam basisdata disebut transaksi. *User* menulis programnya, dan bertanggung jawab menjalankan program secara bersamaan terhadap DBMS. Pada tahun 1999, James Gray memenangkan *Turing award* untuk kontribusinya pada manajemen transaksi dalam DBMS.

Pada akhir tahun 1980 dan permulaan tahun 1990, banyak bidang sistem basisdata dikembangkan. Penelitian dibidang basisdata meliputi bahasa *query* yang *powerful*, model data yang lengkap, dan penekanan pada dukungan analisis data yang kompleks semua bagian organisasi. Beberapa vendor (misalnya IBM, DB2, Oracle8, dan Informix UDS) memperluas sistemnya dengan kemampuan menyimpan tipe data baru misalnya image dan text serta kemampuan *query* yang kompleks. System khusus dikembangkan banyak vendor untuk membuat [*data warehouse*](http://informatika.web.id/data-warehouse.htm) dan mengonsolidasi data beberapa basisdata.

Suatu fenomena menarik adalah munculnya *enterprice resource planning* (ERP) dan *management resource planning* (MRP), yang menambah lapisan substansial dari fitur berorientasi aplikasi pada DBMS utama. Paket yang digunakan secara luas meliputi Baan, Oracle, PeopleSoft, SAP, dan Siebel. Paket tersebut mengidentifikasi kumpulan tugas umum (misalnya manajemen inventori, perencanaan sumber daya manusia, dan analisis keuangan) yang dihadapi oleh sejumlah besar organisasi dan menyediakan lapisan aplikasi umum untuk melaksanakan tugas. Data disimpan dalam DBMS relasional. Kemudian, lapisan aplikasi dapat disesuaikan pada perusahaan berbeda sehingga biaya keseluruhan perusahaan menjadi lebih rendah disbanding biaya pembuatan lapisan aplikasi dari awal. Lebih jauh, DBMS memasuki dunia internet. Saat generasi pertama, web site menyimpan datanya secara ekskulisif dalam file sistem operasi. Pada saat ini, DBMS dapat digunakan untuk menyimpan data yang dapat diakese melalui web browser. Query dapat dibuat melalui form web dan format jawabannya dengan menggunakan *markup language semisal* HTML untuk mempermudah tampilan pada browser. Semua vendor basisdata menambah fitur ini untuk DBMS mereka.

Manajemen basis data mempertimbangkan pentingnya suatu data bersifat *online* dan dapat diakses melalui jaringan komputer. Saat ini, bidang seperti ini diwujudkan dalam basisdata [multimedia](http://informatika.web.id/multimedia.htm), video unteraktif, perpustakaan digital, proyek ilmuwan seperti proyek pemetaan, proyek system obeservasi bumi milik NASA, dan lain sebagainya (Ramakrishnan and Gehrke, 2003).

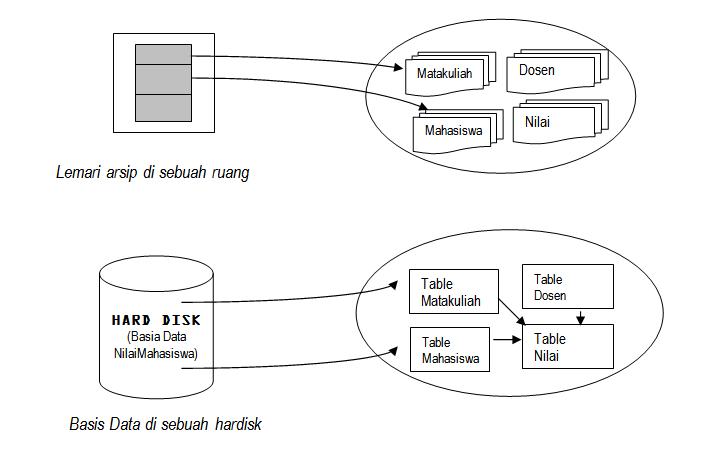
Menurut George Tsu-der Chou basis data merupakan kumpulan informasi bermanfaat yang diorganisasikan kedalam aturan yang khusus.

Definisi lain dari basis data menurut Fabbri dan Schwab adalah sistem berkas terpadu yang dirancang terutama untuk meminimalkan duplikasi data.

Menurut Ramez Elmasri mendefinisikan basis data lebih dibatasi pada arti implisit yang khusus, yaitu:

* + - * 1. Basis data merupakan penyajian suatu aspek dari dunia nyata (*real word*).
        2. Basis data merupakan kumpulan data dari berbagai sumber yang secara logika mempunyai arti implisit. Sehingga data yang terkumpul secara acak dan tanpa arti, tidak dapat disebut basis data.
        3. Basis data perlu dirancang, dibangun dan data dikumpulkan untuk suatu tujuan. Basis data dapat digunakan oleh beberapa *user* dan beberapa aplikasi yang sesusai dengan kepentingan *user*.

Dari beberapa definisi-definisi tersebut dapat sikatakan bahwa basis data mempunyai sumber data dalam pengumpulan data, bervariasi derajat interaksi derajat kejadian dari dunia nyata, dirancang dan dibangun agar dapat digunakan oleh beberapa *user* untuk berbagai kepentingan (Haidar Dzacko, 2007).

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja yang sama. Prinsip utamanya adalah pengaturan data/arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip. Perbeaannyya hanya terdapat pada media penyimpanan yang digunakan. Jika lemari arsip menggunakan lemari dari besi atau kayu sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti cakram magnetis (*magnetic disk* atau disingkat sebagai *disk* saja). Hal ini merupakan konsekuensi yang logis, Karena lemari arsip langsung dikelola oleh manusia, sementra basis data dikelola melalui perantaraan mesin pentar elektronis (yang kita kenal sebagai komputer). Perbedaan media ini yang selanjutnya melahirkan perbedaan-perbedaan lain yang menyangkut jumlah dan metode yang dapat digunakan dalam upaya penyimpanan.

**Gambar 2.1** Lemari Arsip dan Basis Data

* + 1. **Keamanan** **Jaringan (*Network Security*)**

Sering ada kesalah pahaman masyarakat awam antara keamanan jaringan dan keamanan komputer. Ada yang menganggap bahwa keduanya adalah sama, dan ada pula yang salah menjabarkan arti dari keduanya. Perbedaaan antara keamanan jaringan dan keamanan komputer dapat dijabarkan sebagai berikut (Stalling, 2006):

Keamanan komputer adalah kumpulan piranti yang diranncang untuk melindungi komputer sehingga data pada komputer terlindungi. Sedangkan, keamanan jaringan adalah kumpulan piranti yang dirancang untuk melindungi data ketika transmisi terhadap ancaman pengaksesan, pengubahan dan penghalangan oleh pihak yang tidak berwenang. Sistem anti virus komputer merupakan contoh piranti keamanan komputer, sedangkan protocol web yang aman merupakan contoh piranti keamanan jaringan.

