BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Sistem

Kata sistem mempunyai beberapa pengertian, tergantung dari sudut pandang mana kata tersebut didefinisikan. Secara garis besar ada dua kelompok pendekatan, yaitu (Kusrini dan Kaniyo, 2007) :

1. Pendekatan sistem yang lebih menekankan pada elemen-eleman atau kelompoknya, yang dalam hal ini sistem itu didefinisikan sebagai "Suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelasaikan suatu aturan tertentu".
2. Pendekatan sistem sebagai jaringan kerja dari prosedur, yang lebih menekankan urutan operasi didalam sistem. Prosedur didefinisikan oleh Richard F. Neushl sebagai "Urutan operasi kerja (tulis-menulis), yang biasanya melibatkan beberapa orang di dalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi bisnis yang terjadi.

Pendekatan sistem yang lebih menekankan pada elemen-elemen atau komponennya mendefinisikan sistem sebagai sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Dengan demikian di dalam suatu sistem, komponen-komponen ini tidak dapat berdiri

sendiri-sendiri, tetapi saling berhubungan hingga membentuk satu kesatuan sehingga tujuan sistem itu dapat tercapai.

2.2 Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain adalah (Kusrini dan Koniyo, 2007) :

1. Komponen Sistem (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian dari sistem.

2. Batasan Sistem (*Boundary*)

   Merupakan daerah yang membatasi suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.

3. Subsistem

   Bagian-bagian dari sistem yang beraktivitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

4. Lingkungan Luar Sistem (*Environment*)

   Suatu sistem yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem.

5. Penghubung Sistem (*Interface*)

   Media penghubung antara suatu subsistem dengan subsistem lain. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

6. Masukan Sistem (*Input*)

   Energi yang masuk ke dalam sistem, berupa perawatan dan sinyal. Masukan perawatan adalah energi yang dimasukan supaya sistem tersebut dapat berinteraksi.

7. Keluaran Sistem (*Output)*

   Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukkan untuk sub sistem yang lain atau kepada supra sistem.

8. Pengolahan Sistem (*Process*)

   Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran.

2.3 Pengertian Sistem Informasi

Menurut Sutono (2007), sistem informasi dalam suatu pemahaman yang sederhana dapat didefinisikan sebagai satu sistem berbasis komputer yang menyediakan informasi bagi beberapa pemakai dengan kebutuhan yang serupa. Para pemakai biasanya tergabung dalam suatu entitas organisasi formal, seperti Departemen atau Lembaga suatu Instansi Pemerintahan yang dapat dijabarkan

menjadi Direktorat, Bidang, Bagian sampai pada unit terkecil dibawahnya. Informasi menjelaskan mengenai organisasi atau salah satu sistem utamanya mengenai apa yang telah terjadi di masa lalu, apa yang sedang terjadi sekarang dan apa yang mungkin terjadi dimasa yang datang tentang organisasi tersebut.

Sistem informasi memuat berbagai informasi penting mengenai orang, tempat, dan segala sesuatu yang ada di dalam atau di lingkungan sekitar organisasi. Informasi sendiri mengandung suatu arti yaitu data yang telah diolah ke dalam suatu bentuk yang lebih memiliki arti dan dapat digunakan untuk pengambilan keputusan. Data sendiri merupakan fakta-fakta yang mewakili suatu keadaan, kondisi, atau peristiwa yang terjadi atau ada di dalam atau di lingkungan fisik organisasi. Data tidak dapat langsung digunakan untuk pengambilan keputusan, melainkan harus diolah lebih dahulu agar dapat dipahami, lalu dimanfaatkan dalam pengambilan keputusan. Informasi harus dikelola dengan baik dan memadai agar memberikan manfaat yang maksimal. Penerapan sistem informasi di dalam suatu organisasi dimaksudkan untuk memberikan dukungan informasi yang dibutuhkan, khususnya oleh para pengguna informasi dari berbagai tingkatan manajemen.

Dewasa ini, sistem informasi yang digunakan lebih berfokus pada sistem informasi berbasis komputer (*computer-based information system*). Harapan yang ingin diperoleh di sini adalah bahwa dengan penggunaan teknologi informasi atau sistem informasi berbasis komputer, informasi yang dihasilkan dapat lebih akurat,

berkualitas, dan tepat waktu, sehingga pengambilan keputusan dapat lebih efektif dan efisien (Sutono, 2007).

2.4 Pengertian Data

Menurut Petroutsos (2002), data merupakan bentuk yang masih mentah atau informasi kasar berupa fakta, angka-angka yang belum dapat bercerita banyak, sehingga perlu lebih lanjut, data diolah melalui sebuah model untuk menghasilkan informasi. Siklus Perkembangan Pengolahan Data (*Expended data processing cycle*) yaitu Masukan (*Input)*, Memproses (*Processing*) dan Keluaran (*Output*) dapat ditambahkan tiga atau lebih tahapan lagi yaitu Pengorganisasian (*Organitation*), Penyimpanan (*Storage*), Pendistribusian (*Distribution*).

2.5 *Databases*

Database adalah salah satu koleksi terorganisasi dari data terstruktur, yang disimpan dengan duplikasi item data yang minimum guna memberikan pool (kelompok) data yang konsisten dan terkontrol. Data ini umum bagi semua sistem, namun independen terhadap program yang menggunakan data itu (Sumin dan Soeparlan, 1995).

Database disimpan di dalam tabel, dan tabel mengandung data yang berhubungan, atau *entity*, seperti misalnya orang, produk, pesanan, dan sebagainya. Tujuannya adalah menjaga table tetap kecil dan dapat dikelola, serta *entity-entity* yang terpisah disimpan dalam tabel-tabel tersendiri. Tentu saja *entity* tidak dapat

independen satu sama lain. Di dalam sebuah *database*, setiap tabel memiliki sebuah *field* yang memiliki nilai unik untuk setiap baris (Petroutsos, 2002).

