BAB II

LANDASAN TEORI

2.1 Konsep Dasar Algoritma

2.1.1 Sejarah Algoritma

Kata algoritma berasal dari latinisasi nama seorang ahli matematika dari Uzbekistan yaitu Al Khawārizmi (hidup sekitar abad ke-9), sebagaimana tercantum pada terjemahan karyanya dalam bahasa latin dari abad ke-12 "*Algorithmi de numero Indorum*". Pada awalnya kata algorisma adalah istilah yang merujuk kepada aturan-aturan aritmetis untuk menyelesaikan persoalan dengan menggunakan bilangan numerik arab (sebenarnya dari India). Pada abad ke-18, istilah ini berkembang menjadi algoritma, yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan.

2.1.2 Definisi Algoritma

Dalam matematika dan komputasi, algoritma atau algoritme merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik. Algoritma sering mempunyai langkah pengulangan

(iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya selesai. *Programmer* komputer akan lebih mudah menuangkan prosedur komputasinya atau urutan langkah proses dengan terlebih dahulu membuat gambaran (diagram alur) diatas kertas.

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

Algoritma dapat dituliskan dengan menggunakan banyak macam notasi, contohnya bahasa natural atau bahasa sehari-hari, pseudocode, diagram alir, bahasa pemrograman dan *control tables.* Penggunaan bahasa sehari-hari untuk menyatakan suatu urutan algoritma akan menimbulkan banyak keambiguan terhadap pernyataan yang dibuat, oleh karenanya pseudocode, diagram alir dan *control tables* menjadi lebih umum digunakan untuk menuliskan algoritma karena dapat menghindari keambiguan.

Pertimbangan dalam pemilihan algoritma adalah algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli serumit apapun algoritma, jika memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan lain yang harus diperhatikan adalah seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk

menyelesaikan masalah yang memerlukan aproksimasi hasil. Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Selanjutnya adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar, tetapi jika harus menunggu lama untuk mendapatkan keluarannya, algoritma tersebut bukanlah algoritma yang baik, setiap orang menginginkan keluaran yang cepat.

Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya diharapkan keluaran yang sama.

2.2 Konsep Dasar Algoritma Kriptografi

2.2.1 Kriptografi

Menurut Bruce Schneier pada bukunya *Applied Cryptography*, Kriptografi adalah ilmu dan sekaligus seni untuk menjaga kerahasiaan berita. Selain pengertian tersebut, kriptografi juga sering diartikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data [A. Menezes, P. van Oorschot and S. Vanstone - Handbook of Applied Cryptography]. Tidak semua aspek keamanan informasi ditangani oleh kriptografi.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. Autentikasi, adalah berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. Non Repuidasi adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/membuat.

2.2.2 Algoritma Kriptografi

Algoritma sandi (algoritma kriptografi) adalah algoritma yang berfungsi untuk melakukan tujuan kriptografis. Menurut Shannon, Algoritma kriptografi harus memiliki kekuatan untuk melakukan :

1. Konfusi / pembingungan (*confusion*), dari teks terang sehingga sulit untuk direkonstruksikan secara langsung tanpa menggunakan algoritma dekripsinya.
2. Difusi / peleburan (*difusion*), dari teks terang menjadi *ciphertext* sehingga karakteristik dari teks terang tersebut hilang.

Pada implementasinya sebuah algoritma kriptografi harus memperhatikan kualitas layanan / *Quality of Service* atau QoS dari keseluruhan sistem dimana dia diimplementasikan. Algoritma kriptografi yang handal adalah algoritma kriptografi yang kekuatannya terletak pada kunci dan distribusi kunci, bukan pada kerahasiaan algoritma itu sendiri. Teknik dan metode untuk menguji kehandalan algoritma kriptografi adalah kriptanalisa.

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen teks terang / *plaintext* dan yang berisi elemen teks sandi / *ciphertext*. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. Apabila elemen-elemen teks terang dinotasikan dengan P, elemen-elemen teks sandi dinotasikan dengan C, sedang untuk proses enkripsi dinotasikan dengan E, dekripsi dengan notasi D.