Jenis-jenis keamanan jaringan berdasarkan rekomendasi lembaga internasional *International Telecommunication Union-Telecomunication Standardiation Sector* (ITU-T) pada dokumen X..800, (ITU, 1991):

1. Otentikasi (*Authentication*)

Otentikasi (*Authentication*) adalah layanan keamanan jaringan yang memastikan penggunaan sistem oleh orang yang berhak mengakses. Biasanya melibatkan username dan password, tapi dapat menyertakan metode lain yang menunjukan identitas, seperti kartu pintar, sidik jari, dll.

1. Kendali Akses (*Access Control*)

Kendali Akses (*Access Control*) adalah layanan keamanan jaringan yang menghalangi penggunaan tidak terotorisasi terhadap sumber daya. Misalnya sebuah data rekam medik elektronik hanya dapat diakses oleh pasien dan para medis yang terlibat.

1. Kerahasian Data (*Data Confidentiality*)

Kerahasian Data (*Data Confidentiality*) adalah layanan keamanan jaringan yang memproteksi data transmisi terhadap pengungkapan oleh piihak yang tidak berwenang.

1. Keutuhan Data (*Data Integrity*)

Keutuhan Data (*Data Integrity*) adalah layanan keamannan jaringan yang memastikan bahwa data yang diterima oleh penerima adalah benar-benar sama dengan data yang dikirim oleh pengirim.

1. *Non-Repudiation*

Layanan *Non-Repudiation* adalah layanan keamanan jaringan yang menghindari penolakan atas penerimaan/pengiriman data yang telah dikirim.

1. Ketersediaan (*Availability*)

Layanan ketersediaan (*Availability*) adalah layanan sistem yang membuat sumber daya sistem tetap dapat diakses dan dapat digunakan ketika ada permintaan dari pihak-pihak berwenang.

Rekomendasi ITU-T (X.800) juga mendefinisikan beberapa mekanisme keamanan jaringan, yaitu sebagai berikut:

1. *Encipherment*

*Encipherment* merupakan mekanisme keamanan jaringan yang digunakan untuk menyembunnyikan data

1. Keutuhan Data

Mekanisme keutuhan data dapat digunakan untuk memastikan keutuhan dapa pada unit data atau pada suatu aliran (*stream*) data unit. Cara yang digunakan adalah dengan menambahkan nilai penguji dihitung terlebih dahuulu dan kemudian data dan penguji dikirim bersamaan.

1. *Digital Signature*

*Digital signature* merupakan mekasnisme keamanan jaringan yang menyediakan cara bagi pengirim data untuk “menandatangani” secara elektronik sebuah data dan penerima dapat memverifikasi “tanda tangan itu secara elektronik. *Digital signature* ditambahkan pada data unit dan digunakan sebagai bukti sumber pengirim dan menghindari pemalsuan (*forgery*) tanda tangan.

1. *Authentication Exchange*

Mekanisme ini memberikan cara agar dua entitas dapat saling meng-otentikasi dengan cara bertukar pesan untuk saling membuktikan identitas.

1. *Traffic Padding*

*Traffic Padding* menyediakan cara untuk pencegahan analisis lalu lintas data pada jaringan yaitu dengan menambah data palsu pada lalu lintas data.

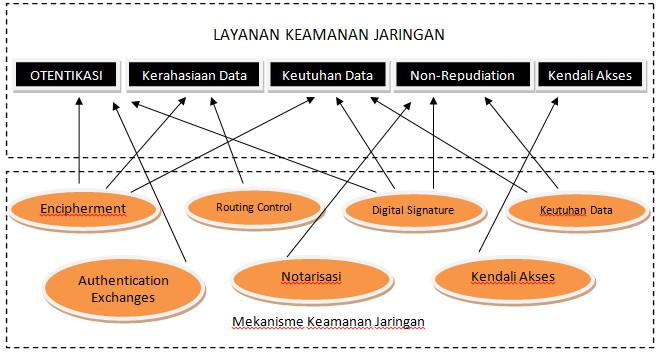
1. *Routing Control*

*Routing Control* menyediakan cara untuk memilih dan secara terus-menerus mengubah alur (*route*) pada jaringan komputer antara pengirim dan penerima. Mekanisme ini menghindarkan komunikasi dari penguping (*eavedropper*).

1. Notarisasi

Notarisasi (*notarization*) menyediakan cara untuk memilih pihak ketigayang terpercaya sebagai pengendali komunikasi antara pengirim dan penerima.

1. Mekanisme kendali akses

Mekanisme kendali akses memberikan cara bagi pengguna untuk memperoleh hak akses sebuah data.

**Gambar 2.2** Hubungan antara layanan dan mekanisme dan keamanan jaringan

* + 1. **Serangan Keamanan Jaringan**

Secara umum serangan pada sistem keamanan jaringan dapat dikategorikan menjadi 2 jenis: serangan pasif (*passive attack*) dan serangan aktif (*active attack*).

1. Serangan pasif

Pada serangan pasif penyerang hanya mengumpulkan data yang melintas pada jaringan pubik (jaringan yang bias diakses oleh penyerang). Serangan pasif tidak melakukan modifikasi data yang melintas atau merusak sistem, penyerang hanya punya kemampuan membaca saja (*read only*). Lalu berdasarkan data yang dikumpulkan, penyerang melakukan analisis untuk menggagalkan tujuan layanan keamanan jaringan. Karena tidak melakukan perubahan data dan mengganggu sistem, serangan pasif susah untuk dideteksi namun serangan pasif dapat dicegah dengan cara misalnya selalu memberikan sandi (*encryption*) ketika pengiriman pesan. Berikut ini beberapa serangan yang digolongkan sebagai serangan pasif:

1. *Snooping*

*Snooping* merujuk pada kegiatan yang bermaksud mendapatkan data yang tengah terkirim pada jaringan biasanya melalui akses yang tak berwenang.

1. *Traffic Analysis*

*Traffic Analysis* merupakan kegiatan serangan pasif dengan melakukan *monitoring* terhadap lalulintas data pada jaringan. Data-data lalu lintas jaringan dikumpulkan dan kemudian dianalisis sehingga penyerang dapat mengetahui maksud data-data itu.

1. Serangan aktif

Sebuah serangan aktif (*active attack*) dapat mengakibatkan perubahan data yang terkirim dan jalannya sistem terganggu. Pada serangan aktif seakan-akan penyerang memperoleh kemampuan untuk mengubah data pada lalu lintas data selain kemampuan baca. Jenis-jenis serangan aktif adalah sebagai berikut:

1. *Masquerade*

*Masquerade* adalah serangan aktif yang dilakukan penyerang mengambil alih (menirukan) perilku pengirim atau penerima.