Dalam pengembangan sistem pengarsipan surat pada skripsi ini penulis menggunakan aplikasi *databases MySQL,* adapun beberapa penjelasan tentang *databases MySQL* sebagai berikut :

2.5.1 *MySQL*

MySQL adalah salah satu aplikasi sistem manajemen *databases* relasional yang handal dalam mengelolah *databases* yang sederhana maupun komplek. *MySQL* mempunyai dua macam lisensi yang dikeluarkan oleh *MySQL AB,* suatu perusahaan Swedia, lisensi tersebut yaitu :

1) ***Open Source software :*** *MySQL* tersedia via GNU GPL (*General Public License)* untuk yang gratis.

2) ***Commercial License :*** tersedia bagi siapa saja yang menyukai GPL, jika ingin mengembangkan dan menggunakan *MySQL* sebagai bagian dari *software* produk baru maka pengembang harus membeli *license commercial* ini.

2.5.2 Keunggulan *MySQL*

Dibawah ini beberapa keunggulan dari *databases MySQL (*Sinarmata, 2006) *:*

1) **Cepat :** tujuan utama dari pengembangan MySQL adalah kecepatan dalam mengakses dan mengolah *databases.*

2) **Tidak mahal :** dibawah *Open Source software license* maka siapapun dapat menggunakan aplikasi *MySQL* secara gratis.

3) **Mudah digunakan :** kita dapat membangun dan berinteraksi dengan *databases MySQL* cukup dengan pernyataan sederhana didalam bahasa *SQL*.

4) **Dapat berjalan pada beberapa system operasi :** seperti Windows, Linux, Mac OS, Unix (solaris, AIX, DEC unix) FreeBSD, OS/2, Irix, dan lainnya.

5) **Aman :** *MySQL* adalah sistem oktorisasi fleksibel yang mengijinkan beberapa atau semua privilege *databases* untuk pengguna khusus atau kelompok pengguna.

2.6 Tatalaksana Kearsipan

Sub bab 2.6 ini direferensikan dari buku terbitan PT PLN (Persero) 2004. Tatalaksana surat mengatur cara penerbitan surat sebagai sarana komunikasi kedinasan di lingkungan PT PLN (Persero) secara terpadu guna menyampaikan/memperoleh data dan informasi yang cepat, tepat dan lengkap untuk pengambilan keputusan.

2.6.1 Jenis Surat

Surat sebagai sarana komunikasi kedinasan, berdasarkan jenisnya dibedakan atas surat, produk hukum dan produk media (Adji, 2004).

2.6.1.1 Surat

Berdasarkan ruang lingkupnya, surat dibedakan menjadi tiga jenis yaitu :

1. Surat Ekstern

Surat ekstern adalah surat yang ditujukan satu (tunggal) atau lebih dari satu (kolektif) kepada satuan organisasi, instansi pemerintah, swasta atau perorangan.

2. Surat Intern

Surat intern atau disebut Nota Dinas, adalah surat yang diperuntukkan sebagai sarana komunikasi di dalam lingkungan satuan organisasi dan dipergunakan sesuai dengan hirarki yang berlaku.

3. Surat Khusus

Surat bentuk khusus adalah surat yang dibuat secara sepihak dan mengikat kedua belah pihak berupa nota kesepahaman/*memorandum of understanding (MoU),* surat perjanjian, surat peringatan/teguran, surat pernyataan, surat tugas, surat perintah perjalanan dinas, laporan, formulir, daftar pengantar dan undangan.

2.6.2.2 Produk Hukum

Surat yang memiliki ruang lingkup sebagai dasar hukum dalam pelaksanaan tugas di PT PLN (Persero), berisi ketentuan-ketentuan yang bersifat pengaturan atau penetapan tentang sesuatu hal yang mengikat dan wajib dilaksanakan, baik seluruh atau sebagian satuan Organisasi, swasta maupun perorangan, dibedakan menjadi lima jenis yaitu (Adji, 2004) :

1. Keputusan

Keputusan adalah surat yang berisi pengaturan atau penetapan kebijaksanaan yang dikeluarkan oleh pejabat yang berwenang.

2. Intruksi

Intruksi adalah surat yang memuat perintah dengan petunjuk teknis pelaksanaan suatu kebijakan/ketetepan baik bersumber dari peraturan yang lebih tinggi maupun berdasarkan suatu kebijakan Direksi.

3. Edaran

Edaran adalah surat yang isinya memuat petunjuk atau penjelasan tentang hal-hal yang harus diperhatikan dan dilaksanakan berdasarkan peraturan/ketetapan/keputusan yang ada. Edaran bersifat umum dan berlaku tetap untuk seluruh atau sebagian Satuan Organisasi.

4. Pengumuman

Pengumuman adalah surat yang memuat suatu informasi atau penjelasan yang berlaku umum untuk waktu satu kali atau untuk waktu terbatas/tertentu, sampai isi pengumuman itu diketahui atau untuk dilaksanakan oleh Satuan Organisasi dan seluruh atau sebagian masyarakat di wilayah kerja Satuan Organisasi.

5. Pemberitahuan

Pemberitahuan pada dasarnya sama dengan Pengumuman, tetapi hanya berlaku di lingkungan kerja terbatas, dengan maksud untuk diketahui dan atau dilaksanakan oleh pegawai dalam Satuan Organisasi.