Enkripsi : $E(P) = C$

Dekripsi : $D(C) = P_{\text{atau}}\, D(E(P)) = P$

Secara umum berdasarkan kesamaan kuncinya, algoritma kriptografi dibedakan menjadi :

1. Kunci-simetris / *symetric-key*, kunci yang digunakan untuk melakukan enkripsi sama dengan kunci yang digunakan untuk dekripsi

2. Kunci-asimetris / *asymetric-key* , kunci enkripsi dan kunci dekripsi tidak sama

Berdasarkan kerahasiaan kuncinya, algoritma kriptografi dibedakan menjadi :

1. Algoritma kriptografi kunci rahasia *secret-key*
2. Algoritma kriptografi kunci publik *publik-key*

2.2.3 Algoritma Kriptografi Kunci-Simetris

Skema algoritma kriptografi akan disebut kunci-simetris apabila untuk setiap proses enkripsi maupun dekripsi data secara keseluruhan digunakan kunci yang sama. Skema ini berdasarkan jumlah data per proses dan alur pengolahan data didalamnya dibedakan menjadi dua kelas, yaitu *block-cipher* dan *stream-cipher.*

1. Block-Cipher

Block-cipher adalah skema algoritma kriptografi yang akan membagi teks terang yang akan dikirimkan dengan ukuran tertentu (disebut blok) dengan panjang t, dan setiap blok dienkripsi dengan menggunakan kunci yang sama. Pada umumnya, *block-cipher* memproses teks terang dengan blok yang relatif panjang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci.

2. Stream-Cipher

Stream-cipher adalah algoritma kriptografi yang mengenkripsi data persatuan data, seperti bit, byte, nible atau per lima bit(saat data yang di enkripsi

berupa data Boudout). Setiap mengenkripsi satu satuan data digunakan kunci yang merupakan hasil pembangkitan dari kunci sebelum.

Beberapa contoh algoritma yang menggunakan kunci-simetris:

a) DES - *Data Encryption Standard*
b) Twofish
c) AES - *Advanced Encryption Standard*, yang bernama asli rijndael
d) *Message Digest 5* (MD5)

2.2.3.1 Data Encryption Standard (DES)

DES dikembangkan pertama kali pada tahun 1970 di IBM yang kemudian diteruskan oleh National Security Agency (NSA) pada tahun 1976 untuk kemudian dijadikan standar keamanan dalam dunia kriptografi pada tahun yang sama. DES merupakan algoritma kriptografi dari jenis *block-cipher* yang berarti membagi sebuah deret menjadi blok untuk kemudian dienkripsi.Panjang kunci yang bisa dibentuk oleh DES adalah 56 bit.

2.2.3.2 Advanced Encryption Standar (AES)

AES merupakan algoritma lanjutan dari DES sebagai algoritma standar internasional untuk menggantikan DES yang telah dipecahkan pada tahun 2001. AES menyerupai DES yang merupakan algoritma *block cipher* dimana algoritma ini akan membagi teks menjadi blok yang telah ditetapkan panjang bit nya. Jika pada DES panjang kunci hanya maksimal 56bit, sedangkan pada AES panjang kunci terbagi menjadi 3 jenis yaitu 128bit dengan 10 perulangan algoritma, 192bit

dengan 12 kali perulangan algoritma dan 256bit dengan 14 kali perulangan algoritma.

2.2.3.3 Message Digest 5 (MD5)

MD5 merupakan algoritma kriptografi untuk pengesahan dokumen, *hash*. MD5 menghasilkan 128bit nilai hash dengan masing-masing blok maksimal 32bit dan blok terbanyak adalah 16blok. MD5 hingga saat ini masih banyak digunakan dalam autentikasi dokumen meskipun telah dikembangan *algorithm hash function* yang baru seperti SHA-1.

2.2.3.4 Twofish

Twofish adalah algoritma simetris rancangan Bruce Schneier dengan panjang kunci sama dengan AES yaitu 128bit, 192bit dan 256bit dengan perulangan algoritma sebanyak 16 kali. Twofish merupakan salah satu nominasi dalam *Advanced Encryption Standard Contest* namun tidak terpilih.

2.2.4 Algoritma Sandi Kunci-Asimetris

Skema ini adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Skema ini disebut juga sebagai sistem kriptografi kunci publik karena kunci untuk enkripsi dibuat untuk diketahui oleh umum (*public-key*) atau dapat diketahui siapa saja, tapi untuk proses dekripsinya hanya dapat dilakukan oleh yang berwenang yang memiliki kunci rahasia untuk mendekripsinya, disebut *private-key*. Dapat dianalogikan seperti kotak pos yang