1. *Modification*

*Modification* adalah serangan aktif yang dilakukan oleh penyerang dengan cara penyerang mengambil alih jalur komunikasi untuk mengubah, menghapus, atau menunda pesan yang sedang terkirim untuk keuntungan penyerang.

1. *Replay*

*Replay* seranga aktif yang terdiri atas pencatatan secara pasif data unit dan transisi ulang untuk menimbulkan efek yang diinginkan penyerang.

1. *Denial Of Service*

*Denial Of Service* adalah serangan aktif yang bertujuan agar sistem menjadi *coolapse* sehingga tidak mampu memberikan respons atau layanan semestinya kepada pengguna. Serangan ini biasanya dilakukan dengan membuat *server* menjadi *overload* dengan permintaan bodong (*dummy*).

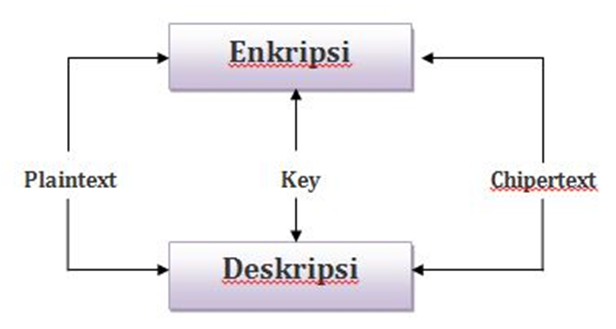
1. *Man In The Middle*

Serangan keamanan jaringan *Man In The Middle* terjadi saat user perusak dapat memposisikan diantara dua titik link komunikasi. Para penyerang ini tidak tampak pada kedua sisi link komunikasi ini dan bias mengubah isi dan arah *traffic*. Dengan cara ini para penyerang bsa menangkap *logon credensial* atau data *sensitive* ataupun mampu mengubah isi pesan dari kedua titik komunikasi ini.

Layanan keamanan jaringan diwujudkan dengan menggunakan mekanisme keamanan jaringan. Mekanisme keamanan jaringan pada implementasinya menggunakan teknik-teknik penyandian, salah satunya yaitu kriptografi.

* + 1. **Kriptografi**

Kriptografi berasal dari dua kata Yunani, yaitu *Crypto* yang berarti rahasia dan *Grapho* yang berarti menulis. Secara umum kriptografi dapat diartikan sebagai ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan (Y. Yohakim Marwanta, 2008).

Secara umum *enkripsi* dan *dekripsi* dapat digambarkan sebagai berikut:

**Gambar 2.3** Gambar proses *enkripsi* dan *dekripsi*

Dalam sistem komputer, pesan terbuka (*plaintext*) diberi lambing M, yang merupakan singkatan dari *message*. *Plaintext* ini dapat berupa tulisan, foto, atau video yang berbentuk data *biner*. *Plaintext* inilah yang nantinya akan di dekripsi menjadi pesan rahasia atau *ciphertext* yang dilambangkan denga C (*ciphertext*). Secara matematis, fungsi enkripsi dinotasikan dengan:

|  |
| --- |
| ***E(M)=C*** |

Sedangkan fungsi dekripsi adalah proses pembaliakn dari *ciphertext* menjadi *plaintext* kembali. Secara matematis dinotasikan sebagai berikut:

|  |
| --- |
| ***D(C)=M***  ***D(E(M))=M*** |

Sejarah penulisan rahasia tertua dapat ditemukan pada peradaban mesir kuno, yakni tahun 3000 SM. Bangsa Mesir menggunakan ukiran rahasia yang disebut *hieroglyphics* untuk menyampaikan pesan kepada orang-orang yang berhak.

[](http://2.bp.blogspot.com/-MC-4E5BXknc/UtOplOJCVAI/AAAAAAAAEVY/yKRWQ4MLMKQ/s1600/Skytale.jpg)Awal tahun 400 SM bangsa Spartan di Yunani memanfaatkan kriptografi di bidang militer dengan menggunakan alat yang disebut *scytale*, yakni pita panjang berbahan daun papyrus yang dibaca dengan cara digulungkan ke sebatang silinder. Sedangkan peradaban Cina dan Jepang menemukan kriptografi pada abad 15 M.

**Gambar 2.4** *Scytale*

Peradaban Islam juga menemukan kriptografi karena penguasaannya terhadap matematika, statistik, dan linguistik. Bahkan teknik kriptanalisis dipaparkan untuk pertama kalinya pada abad 9 M oleh seorang ilmuwan bernama Abu Yusuf Ya’qub ibn ‘Ishaq as-Shabbah al Kindi atau dikenal dengan Al-Kindi yang menulis kitab tentang seni memecahkan kode. Kitabnya berjudul ***Risalah fi Istikhraj al-Mu’amma*** (Manuskrip untuk memecahkan pesan-pesan Kriptografi). Terinspirasi dari perulangan huruf dalam Al-Qur’an, Al-Kindi menemukan teknik analisis frekuensi, yakni teknik untuk memecahkan ciphertext berdasarkan frekuensi kemunculan karakter pada sebuah pesan (Wirdasari, 2008).

|  |
| --- |
| [Risalah fi Istikhraj al-Mu’amma](http://4.bp.blogspot.com/-wjF0V2w5MLQ/UtOqdqe83pI/AAAAAAAAEVk/jTDf6ZEaKvg/s1600/Risalah+fi+Istikhraj+al-Mu%E2%80%99amma.gif) |
|  |

**Gambar 2.5** Risalah fi Istikhraj al-Mu’amma

* + 1. **Tujuan Kriptografi**

Kriptografi bertujuan untuk memberikan layanan pada aspek-aspek keamanan antara lain:

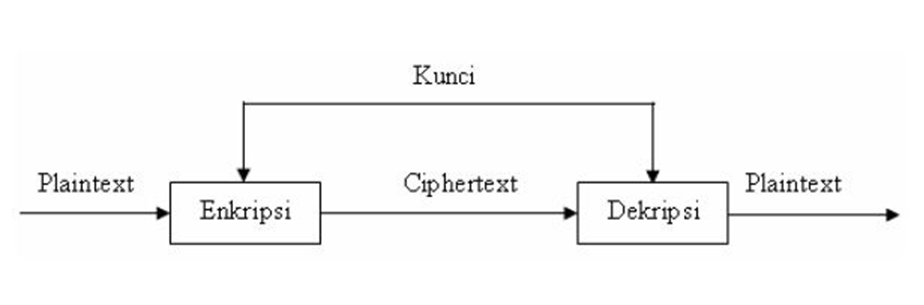
1. Kerahasiaan (*confidentiality*), yaitu menjaga supaya pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak,
2. Integritas data (*data integrity*), yaitu memberikan jaminan bahwa untuk tiap bagian pesan tidak akan mengalami perubahan dari saat data dibuat/dikirim oleh pengirim sampai dengan saat data tersebut dibuka oleh penerima data,
3. Otentikasi (*authentication*), yaitu berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi maupun mengidentifikasi kebenaran sumber pesan,
4. Nirpenyangkalan (*non repudiation*), yaitu memberikan cara untuk membuktikan bahwa suatu dokumen datang dari seseorang tertentu sehingga apabila ada seseorang yang mencoba mengakui memiliki dokumen tersebut, dapat dibuktikan kebenarannya dari pengakuan orang tersebut.
   * 1. **Istilah-Istilah Dalam Kriptografi**