2.6.3 Pemberian Kode

Pemberian kode adalah pembuatan/ penulisan kode surat, yang bertujuan untuk mempermudah identifikasi atau pengenalan surat dalam rangka membantu terlaksananya kegiatan menghimpun, menyimpan dan menyajikan kembali. Pelaksanaan penerbitan nomor surat dipusatkan atau diberikan oleh Unit Tata Usaha, sedangkan untuk Nota Dinas nomor penerbitannya diberikan oleh Unit Pengolah (Adji, 2004).

Kode pokok masalah yang dipergunakan dalam pemberian nomor surat dalam bentuk angka :

Kode angka 0 = Manajemen

Kode angka 1 = Ketenagalistrikan

Kode angka 2 = Penelitian dan Pengembangan

Kode angka 3 = Pendidikan dan Pelatihan

Kode angka 4 = SDM dan Organisasi

Kode angka 5 = Keuangan

Kode angka 6 = Logistik

2.7 Kriptografi

Kriptografi berasal dari bahasa Yunani, *crypto* dan *graphia*. *Crypto* berarti *secret* (rahasia) dan graphia berarti *writing* (tulisan). Menurut terminologinya,

kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain (Ariyus, 2008)

2.7.1 Algoritma Kriptografi

Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu (Ariyus, 2008) :

1. *Enkripsi* : merupakan hal yang penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut plaintext, yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan cipher atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata maka kita akan melihatnya di dalam kamus atau daftar istilah. Beda halnya dengan enkripsi, untuk mengubah teks-asli ke bentuk teks-kode kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.
2. *Dekripsi* : merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks-asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.
3. Kunci : yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, yaitu kunci rahasia (*private key)* dan kunci umum *(public key).*

Keamanan dari algoritma kriptografi tergantung pada bagaimana algoritma itu bekerja. Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakai (Ariyus, 2008) :

1. Algorima Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)

   Algoritma ini disebut algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut dapat melakukan enkripsi dan dekripsi terhadap pesan.

   Algoritma yang memakai kunci simetri diantaranya adalah :

   a. Data Encryption Standar (DES).
   b. RC2, RC4, RC5, RC6
   c. International Data Encryption Algorithm (IDEA)
   d. Advanced Encryption Standard (AES)
   e. One Time Pad (OTP)
   f. A5, dan lain sebagainya.

2. Algorima Asimetri (menggunakan kunci yang berbeda untuk enkripsi dan dekripsinya)

   Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kunci kata yang digunakan untuk melakukan enkripsi dan dekripsi berbeda.

   Algoritma yang memakai kunci publik di antaranya adalah :

   a. Digital Signature Algorithm (DSA)
   b. RSA

c. Diffie-Hellman (DH)

d. Elliptic Curve Cryptography (ECC)

e. Kriptografi Quantum, dan lain sebagainya.

3. Hash Function

Fungsi Hash sering disebut dengan fungsi Hash satu arah (*one-way function), message digest, fingerprint,* fungsi kompresi dan *message authentication cade* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam biner dengan panjang yang tetap. Fungsi Hash biasanya diperlukan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang diinginkan.

2.7.2 Algoritma Twofish

Twofish merupakan algoritrma yang beroperasi dalam mode blok. Algoritma twofish sendiri merupakan pengembangan dari algoritma Blowfish. Tujuan perancangan Twofish yang selaras dengan kriteria NIST (*National Institute of Standars and Technology*) untuk AES (*Advanced Encryption Standard*) adalah sebagai berikut (Ariyus, 2008) :

1. Merupakan blok kode dengan kunci simetri dan blok sepanjang 128 bit.
2. Panjang kunci yang digunakan adalah 128 bit, 192 bit, dan 256 bit.
3. Tidak mempunyai kunci lemah.
4. Efisiensi algoritma, baik pada *Intel Pentium Pro* dan perangkat lunak lainnya serta *platform* perangkat keras.