hanya dapat dibuka oleh tukang pos yang memiliki kunci tapi setiap orang dapat memasukkan surat ke dalam kotak tersebut. Keuntungan algoritma model ini, untuk berkorespondensi secara rahasia dengan banyak pihak tidak diperlukan kunci rahasia sebanyak jumlah pihak tersebut, cukup membuat dua buah kunci, yaitu kunci publik bagi para korensponden untuk mengenkrip pesan, dan kunci privat untuk mendekrip pesan. Berbeda dengan skema kunci-simetris, jumlah kunci yang dibuat adalah sebanyak jumlah pihak yang diajak berkorespondensi. Beberapa contoh algoritma yang menggunakan kunci-asimetris antara lain:

a) Knapsack
b) RSA - Rivert-Shamir-Adelman
c) Diffie-Hellman

2.2.4.1 Knapsack

Algoritma kriptografi Knapsack ditemukan oleh Mekle-Hellman pada tahun 1978. Knapsack merupakan algoritma kriptografi yang mirip dengan RSA namun tidak memiliki fungsi *digital signature*. Panjang kunci pada algoritma knapsack sama dengan algoritma asimeteris lainnya yang dapat diatur oleh pengguna. pada tahun 1984, algoritma knapsack dipecahkan oleh Adi Shamir dengan metode *polynomial-time algorithm*.

2.2.4.2 Diffie-Hellman

Algoritma kriptografi Diffie-Hellman merupakan algoritma kriptografi asimetris dimana kedua pasangan kunci didistribusikan secara menyilang antara

pengirim dan penerima pesan. Diffie-Hellman disebut juga algoritma pertukaran kunci yang ditemukan pada tahun 1976 oleh Whitfield Diffie dan Martin Hellman, Hellman sendiri merupakan salah satu tokoh yang juga menyusun algoritma kriptografi Knapsack. Pemecahan algoritma Diffie-Hellman dapat ditemukan dengan menyesuaikan pada *discrete logarithms problem*.

2.2.4.3 Algoritma Kriptografi RSA (Rivest , Shamir, Adleman)

1. Sejarah dan Definisi RSA

Algortima RSA dijabarkan pada tahun 1977 oleh tiga orang : Ron Rivest, Adi Shamir dan Len Adleman dari *Massachusetts Institute of Technology*. Huruf RSA itu sendiri berasal dari inisial nama mereka (Rivest—Shamir—Adleman). RSA merupakan bagian dari algoritma kriptografi asimetris. Algoritma RSA masih digunakan hingga pada saat ini.

Menurut M. Zaki Riyanto dan Ardhi Ardhian (*Kriptografi Kunci Publik : Sandi RSA*, 2008) : Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar.

Pada algoritma RSA, panjang kunci tidak ditentukan didalam algoritmanya sendiri, penentuan dua buah nilai prima yang nantinya menentukan hasil dari panjang kunci. Panjang kunci akan menjadi acuan dalam proses enkripsi termasuk didalamnya ketika membagi blok. Pada algoritma RSA, fleksibilitas panjang kunci dapat diatur sedemikian rupa sehingga menguntungkan dalam pengimplementasiannya karena dapat menyesuaikan dengan faktor pendukung yang lain seperti dukungan perangkat keras, perangkat lunak dan lain sebagainya.

2. Algoritma Pembentukan Kunci, Enkripsi dan Dekripsi di RSA

Algoritma pembentukan kunci algoritma RSA adalah sebagai berikut :

1. Tentukan *p* dan *q* yang bernilai dua bilangan Prima besar, p ≠q, acak dan dirahasiakan.

   Syarat p ≠ q, jika nilai p=q maka akan lebih mudah untuk mendapatkan bilangan komposit dari nilai modulus yang nantinya akan dibentuk dari formula n = p*q, dengan mencari akar dari bilangan modulus itu sendiri.

2. Menghitung *(RSA) modulus* diwakilkan oleh variabel n

   Formula : $n = p*q$.

3. Menghitung nilai *(RSA) enciphering exponent* yang diwakilkan variabel (e), merupakan pasangan kunci enkripsi.

   Formula e : GCD(e, m) = 1, disebut juga "e relatif prima dengan m"

   Gcd (*greatest common divisor*) adalah Faktor Pembagi Terbesar (FPB). Dengan syarat 1 < e < m .

   Berarti untuk mencari nilai e terlebih dahulu harus didapatkan nilai m.

   Formula pencarian nilai m : m=(p-1) * (q-1).

4. Menghitung nilai *(RSA) deciphering exponent* diwakilkan pada variabel d yang berfungsi sebagai pasangan kunci dekripsi.

   Menghitung bilangan khusus d, syarat $1 < d < \iota(n)$

   d ≡ e-1 mod m

   e*d ≡ 1 (mod m)

   e*d ≡ 1 + (k*m) untuk nilai k adalah integer sembarang.