Di dalam kriptografi akan ditemui beberapa istilah diantaranya:

* + - * 1. Plaintext adalah pesan yang hendak dikirimkan (berisi data asli).
        2. Ciphertext adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
        3. Kunci adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.
        4. Enkripsi adalah sebuah proses penyandian yang melakukan perubahan sebuahkode (pesan) dari yang bisa dimengerti (plainteks) menjadi sebuah kode yang tidakbisa dimengerti (cipherteks).
        5. Dekripsi adalah proses merubah cipherteks menjadi plainteks disebut. Proses enkripsi dan dekripsi memerlukan suatu mekanisme dan kunci tertentu.
        6. Cipher adalah suatu fungsi matematis yang digunakan untuk melakukan enkripsi dan dekripsi.
        7. Kriptoanalisis (cryptanalysis) adalah kebalikan dari kriptografi, yaitu suatu ilmu untuk memecahkan mekanisme kriptografi dengan cara mendapatkan kunci dari cipherteks yang digunakan untuk mendapatkan plainteks.
        8. Kriptanalis (cryptanalys) adalah orang yang melakukan kriptanalisis.
        9. Kroptografer (*cryptographer*) adalah seseorang yang menekuni dan melakukan aktifitas di bidang kriptografi.
        10. Kriptologi (cryptology) adalah ilmu yang mencakup kriptografi dan kriptoanalisis.
        11. Alice adalah pengirim pesan.
        12. Bob adalah penerima pesan.
        13. Eva (*eavedropper*) adalah penyadap.
    1. **Jenis-Jenis Algoritma Kriptografi**

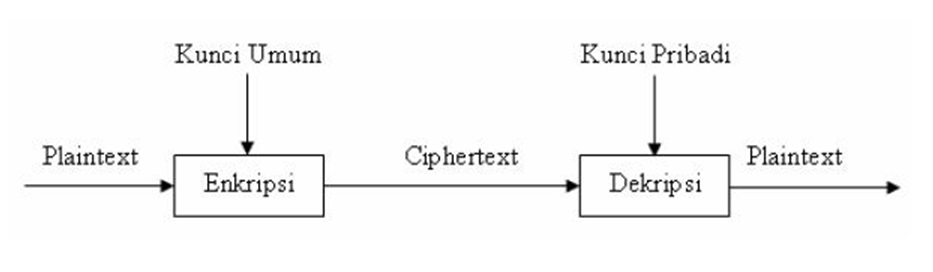
Secara umum berdasarkan kesamaan kuncinya, algoritma kriptografi dibedakan menjadi:

* + - * 1. Algoritma simetris (*symmetric-key*)

*Key* yang digunakan pada algoritma ini, antara pengirim dan penerima adalah sama, sedangkan proses yang dilakukan untuk dekripsinya melakukan kebalikan dari proses enkripsi. Kekuatan algoritma simetrik sangat bergantung pada satu *key* yang digunakan. Jika *key* dapat dikirimkan secara aman maka kemungkinan mendapatkan plaintext dan chipertext yang dikirimkan akan semakin kecil. Contoh: RSA, ElGamal, McEliece, LUC, DSA.

**Gambar 2.6** Gambar proses *enkripsi* dan *dekripsi symmetric-key*

* + - * 1. Algoritma asimetri (*asymmetric-key*)

Public Key menggunakan dua key yang berbeda dalam melakukan proses enkripsi dan dekripsi. Public Key yang digunakan untuk melakukan enkripsi dan boleh diketahui umum. Sedangkan Private Key hanya boleh diketahui oleh pihak penerima. Contoh: DES (*Data Encryption Standard*), *Blowfish*, *Twofish*, MARS, IDEA, 3DES (DES diaplikasikan 3 kali), AES (*Advanced Encryption Standard*, yang bernama asli rijndael), RC6, A5.**

**Gambar 2.7** Gambar proses *enkripsi* dan *dekripsi asymmetric-key*

* + - * 1. Fungsi *Hash*

Fungsi *hash* sering disebut dengan fungsi *hash* satu arah (*one-way function*), *message digest*, *fingerprint*, fungsi kompresi dan *message authentication code* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya kedalam urutan biner denganpanjang yang tetap. Fungsi *hash* biasanya diperlukan jika ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan duatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang di inginkan.

Berdasaran arah implementasi dan pembabakan zaman, algoritma kriptografi dibedakan menjadi:

1. Algoritma sandi klasik

Algoritma ini digunakan sejak sebelum era komputerisasi dan kebanyakan menggunakan teknik kunci simetris. Metode menyembunyikan pesannya adalah dengan teknik substitusi atau transposisi atau keduanya (Sadikin, 2012). Teknik substitusi adalah menggantikan karakter dalam plaintext menjadi karakter lain yang hasilnya adalah ciphertext. Sedangkan transposisi adalah teknik mengubah plaintext menjadi ciphertext dengan cara permutasi karakter. Kombinasi keduanya secara kompleks adalah yang melatarbelakangi terbentuknya berbagai macam algoritma kriptografi modern (Prayudi, 2005).

1. Algortima sandi modern

Algoritma ini memiliki tingkat kesulitan yang kompleks (Prayudi, 2005), dan kekuatan kriptografinya ada pada key atau kuncinya (Wirdasari, 2008). Algoritma ini menggunakan pengolahan simbol biner karena berjalan mengikuti operasi komputer digital. Sehingga membutuhkan dasar berupa pengetahuan terhadap matematika untuk menguasainya (Sadikin, 2012).

Berdasarkan besarnya data yang diolah, algoritma kriptografi dibedakan menjadi:

1. Cipher aliran (*stream cipher*)

Stream Chiper melakukan pengkodean 1 bit atau byte dalam satu kali prosesnya Stream Chiper lebih muda diimplementasikan dalam hardware. Hardware bekerja berdasarkan bit-bit yang merupakan satuan terkecilnya dalam melakukan proses perhitungannya. Contoh: RC4, Seal, A5, Oryx, dan lain-lain.