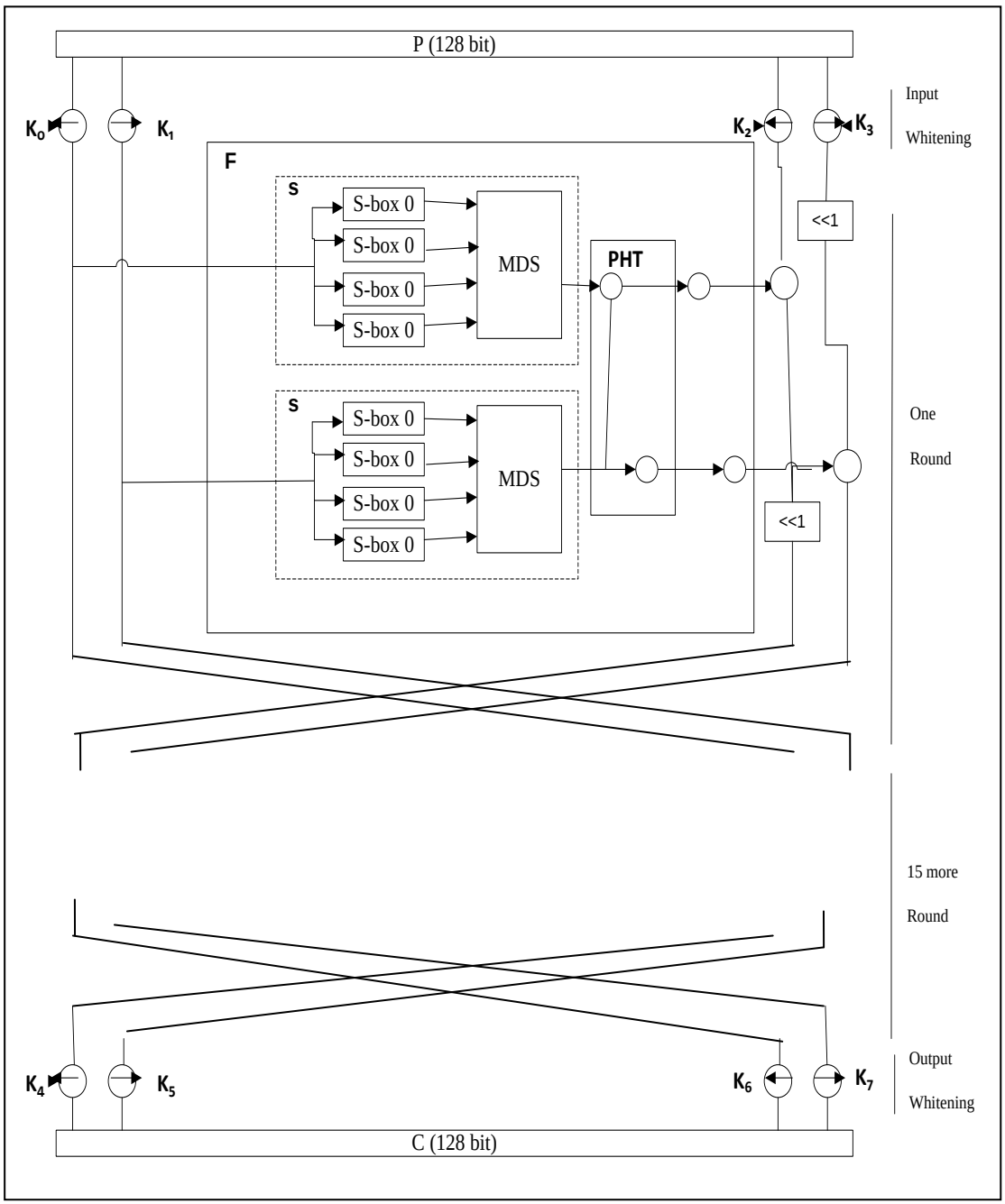
5. Rancangan yang fleksibel, yang dapat diartikan, misalnya, dapat menerima panjang kunci tambahan.
6. Rancangan yang sederhana agar memudahkan proses analisis dan implementasi algoritma.

Selain kriteria-kriteria yang telah disebutkan diatas, pada Twofish juga ditambahkan kriteria performansi berikut (Hassouna, 2013) :

1. Menerima kunci dengan panjang berapapun hingga 256 bit.
2. Mengenkripsikan data dalam waktu kurang dari 500 *clock cycles* per blok pada Intel Pentium, Pentium Pro, dan Pentium II, untuk versi algoritma yang teroptimasi sepenuhnya.
3. Mampu membentuk kunci 128 bit (untuk kecepatan enkripsi yang optimal) dalam waktu yang kurang dari waktu yang dibutuhkan untuk mengenkripsi 32 blok pada Pentium, Pentium Pro, dan Pentium II.
4. Tidak menggunakan operasi yang membuat Twofish tidak efisien pada mikroprosesor selain 32 bit, mikroprosesor 8 bit, dan mikroprosesor 16 bit.

Twofish menggunakan struktur sejenis Feistel dalam 16 putaran dengan tambahan teknik *whitening* terhadap masukan dan keluaran. Teknik *whitening* adalah teknik melakukan operasi XOR terhadap materi kunci sebelum putaran pertama dan sesudah putaran akhir. Elemen di luar jaringan Feistel normal yang terdapat dalam algoritma Twofish adalah rotasi 1 bit. Proses rotasi ini dapat dipindahkan ke dalam fungsi F untuk membentuk struktur jaringan Feistel yang murni, tetapi hal ini membutuhkan tambahan rotasi kata sebelum langkah whitening keluaran.

Blok diagram Twofish dapat dilihat secara global pada Gambar 2.1 (Siddik, 2012) :



Gambar 2.1 Blok Diagram Twofish (Siddik, 2012)

Blok yang membangun Twofish seperti di bawah ini (Siddik, 2012) :

1. Jaringan Feistel

Jaringan Feistel adalah metode umum untuk mentransformasi suatu fungsi menjadi bentuk permutasi.

2. Kotak-S (S-boxes)

Kotak-S (S-boxes) adalah matriks yang berisi substitusi non-linear yang memetakan satu atau lebih bit dengan satu atau lebih bit lain dan digunakan di banyak blok kode.

3. MDS Matrices

Kode MDS (*Maximum Distance Separable)* pada sebuah field adalah pemetaan linear dari x elemen field ke y elemen field, dan menghasilkan vektor komposit x + y elemen, dengan ketentuan bahwa jumlah minimum dari elemen bukan nol pada setiap vektor bukan nol paling sedikit y + 1. Dengan kata lain, jumlah elemen yang berbeda di antara dua vektor berbeda yang dihasilkan oleh pemetaan MDS paling sedikit y + 1.

4. Transformasi Pseudo-Hadamard (PHT)

Transformasi Pseudo-Hadamard (PHT) adalah sebuah operasi pencampuran sederhana yang berjalan secara cepat dalam perangkat lunak PHT 32-bit dengan dua masukan didefinisikan sebagai :

$a' = a + b \bmod 2^{32}$

$b' = a + 2b \bmod 2^{32}$

SAFER menggunakan PHT 8-bit untuk difusinya. Twofish menggunakan PHT 32-bit untuk mengubah keluaran dari fungsi *g*-nya. PHT ini dapat dieksekusi dalam dua *opcodes* di mikroprosesor modern seperti keluarga pentium.