1. Cipher blok (*block cipher*)

Block Chiper melakukan pengkodean 1 block dalam sekali proses. Ukuran block ini sendiri dapat ditentukan sesuai keinginan. Namun dalam prakteknya ukuran block yang digunakan memenuhi rumu 2n dengan n bilangan integer. Contoh: *Blowfish*, DES, Gost, *Idea*, RC5, Safer, *Square*, *Twofish*, RC6, Loki97, dan lain-lain.

Algoritma yang akan peneliti gunakan pada penelitian ini adalah salah satu algoritma simetri (*symmetric-key*) dengan kunci publik (*public-key*), yaitu algoritma *steream cipher* RC4 (*Rivest Code* 4).

* + 1. **RC4** **(*Rivest Code* 4)**

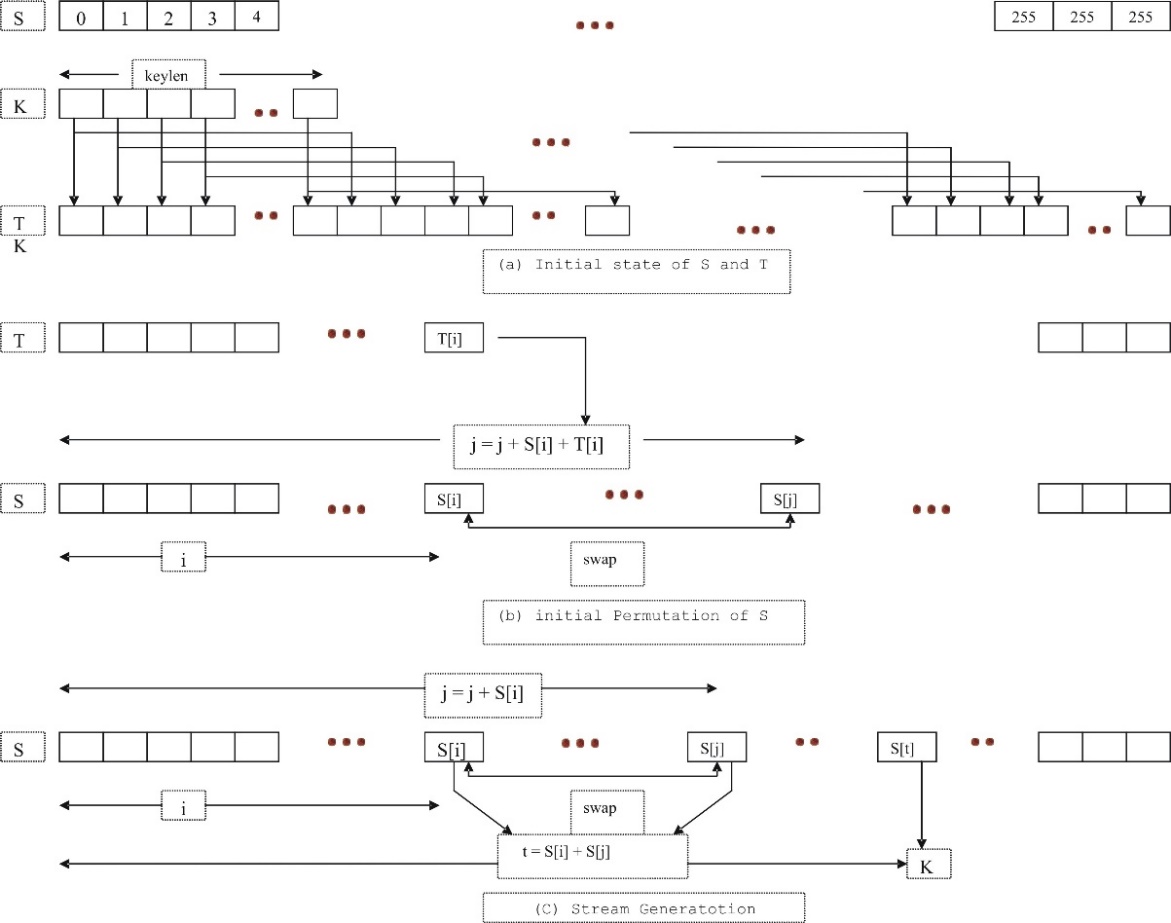
Algoritma kriptografi *Rivest Code* 4 (RC4) merupakan salah satu algoritma kunci simetris dibuat oleh RSA *Data Security Inc* (RSADSI) yang berbentuk *stream cipher*. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA (merupakan singkatan dari tiga nama penemu: Rivest Shamir Adleman). RC4 menggunakan panjang kunci dari 1 sampai 256 *byte* yang digunakan untuk menginisialisasikan tabel sepanjang 256 *byte*. Tabel ini digunakan untuk generasi yang berikut dari *pseudo random* yang menggunakan XOR dengan *plainteks* untuk menghasilkan *cipherteks*. Masing-masing elemen dalam tabel saling ditukarkan minimal sekali.

RC4 merupakan merupakan salah satu jenis *stream cipher*, yaitu memproses unit atau *input* data pada satu saat. Dengan cara ini *enkripsi* atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah *input* data tertentu sebelum diproses, atau menambahkan *byte* tambahan untuk mengenkrip. Metode *enkripsi* RC4 sangat cepat kurang lebih 10 kali lebih cepat dari DES (Heru Setyawan, 2016).

RC4 menggunakan panjang variabel kunci dari 1 s.d 256 *byte* untuk menginisialisasi *state* tabel.  State table digunakan untuk pengurutan menghasilkan *byte pseudo-random* yang kemudian menjadi *stream pseudo-random*. Setelah di-XOR dengan *plaintext* sehingga didapatkan *ciphertext*. Tiap elemen pada *state* table di *swap* sedikitnya sekali. Kunci RC4 sering dibatasi sampai 40 bit, tetapi dimungkinkan untuk mengunakan kunci 128 bit. RC4 memiliki kemampuan penggunaan kunci antara 1 sampai 2048 bit. Panjang kunci merupakan faktor utama dalam sekuritas data. RC4 dapat memiliki kunci sampai dengan 128 bit. Protokol keamanan SSL (*Secure Socket Layer*) pada *Netscape Navigator* menggunakan algoritma RC4 40-bit untuk enkripsi simetrisnya.