5. Whitening

Whitening adalah sebuah teknik meng-XOR-kan material kunci sebelum putaran pertama dan setelah putaran terakhir.

6. Penjadwalan kunci

Penjadwalan kunci adalah proses pengubahan bit-bit kunci menjadi upa-kunci tiap putaran yang dapat digunakan oleh kode.

2.8 *Visual Basic*

Visual Basic adalah salah satu produk bahasa pemrograman yang dikeluarkan *Microsoft*, salah satu perusahaan *software* terkemuka di dunia. *Visual Basic* merupakan bahasa pemrograman yang mudah digunakan untuk pengembangan sistem, baik itu sistem kecil maupun sistem besar. Dengan banyaknya komponen kontrol yang disediakan oleh *Visual Basic*, membuat *programmer* dan para pengembang sistem lebih mudah dalam pembuatan sistem. *Visual Basic* banyak dipakai oleh *programmer* dan para pengembang sistem, karena kemudahan yang ditawarkan. Dalam pengembangan sistem, para *programmer* tidak terlalu dipusingkan dengan tampilan program, karena *Visual Basic* menyediakan banyak komponen kontrol untuk desain tampilan dari program, dengan *Visual Basic* dapat dikembangkan berbagai jenis sistem, seperti sistem *database*, jaringan internet, multimedia grafik, dan lainnya (Firdaus, 2006).

2.9 *Crystal Reports*

Crystal Reports adalah sebuah program sistem untuk membuat laporan-laporan dalam berbagai bentuk dan dari berbagai sumber data. Sumber data yang dapat

diolah oleh *Crystal Reports* dapat berasal dari sumber data lokal maupun sumber data *remote*, misalnya dari komputer *server*. Sumber data dapat berasal dari berbagai macam program sistem, misalnya *dBase, Delphi, Access, Oracle, SQL Server, MySQL,* dan lain-lain (Alam, 2005).

Crystal Report dapat dioperasikan secara mandiri untuk membuat laporan dari suatu sumber data atau bisa juga dioperasikan dari suatu program sistem, misalnya dari program *Microsoft Visual Basic* atau program *Borland Delphi* (Alam, 2005).

2.10 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah (Sukamto, 2009) :

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.

Karakteristik umum dari pengujian perangkat lunak adalah sebagai berikut (Sukamto, 2009) :

1) Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.
2) Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3) Pengujian diadakan oleh *software developer* dan untuk proyek yang besar oleh *group testing* yang *independent.*
4) *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing*

Metode pengujian perangkat lunak ada 3 jenis, yaitu (Sukamto, 2009) :

1) *White Box/Glass Box* - pengujian operasi
2) *Black Box* - untuk menguji sistem
3) *Use case* - untuk membuat input dalam perancangan *black box* dan pengujian *statebased*

2.11 *Black Box Testing*

Pengujian menggunakan sekumpulan aktifitas validasi, dengan pendekatan *black box testing.* Menurut Shalahuddin dan Rosa (2011), *black box testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai

dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box testing* harus dibuat dengan kasus benar dan kasus salah.

Menurut Pressman (2010), *black box testing* juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program.

1. *Equivalence Partitioning*

Equivalence Partitioning merupakan metode *black box testing* yang membagi domain masukan dari program kedalam kelas-kelas sehingga test case dapat diperoleh. *Equivalence Partitioning* berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk *Equivalence Partitioning* berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi *boolean.*

Kesetaraan kelas dapat didefinisikan menurut panduan berikut (Pressman, 2001) :

1. Jika masukan kondisi menentukan kisaran, satu sah dan dua diartikan tidak valid kesetaraan kelas.
2. Jika masukan membutuhkan nilai, kondisi tertentu satu sah dan dua tidak valid kesetaraan kelas diartikan.
3. Jika masukan kondisi menentukan anggota dari set, satu sah dan satu tidak valid kesetaraan kelas diartikan.
4. Jika kondisi yang input, *boolean* satu sah dan satu tidak valid kelas diartikan.

Sebagai contoh, pemeliharaan data untuk aplikasi bank yang sudah diotomatisasikan. Pemakai dapat memutar nomor telepon bank dengan menggunakan mikro komputer yang terhubung dengan *password* yang telah ditentukan dan diikuti dengan perintah-perintah. Data yang diterima adalah :

1. Kode Area : kosong atau 3 digit
2. Prefix : 3 digit atau tidak diawali 0 atau 1
3. Suffix : 4 digit
4. Password : 6 digit alfanumerik
5. Perintah : *check, deposit,* dll

Selanjutnya kondisi masukan digabungkan dengan masing-masing data elemen, dapat ditentukan sebagai berikut :

1. Kode Area : kondisi masukan, *Boolean*-kode area mungkin ada atau tidak. Kondisi masukan, kisaran nilai ditentukan antara 200-999.

2. Prefix : kondisi masukan kisaran lebih besar 200 atau tidak diawali 0 atau 1.
3. Suffix : kondisi masukan nilai 4 digit.
4. Password : kondisi masukkan *Boolean*-pw mungkin diperlukan atau tidak. Kondisi masukan nilai dengan 6 karakter string.
5. Perintah : kondisi masukan diatur dengan berisi perintah-perintah yang telah didefinisikan.

Menerapkan pedoman untuk derivasi kelas kesetaraan, uji kasus untuk setiap masukan domain item data dapat dikembangkan dan dilaksanakan. Uji kasus dipilih sehingga jumlah terbesar dari atribut dari kelas kesetaraan tersebut dilakukan sekaligus.

Beberapa kata kunci dalam pengujian perangkat lunak yang dapat diperhatikan, yaitu (Simarmata, 2009) :

1. Dinamis

   Pengujian perangkat lunak dilakukan pada masukan yang bervariasi. Masukan ini ditentukan sebelum pengujian dilakukan dengan batasan yang disesuaikan dengan kemampuan perangkat lunak. Masukan tidak harus sesuatu yang dimungkinkan terjadi pada penggunaan program lebih lanjut, melainkan meliputi keseluruhan batasan yang dapat dijangkau perangkat lunak dan dilakukan pemercontohan (*sampling)* secara acak untuk proses pengujian.

2. Terbatas

   Meskipun pengujian dilakukan pada perangkat lunak sederhana sehingga rumit sekalipun, pengujian dilakukan dengan memenuhi batasan-batasan tertentu sesuai dengan kemampuan program. Batasan ini juga diberlakukan pada masukan-masukan yang dipilih untuk pengujian. Tidak semua kemungkinan masukan diujika pada perangkat lunak karena akan memakan waktu yang cukup panjang mengingat begitu banyaknya kemungkinan yang bisa terjadi. Untuk mengatasi hal ini, pemilihan masukan-masukan pada proses pengujian secara acak yang diperkirakan mampu memenuhi kebutuhan pengujian perangkat lunak akan dilakukan.

3. Tertentu

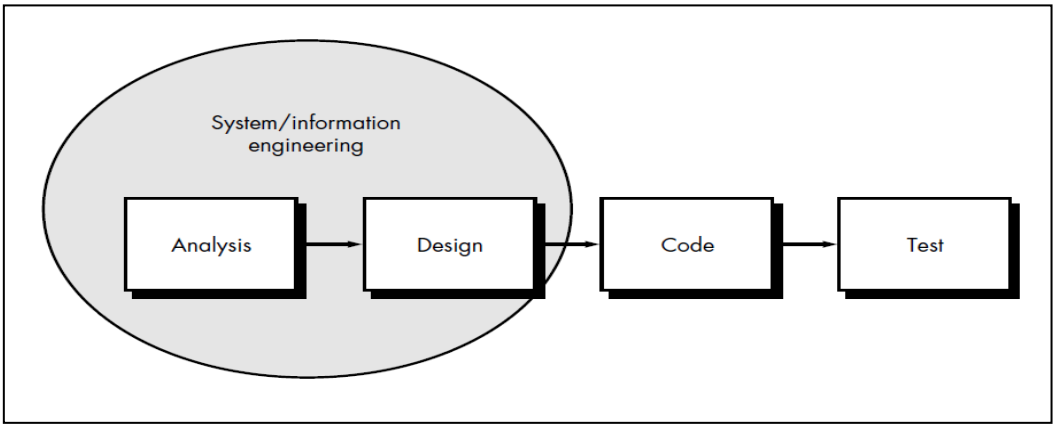
   Pengujian dilakukan dengan batasan tertentu disesuaikan dengan harapan pada fungsi, respon, dan karakteristik perangkat lunak tersebut. Batasan tersebut akan disesuaikan dengan teknik-teknik pengujian yang ada. Pemilihan kriteria pengujian yang paling tepat merupakan hal yang kompleks. Dalam praktiknya, analisis risiko pengujian dan pengalaman terhadaop pengujian-pengujian sejenis akan diperlukan.

4. Harapan

   Kata kunci ini memiliki keadaan-keadaan yang diharapkan, baik berupa respon sistem terhadap masukan maupun karakteristik responnya. Dalam hal ini, batasan-batasan hasil pengujian yang diharapkan harus ditentukan. Dengan demikian, dapat diketahui apakah perangkat lunak tersebut telah memenuhi hasil pengujian yang diharapkan atau memerlukan pembenahan kembali, baik berupa perbaikan maupun pengembangan perangkat lunak.

2.12 Metodologi Pengembangan Sistem

Salah satu metodologi pengembangan sistem adalah *waterfall model*. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Gambar 2.2 menggambarkan tahapan pada model *waterfall* menurut Pressman (2001).



Gambar 2.2 Model *Waterfall* (Pressman, 2001)

Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model *Waterfall* menurut Pressman:

1. *System / Information Engineering and Modeling*

   Permodelan ini diawali dengan mencari kebutuhan dari keseluruhan sistem yang diaplikasikan ke dalam bentuk *software*. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dan sebagainya. Tahap ini sering disebut dengan *Project Definition*.

2. *Software Requirements Analysis*

   Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang dibuat, maka para *software engineer* (Analis) harus mengerti tentang informasi dari *software*, misalnya fungsi yang dibutuhkan dan *user interface*. Dari dua aktivitas tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada pelanggan.

3. *Design*

   Proses ini digunakan untuk mengubah kebutuhan-kebutuhan *software* menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

4. *Coding*

   Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding.* Tahap ini merupakan implementasi dari tahap *design* yang secara teknis dikerjakan oleh programmer.

5. *Testing / Verification*

   Sesuatu yang dibuat haruslah diujicobakan. Demikian juga dengan *software*. Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya. Menurut Bell *et. al.*, (2006), semua desain peta

harus diuji untuk memastikan bahwa peta mengkomunikasikan pesan yang dimaksud oleh *user*. Penyertaan dalam pengujian harus sebanding dengan konsekuensi dari kesalahan interpretasi (kesalahan dalam menjelaskan isi dalam peta). Pengujian desain dapat berkisar dari yang sederhana melalui pengujian isi peta (seperti navigasi peta) dimaksudkan untuk komunikasi internal saja, untuk pengujian kegunaan yang lebih menyeluruh dengan perwakilan dari *user* diharapkan memiliki distribusi yang luas.