Algoritma RC4 memiliki dua fase, setup kunci dan pengenkripsian. Setup untuk kunci adalah fase pertama dan yang paling sulit dalam algoritma ini. Dalam setup S-bit kunci (S merupakan panjang dari kunci), kunci *enkripsi* digunakan untuk menghasilkan variabel *enkripsi* yang menggunakan dua buah *array*, *state* dan kunci, dan sejumlah-S hasil dari operasi penggabungan. Operasi penggabungan ini terdiri dari pemindahan (*swapping*) *byte*, operasi modulo, dan rumus lain. Operasi modulo merupakan proses yang menghasilkan nilai sisa dari satu pembagian. Sebagai contoh, 11 dibagi 4 adalah 2 dengan sisa pembagian 3, begitu juga jika tujuh modulo empat maka akan dihasilkan nilai tiga. Variabel *enkripsi* dihasikan dari setup kunci dimana kunci akan di XOR-kan dengan *plaintext* untuk menghasilkan teks yang sudah ter*enkripsi*. XOR merupakan operasi logik yang membandingkan dua bit biner. Jika bernilai beda maka akan dihasilkan nilai 1. Jika kedua bit sama maka hasilnya adalah 0. Kemudian penerima pesan akan mendekripnya dengan meng XOR-kan kembali dengan kunci yang sama agar dihasilkan pesan dari *plaintext* tersebut. Untuk menunjukan cara kerja dari algoritma RC4, berikut dapat dilihat pada blok di bawah:

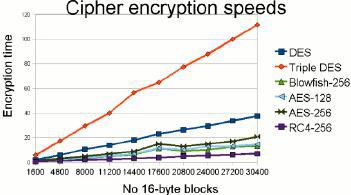
RC4 menggunakan dua buah kotak substitusi (S-*Box*) *array* 256 *byte* yang berisi permutasi dari bilangan 0 sampai 255 dan S-*Box* kedua yang berisi permutasi fungsi dari kunci dengan panjang yang variabel.

**Gambar 2.8** Cara kerja RC4 (*Muhammad Ridho, Rika Fitriana, Sepri Hardianti, Suhendi, 2010)*

* + 1. **Kecepatan RC4 Sebagai Salah Satu Metode Enkripsi**

(Jumrin, Sutardi, Subardin, 2016) Kinerja RC4 sebagai metode enkripsi tergolong sangat cepat. Selain cepat, waktu RC4 tidak terpengaruh dengan panjang keylength yang dipakai. *Byte* K di-XOR-kan dengan *plaintext* untuk menghasilkan *ciphertext* atau di-XOR-kan dengan *ciphertext* untuk menghasilkan *plaintext*. Enkripsi sangat cepat kurang lebih 10 kali lebih cepat dari DES.

Gambar 2.9 menunjukkan perbandingan waktu yang digunakan untuk enkripsi dari berbagai metode.



**Gambar 2.9** Perbandingan waktu enkripsi dari berbagai metode (Jumrin, Sutardi, Subardin, 2016)

* + 1. **Kelebihan dan Kekurangan Algoritma RC4**

Kelebihan algoritma RC4 yaitu:

1. Kesulitan mengetahui sebuah nilai dalam table.
2. Kesulitan mengetahui lokasi mana di dalam table yang digunakan untuk menyeleksi masing-masing nilai.
3. Kunci RC4 tentu hanya dapat digunakan sekali.
4. Model enkripsi ini 10 kali lebih cepat dari DES.

Kekurangan algoritma RC4 yaitu:

1. Algoritma RC4 lebih mudah diserang dari dengan menggunakan analisa dari bagian dalam table.
2. Salah satu dari 256 kunci dapat menjadi kunci yang lemah. Kunci ini di identifikasi oleh kriptoanalisis yang dapat menemukan keadaan dimana salah satu dari bit yang dihasilkan mempunyai korelasi yang kuat dengan sedikit bit kunci.
   * 1. **Penerapan** **Algoritma RC4**

Untuk menunjukan bagaimana RC4 bekerja pada tingkat dasar, mari kita *state*-*array* 4 bit. Hal ini dikarenakan akan sangat sulit menggambarkan proses RC4 secara manual dengan *state*-*array* 256 bit. Kali ini saya akan mengenkripsi kata HALO dengan kunci 2573.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 5 | 7 | 3 |

Pertama menginisialisasi *array* S 4 bit sehingga terbentuk *state*-*array* S dan *state*-*array* K sebagai berikut:

*Array* S *Array* K

Inisialisasi i dan j dengan 0 kemudian dilakukan KSA agar tercipta *state*-*array* yang acak. Penjelasan iterasi lebih lanjut dapat dijelaskan sebagai berikut:

* **Iterasi 1**

i = 0

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

= (0 + 0 + 2) mod 4 = 2

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 1 | 0 | 3 |

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

Hasil *array* S

* **Iterasi** **2**

i = 1

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

= (2 + 1 + 5) mod 4 =0

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 0 | 3 |

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

Hasil *array* S

* **Iterasi 3**

i = 2

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

= (0 + 0 + 7) mod 4 = 3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 0 |

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

Hasil *array* S

* **Iterasi 4**

i = 3

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

= (3 + 0 + 3) mod 4 = 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 0 | 3 |

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

Hasil *array* S

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 tiap-tiap karakter pada plainteks. Berikut adalah tahapan penghasilan kunci enkripsi dengan PRGA.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 0 | 3 |

*Array* S

Inisialisasi

i = 0

j = 0

* **Iterasi** **1**

i = (0 + 1) mod 4 = 1

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

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 2 | 3 |

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 2 | 3 |

swap (S[3],S[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

Setelah menemukan kunci untuk tiap karakter, maka dilakukan operasi XOR antara karakter pada plaintext dengan kunci yang dihasilkan. Berikut adalah tabel ASCII untuk tiap-tiap karakter pada plainteks yang digunakan.

**Tabel 2.1** Kode ASCII untuk setiap karakter plainteks yang digunakan

|  |  |
| --- | --- |
| Huruf | Kode ASCII (binary 8 bit) |
| H | 01001000 |
| A | 01000001 |
| L | 01001100 |
| O | 01001111 |

**Tabel 2.2** Proses XOR kunci enkripsi dengan plaintext pada enkripsi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | H | A | L | O |
| Plaintext | 01001000 | 01000001 | 01001100 | 01001111 |
| Key | 00000010 | 00000011 | 00000010 | 00000010 |
| Ciphertext | 01001010 (L) | 01000010 (B) | 01001110 (N) | 01101101 (M) |

Setelah terkirim, pesan yang telah dienkripsi akan didekripsikan. Proses pendekripsian dilakukan dengan proses XOR antara kunci dekripsi yang sama dengan kunci dekripsi dengan cipherteks yang dapat dilihat di tabel 2.3.

**Tabel 2.3** Proses XOR kunci dekripsi dengan ciphertext pada dekripsi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | L | B | N | M |
| Ciphertext | 01001010 | 01000010 | 01001110 | 01101101 |
| Key | 00000010 | 00000011 | 00000010 | 00000010 |
| Plaintext | 01001000 (H) | 0100000 (A) | 01001100 (L) | 01001111 (O) |

* 1. **Model Desain Sistem**
     1. **Latar Belakang UML**

UML adalah bahasa spesifikasi standar unntuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. *Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Objek Management Group*, sebuah organisasi yangyang telah mengembangkkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para partisi OOP. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiapp sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

* + 1. **Pengertian UML**

UML (*Unified Modeling Language*) adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dari suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada penelitian bisnis dan sistem non perangkat lunak lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan suatu sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

* + 1. **Tujuan UML**

Tujuan UML adalah:

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan
3. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti decara umum
4. UML bisa juga berfungsi sebagai sebuah (*blue print*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui informasi secara detail tentang coding program atau bahkan membaca program dan menginterpretasikan kembali kedalam bentuk diagram (*reverse enginering*).
   * 1. ***Use Case Diagram***

*Use case* adalah deskripsi fungsi dari sebuah sistem dari prespektif pengguna. *Use case* bekerja dengan cara mendekripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan lanngkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendekripsikan urutan kejadian. Setiap kejadian diinisialisasikan oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna diasa disebut *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Notasi *use case*, diagram *use case* menunjukan 3 aspek dari sistem yaitu: *actor*, *use case*, dan sistem/ subsistem *boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomuunikasi dengan *use case*.

Adapun simbol *Use Case Diagram* adalah sebagai berikut:

**Table 2.4** Simbol *Use Case*

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | *System Bondary* menggambarkan batasan anatara sistem dengan aktor |
|  | Simbol ini menggambarkan interaksi anatara aktor dengan sistem. |
|  | Aktor menggambarkan pengguna sistem, dapat berupa manusia atau sistem *terotomatisasi* lain yang berinteraksi dengan sistem lain untuk berbagi, mengirim, dan menerima informasi. |
|  | Menggambarkan hubungan antar aktor dan *use case*. |

* + 1. ***Sequance Diagram***

*Sequance diagram*  digunakan untuk menggambarkan perilaku *scenario*. Diagram ini menunjukan sebuah contoh objek dengan *message* (pesan) yang diletakkan diantara objek-objek ini didalam *use case*.

Komponen utama *sequance diaigram* terdiri atas objek yang dituliskan dengan kotak segi empat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*.

Gambar dibawah ini menunjukan esensi simbil dari *sequance diagram* dan simbol kerjanya secara bersama-sama.

**Table 2.5** Simbol *Sequance Diagram*

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | *Object Lifeline*: menggambarkan *object* apa saja yang terlibat. |
| Description: C:\Users\Home\Desktop\actor.jpg | *Actor*: menggambarkan hubungan *actor* yang terlibat. |
|  | *Activation*:menggambarkan hubungan antara *object* dengan *message*. |
|  | *Message(call)*: menggambarkan alur *message* yang merupakan kejadian objek pengirim *lifeline* ke objek penerima *lifeline.* |
|  | *Message(return)*: menggambarkan alur pengambilan *message* ke objek pemanggil dan tanda bahwa objek penerima telah menyelesaikan prosesnya. |

* + 1. ***Activity Diagram***

*Activity diagram* adalah teknik untuk menggambarkan logika prosedurial, proses bisnis dan aliran kerja dalam berbagai kasus. Dalam berbagai hal diagram ini memainkan peran seperti diagram alir atau seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *parallel* sedangkan *flowchart* tidak bisa..

Simbol berikut adalah simbol yang sering digunakan pada saat pembuatan *activity diagram*.

**Table 2.6** Simbol *Activity Diagram*

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | *Initial state*menggambarkan awal dari sebuah proses |
|  | *State*, segi empat bersudut tumpul menggambarkan sebuah kegiatan atau tugas yang perlu dilakukan. |
|  | *Control flow*,panah menggambarkan sasaran yang mengawali kegiatan. |
|  | *Transition* (*fork*), merupakan bar sinkronisasi dimana kegiatan dapat dilakukan secara paralel |
|  | *Transition (join),*merupakan bar sinkronisasi dimana dua aktivitas yang mempunyai tujuan yang sama dan tujuan dari keduanya digabung menjadi satu. |
|  | *Decision*, menggambarkan sebuah kegiatan keputusan. |
|  | *Final state*menggambarkan akhir dari sebuah proses |

* 1. **Teknik Pengujian Sistem**
     1. ***White******Box***

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain dan pengkodean.

Pengujian sistem / perangkat lunak memiliki sejumlah aturan yang berfungsi sebagai sasaran pengujian, diantaranya adalah sebagai berikut :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Pengujian *White Box*, adalah metode pengujian yang menggunakan struktur kontrol desain prosedur untuk memperoleh *test case*. Dengan menggunakan metode *white box*, perekayasa sistem dapat melakukan *test case* yang memberikan jaminan bahwa semua jalur independen pada suatu modul telah digunakan paling tidak satu kali, menggunakan semua keputusan logis pada sisi *true* dan *false*, mengeksekusi semua *loop* pada batasan mereka dan pada batas operasional mereka, dan menggunakan stuktur data internal untuk menjamin validitasnya. Pengujian *Basis Path* adalah teknik pengujian *white box* yang diusulkan pertama kali oleh Tom McCabe. Metode *basis path* ini memungkinkan desainer *test case* mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. (Roger S. Pressman, 2002 : 536).

**11**

**9**

**10**

**5**

**4**

**1**

**2**

**3**

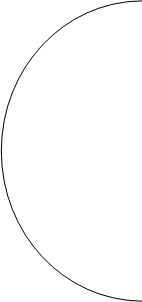
**6**

**7**

**8**

**Gambar 2.10** Bagan Alir

Bagan alir digunakan untuk menggambarkan struktur kontrol program. Dan untuk menggambarkan grafik alir, harus memperhatikan representasi desain prosedural pada bagan alir. Pada gambar dibawah ini, grafik alir memetakan bagan alir tersebut ke dalam grafik alir yang sesuai (dengan mengasumsikan bahwa tidak ada kondisi senyawa yang diisikan di dalam diamond keputusan dari bagan alir tersebut). Masing-masing lingkaran, yang disebut *simpul* grafik alir, merepresentasikan satu atau lebih statemen prosedural. Urutan kotak proses dan permata keputusan dapat memetakan simpul tunggal. Anak panah tersebut yang disebut *edges* atau *links*, merepresentasikan aliran kontrol dan analog dengan anak panah bagan alir. *Edge* harus berhenti pada suatu simpul, meskipun bila simpul tersebut tidak merepresentasikan statemen prosedural. (Roger S. Pressman, 2002 : 536).