6. *Maintenance*

   Pemeliharaan suatu *software* diperlukan termasuk pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu saja melainkan membutuhkan pengembangan atas kekurangan yang ditimbulkan oleh *software* tersebut. Ketika dijalankan mungkin saja masih ada *errors* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.13 Alat Pengembangan Sistem

Alat atau metode yang digunakan di tiap tahap pengembangan sistem di dalamnya yaitu:

1) *Use-case Diagram*

Use-case diagram menggambarkan secara grafis perilaku *software* aplikasi. Diagram tersebut memberikan gambaran mengenai *software* aplikasi menurut

perspektif *user* dari *software* aplikasi tersebut. Sebuah *use-case diagram* mengandung (Suhendar dan Gunadi, 2002) :

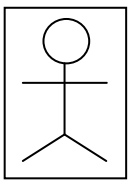
a) *Actor*

b) *Use-case*

c) Interaksi antara *actor* dan *use-case*

a) *Actor*

Actor menggambarkan pengguna *software* aplikasi (*user*). *Actor* membantu memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh, sebuah *actor* dapat memberikan input ke dalam dan menerima informasi dari *software* aplikasi. Perlu dicatat bahwa sebuah *actor* berinteraksi dengan *use-case*, tetapi tidak memiliki kontrol atas *use-case*. Sebuah *actor* mungkin seorang manusia, satu *device hardware*, atau sistem informasi lain. *Actor* dinotasikan seperti Gambar 2.3.



Gambar 2.3 *Actor*

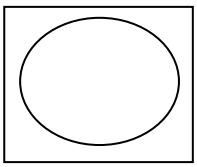
b) *Use-case*

Use-case menggambarkan perilaku *software* aplikasi, termasuk didalamnya interaksi antara *actor* dengan *software* aplikasi tersebut. Secara umum, *use-case* adalah :

1. Pola perilaku *software* aplikasi.
2. Urutan transaksi yang berhubungan yang dilakukan oleh satu *actor* dengan *software* aplikasi
3. Sistem atau "benda" yang memberikan sesuatu yang bernilai kepada *actor*.

Use-case dibuat berdasarkan keperluan *actor*. *Use-case* harus merupakan "apa" yang dikerjakan *software* aplikasi, bukan "bagaimana" *software* aplikasi mengerjakannya. Setiap *use-case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*, Nama *use-case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use-case* yang memiliki nama yang sama. Anda dapat memberikan deskripsi tentang suatu *use-case* dalam jendela dokumentasi untuk memperjelas maksud *use-case* tersebut.

Kadang-kadang anda tidak dapat mencakup semua keperluan suatu *software* aplikasi dalam satu *use-case*. Oleh karena itu, biasanya kita menempatkan dan mengatur sebuah koneksi dari beberapa *use-case* dalam berbagai paket *use-case* (*use-case package*). Secara grafis, *use-case* dinotasikan pada Gambar 2.4



Gambar 2.4 *Use-case*

c) *Use-case* konkret dan *use-case* abstrak

Use-case konkret adalah *use-case* yang dibuat langsung karena keperluan *actor*. *Actor* dapat melihat dan berinisiatif terhadapnya. *Use-case* abstrak adalah *use-case* yang tidak pernah berdiri sendiri. *Use-case* abstrak senantiasa

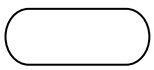
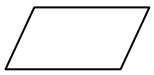
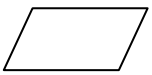
termasuk didalam (*include*), diperluas dari (*extend*), atau memperumum (*generalize*) *use-case* lainnya.

Untuk menggambarkannya dalam *use-case* model biasanya digunakan *association relationship* yang memiliki *stereotype "include"* dan *"extend"*, dan *generalize relationship*. Hubungan *include* menggambarkan bahwa suatu *use-case* seluruhnya meliputi fungsionalitas dari *use-case* lainnya. Hubungan *extend* antar *use-case* berarti bahwa satu *use-case* merupakan tambahan fungsionalitas dari *use-case* yang lain jika kondisi atau syarat tertentu dipenuhi.

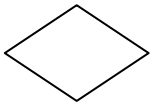
2) *Flowchart*

Flowchart merupakan bagan yang menunjukkan alir di dalam program atau prosedur sistem secara logika. Pada Tabel 2.1 dijelaskan simbol dan kotak *flowchart* atau diagram alur yang paling umum dan sering digunakan sebelum pembuatan program komputer (H.S. dan Sumin, 1997).

Tabel 2.1 Simbol-simbol *Flowchart*

Simbol	Keterangan
	Simbol untuk menyatakan MULAI (*START*) ataupun BERHENTI (*STOP*) atau SELESAI (*END*).
	KOTAK MASUKAN, untuk membaca data yang kemudian diberikan sebagai harga suatu variabel. Juga berfungsi untuk menanyakan/meminta data untuk dijadikan harga suatu variabel, kadang-kadang digunakan kotak.
	KOTAK PENUGASAN, untuk memberi harga kepada suatu variabel, atau untuk melakukan perhitungan matematika yang hasilnya diberikan sebagai harga suatu variabel.
	KOTAK KELUARAN, untuk mencetak (dan/atau menyimpan) hasil/keluaran. Catatan : banyak orang menggunakan kotak selain sebagai masukan juga sebagai keluaran.