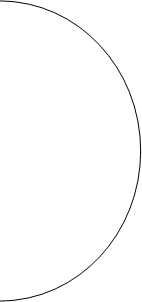


*Edge*

Simpul

Predikat

*Node*



Region

**R1**

**R4**

**R2**

**R3**

**Gambar 2.11** Grafik Alir

* *Node* adalah lingkaran yang merepresentasikan satu atau lebih statemen prosedural.
* *Edge* adalah anak panah pada grafik alir.
* *Region* adalah area yang membatasi *edge* dan *node*
* Simpul Predikat adalah simpul atau *node* yang berisi kondisi yang ditandai dengan dua atau lebih *edge* yang berasal darinya.

Dari gambar *flowgraph* di atas didapat:

*Path* 1 =1– 11

*Path* 2 =1– 2 – 3 – 4 – 5 – 10– 1–11

*Path* 3 =1– 2 – 3 – 6 – 8 – 9 – 10– 1 – 11

*Path* 4 =1– 2 – 3 – 6 – 7 – 9–10–1–11

*Path* 1,2,3,4 yang telah didefinisikan diatas merupakan *basis set* untuk diagram alir.

*Cyclomatic complexity* digunakan untuk mencari jumlah *path* dalam satu *flowgraph*.

Dapat dipergunakan rumusan sebagai berikut:

1. Jumlah region grafik alir sesuai dengan *cyclomatic complexity*.
2. *Cyclomatix complexity*V(G) untuk grafik alir dihitung dengan rumus:

***V(G) =E– N +2*** …………………. (2.2)

Dimana:

E= jumlah *edge* pada grafik alir

N= jumlah *node* pada grafik alir

Jalur 3 : 1 – 2 – 3 – 6 – 8 – 9 – 10 – 1 – 11

Jalur 4 : 1 – 2 – 3 – 6 – 7 – 9 – 10 – 1 – 11

Jalur 1, 2, 3, dan 4 yang ditentukan di atas terdiri dari sebuah basis set untuk grafik alir pada gambar 2.16. Bagaimana kita tahu banyaknya jalur yang dicari? Komputasi kompleksitas siklomatis memberikan jawaban. Fondasi kompleksitas siklomatis adalah teori grafik, dan memberi kita metriks perangkat lunak yang sangat berguna. Kompleksitas dihitung dalam salah satu dari tiga cara berikut:

1. Jumlah region grafik alir sesuai dengan kompleksitas siklomatis.
2. Kompleksitas siklomatis, *V(G),* untuk grafik alir *G* ditentukan sebagai

V(G) = E – N + 2 di mana *E* adalah jumlah *edge* grafik alir dan *N* adalah jumlah simpul grafik alir.

1. Kompleksitas siklomatis, *V(G),* untuk grafik alir G juga ditentukan sebagai *V(G)* = P + 1, dimana P adalah jumlah simpul predikat yang diisikan dalam grafik alir *G*.

Pada gambar 2.16 grafik alir, kompleksitas siklomatis dapat dihitung dengan menggunakan masing-masing dari algoritma yang ditulis di atas:

1. Grafik alir mempunyai 4 region
2. *V(G)* = 11 *edge* – 9 simpul + 2 = 4
3. *V(G)* = 3 simpul yang diperkirakan + 1 =4

Dengan demikian, kompleksitas siklomatis dari grafik alir pada gambar 2.16 adalah 4. Yang lebih penting, nilai untuk *V(G)* memberi kita batas atas untuk jumlah jalur independen yang membentuk basis set, dan implikasinya, batas atas jumlah pengujian yang harus didesain dan dieksekusi untuk menjamin semua statemen program.

* + 1. ***Black******Box***

Pengujian *Black-Box* berusaha menemukan kesalahan dalam kategori:

1. pada struktur data (pengaksesan basis data)
2. Kesalahan Fungsi tidak benar atau hilang
3. Kesalahan antar muka
4. Kesalahan inisialisasi dan akhir program
5. Kesalahan performasi.

Pengujian ini berfokus pada persyaratan fungsional perangkat lunak dan merupakan komplemen dari pengujian *White-Box*. Hal tersebut dapat dicapai melalui:

1. Pengujian *Graph-based*: dimulai dengan membuat grafik sekumpulan *node* yang mempresentasikan objek (misal *New File*, Layar baru dengan atributnya), link (hubungan antar objek), *node-weight* (misal nilai data tertentu seperti atribut layar, perilaku), dan link-weight (karakteristik suatu link, misal menu select).
2. *Equivalence Partitioning*: membagi domain *input* untuk pengujian agar diperoleh kelas-kelas kesalahan (misal kelompok data karakter, atau atribut yang lain).
3. Analisis Nilai Batas: pengujian berdasarkan nilai batas domain *input*.

Pengujian Perbandingan: disebut juga pengujian *back-to-back* yang diterapkan pada pada suatu versi perangkat lunak atau perangkat lunak redundan untuk memastikan konsistensinya.

* 1. ***Tools* Pendukung**

Berikut merupakan *tools* pendukung dalam pembuatan sistem pengamanan basis data:

**Tabel 2.7** *Tools* Pendukung Pembuatan sistem

|  |  |  |
| --- | --- | --- |
| **NO** | ***TOOLS*** | **KET** |
| 1. | PHP | Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Bahasa pemrograman ini digunakan untuk pembuatan sistem. |

* 1. **Kerangka Pemikiran**

**Masalah**

1. Bagaimana cara mengimplementasikan algoritma kriptografi simertis *stream cipher* RC4 pada sebuah sistem enkripsi basis data.?
2. Bagaimana merancang aplikasi dengan menampilkan pilihan tabel dan kolom basis data sebelum melakukan proses enkripsi.?
3. Bagaimana pengaruh panjang kunci dan panjang data terhadap kecepatan dan keamanan proses enkripsi/dekripsi algoritma RC4.?

**Peluang**

1. Kurangnya pengaman basisdata
2. Banyaknya pengguna basis data

**Solusi**

Membangun Sistem Pengaman Berbasis Kriptografi

**Pembangunan Sistem**

PHP

**Analisis Sistem**

1. Sistem berjalan
2. Sistem di usulkan

**Desain Sistem**

1. Desain Model (UML)
2. Desain *User Interface*

* Desain menu utama
* Desain *output*
* Desain *input*

**Pengujian Sistem**

1. *White Box*
2. *Black Box*
3. *Performance*

**Implementasi Sistem**

Masyarakat pengguna internet dan basis data

**Tujuan**

1. Untuk membuat pengembangan sistem keamanan aplikasi program dengan menggunakan enkripsi yang lebih baik.
2. Mengimplementasikan teknik kriptografi *stream cipher* RC4 kedalam sebuah aplikasi sistem.

**Gambar 2.11** Bagan Kerangka Pikir

**Gambar 2.12** Kerangka Pemikiran