Tabel 2.1 Simbol-simbol *Flowchart* (Lanjutan)

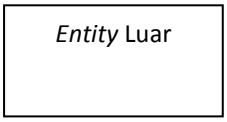
	KOTAK KEPUTUSAN, untuk memutuskan arah atau percabangan yang diambil sesuai dengan kondisi yang saat itu terjadi, BENAR atau SALAH.
	Simbol penghubung, untuk penghubung bila diagram alur terputus disebabkan misalnya oleh pergantian halaman (tak cukup digambar satu halaman)

3) *Data Flow Diagram* (DFD)

Ada beberapa simbol DFD yang dipakai untuk menggambarkan data beserta proses transformasi data, antara lain (Kristanto, 2003) :

a. *Entity* luar

Entity luar digambarkan dengan simbol persegi biasa. *Entity* luar merupakan sumber atau tujuan dari aliran data dari atau ke sistem. *Entity* luar merupakan lingkungan luar sistem, jadi sistem tidak tahu menahu mengenai apa yang terjadi di *entity* luar. *Entity* luar bisa digambarkan secara fisik dengan sekelompok orang atau mungkin sebuah sistem. Bentuk *entity* luar dapat dilihat pada Gambar 2.5.



Gambar 2.5 *Entity* Luar

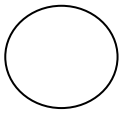
b. Aliran data

Menggambarkan aliran data dari satu proses lainnya. Adapun simbol dari aliran data bentuk garisnya boleh bebas seperti terlihat pada Gambar 2.6

Gambar 2.6 Aliran Data

c. Proses

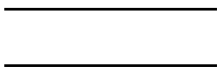
Proses atau fungsi yang mentransformasikan data secara umum digambarkan dengan lingkaran. Bentuk proses dapat dilihat pada Gambar 2.7



Gambar 2.7 Proses

d. Berkas atau tempat penyimpanan

Merupakan komponen yang berfungsi untuk menyimpan data atau file. Simbol dari berkas ini dapat digambarkan dengan garis parallel seperti terlihat pada Gambar 2.8



Gambar 2.8 Tempat Penyimpanan

***4) The Entity Relationship Diagram* (ERD)**

ERD adalah gambar atau diagram yang menunjukkan informasi dibuat, disimpan, dan digunakan dalam sistem bisnis. Entitas biasanya menggambarkan jenis informasi yang sama. Dalam entitas digunakan untuk menghubungkan antar entitas yang sekaligus menunjukkan hubungan antar data. Pada akhirnya ERD bias juga digunakan untuk menunjukkan aturan-aturan bisnis yang ada pada sistem informasi yang dibangun.

Seperti data flow diagram, ERD juga menggunakan simbol-simbol khusus untuk menggambarkan elemen-elemen ERD. Tabel 2.2 menjelaskan simbol-simbol yang digunakan dalam ERD (Al-Fatta, 2007) :

Tabel 2.2 Elemen-elemen dari ERD

	IDEFIX	Chen	Information Engineering
Entitas : Orang, tempat, atau benda Memiliki nama tunggal Ditulis dengan huruf besar Berisi lebih dari 1 instance	*ENTITY NAME* *Identifier*	*ENTITY NAME*	*ENTITY NAME* **Identifier*
Attribute *:* Properti dari entitas Harus digunakan minimal oleh 1 proses bisnis Dipecah dalam detail	*ENTITY NAME* *Attribute-name* *Attribute-name*	*Attribute-name*	*ENTITY NAME* *Attribute-name* . ..
Relationship *:* Menunjukkan hubungan antar 2 entitas Dideskripsikan dengan kata kerja Memiliki modalitas (null/not null) Memiliki kardinalitas (1:1, 1:N, atau M:N)	*Relationship-name*	*Relationship-name*	*Relationship-name*

Berikut penjelasan dari elemen-elemen dari ERD **:**

Entitas :

Entitas bisa berupa orang, kejadian, atau benda dimana data dikumpulkan. Untuk menjadi sebuah entitas, suatu obyek harus menampilkan beberapa kali *event*.

Attribute **:**

a. Informasi yang diambil tentang sebuah entitas.

b. Hanya yang digunakan oleh organisasi yang dimasukkan dalam model.
c. Nama atribut harus merupakan kata benda.
d. Kadang nama entitas diletakkan di depan nama atribut untuk ketelitian.

Identifier :

a. Satu atau lebih atribut dapat menjadi identifier entitas, yang secara unik mengidentifikasikan setiap anggota dari entitas.
b. *Concatenated identifier* (*identifier* gabungan) terdiri dari beberapa atribut.
c. *Identifier* bisa jadi artificial, seperti dengan membuat nomor ID.
d. *Identifier* tidak dikembangkan sampai fase desain.

***Relationship* :**

a. Hubungan antar entitas.
b. Entitas pertama dalam *relationship* disebut entitas induk, entitas kedua disebut dengan entitas anak. *Relationship* harus memiliki nama yang berupa kata kerja.
c. *Relationship* berjalan 2 arah.

Kardinalitas :

a. Kardinalitas mengacu pada beberapa kali *instance* dari suatu entitas dapat berelasi dengan *instance* lain di entitas yang berbeda.
b. Satu *instance* dalam suatu entitas mengacu pada satu dan hanya satu *instance* pada entitas lainnya (1:1).

c. Satu *instance* dalam suatu entitas mengacu ke satu atau lebih *instance* yang berelasi (1:N).
d. Satu atau lebih *instance* dalam suatu entitas mengacu pada satu atau lebih *instance* pada entitas yang berelasi (M:N).

Modalitas :

a. Mengacu pada apakah suatu *instance* dari entitas anak dapat ada tanpa suatu relasi dengan *instance* dari entitas induk atau tidak.
b. *Not Null*, berarti bahwa suatu *instance* pada entitas yang berelasi harus ada untuk suatu *instance* dari entitas lain untuk disebut *valid*.
c. *Null*, berarti bahwa tidak ada *instance* dalam entitas yang berelasi yang diplerlukan untuk *instance* pada relasi lain untuk dikatakan *valid*.