5. Nilai (n,e) adalah nilai yang boleh dipublikasi.

   Nilai d, p, q, m adalah nilai yang harus dirahasiakan.

   Pasangan (n,e) merupakan kunci publik.

   Pasangan (n,d) merupakan kunci rahasia.

Setelah kedua kunci didapatkan, maka langkah berikutnya adalah mengenkrip data atau pesan (M) menjadi Ciphertext (C), langkah untuk mengenkrip data sebagai berikut :

1. Ambil kunci publik yg otentik (n, e)
2. Representasikan *message* sebagai integer M dalam interval [0,n-1]
3. Hitung C = M ^ e (mod n)

Sedangkan untuk melakukan dekripsi menggunakan kunci *private* (d), dilakukan langkah dengan formula :

M = C^(d) (mod n).

2.3 Oracle

2.3.1 Oracle Corporation

Oracle Corporation adalah salah satu perusahaan utama pengembang sistem manajemen basis data (*database management system*), perangkat lunak untuk pengembangan basis data, pengembangan *software Enterprise Resource Planning (ERP),* pengembangan *software Customer Relationalship Management (CRM)* dan *Supply Chain Management* (*SCM)*. *Oracle Corporation* didirikan pada tahun 1977 yang bermarkas di Redwood City, California. Sebelum namanya menjadi *Oracle Corporation*, perusahaan ini beberapa kali mengganti nama perusahaannya, dimulai dari *Software Development Laboratories* di tahun 1977,

kemudian *Relational Software, Inc*. di tahun 1979, *Oracle Systems* di tahun 1982 hingga akhirnya diubah menjadi *Oracle Corporation* di tahun 1995 sampai dengan sekarang. Salah satu produk dari *Oracle Corporation* yang banyak digunakan adalah *Oracle Database* yang merupakan *Database Management System (DBMS).*

Pengertian DBMS sendiri cukup banyak, beberapa contohnya sebagai berikut :

1. Menurut C.J. Date : DBMS adalah merupakan software yang meng-*handle* seluruh akses pada database untuk melayani kebutuhan user.
2. Menurut S, Attre : DBMS adalah *software, hardware, firmware* dan *procedure-procedure* yang mengelola basis data. *Firmware* adalah s*oftware* yang telah menjadi modul yang tertanam pada *hardware (ROM)*
3. Menurut Gordon C. Everest : DBMS adalah manajemen yang efektif untuk mengorganisasi sumber daya data.

2.3.2 RDMS Oracle

Basis data Oracle adalah basis data relasional yang terdiri dari kumpulan data dalam suatu sistem manajemen basis data. Relational *Database Management System (RDMS)* adalah sebuah *DataBase Management System* dimana data yang tersimpan didalam tabel tertentu terhubung dengan data yang lainnya yang tersimpan didalam tabel juga. Perusahaan perangkat lunak Oracle memasarkan jenis basis data ini untuk bermacam-macam aplikasi yang bisa berjalan pada banyak jenis dan merk perangkat keras komputer (*platform*).

2.4 PL/SQL Oracle 10g

PL/SQL (*Procedural Language/Structured Query Language*) merupakan bahasa standar untuk mengakses *database relational. Syntax* umum pada PL/SQL berasal dari ADA dan PASCAL. PL/SQL adalah salah satu dari tiga bahasa yang ditanamkan didalam oracle itu sendiri selain Java dan SQL. Menggunakan SQL memang membuat manipulasi data lebih mudah namun akan menjadi lebih maksimal ketika ditambahkan bahasa prosedural. Oleh karena itu PL/SQL mengintegrasikan konstruksi prosedural dengan SQL. Contohnya untuk melakukan percabangan dan perulangan tidak bisa dilakukan dengan SQL murni, maka dibutuhkanlah PL/SQL, dimana menggabungkan SQL dan bahasa prosedural.

Pada PL/ SQL Oracle mendukung beberapa algoritma kriptografi seperti pada gambar 2.1 yang diambil dari *site* resmi oracle :



DBMS_CRYPTO contains basic cryptographic functions and procedures. To use this package correctly and securely, a general level of security expertise is assumed.

The DBMS_CRYPTO package enables encryption and decryption for common Oracle datatypes, including RAW and large objects (LOBs), such as images and sound. Specifically, it supports BLOBs and CLOBs. In addition, it provides Globalization Support for encrypting data across different database character sets.

The following cryptographic algorithms are supported:

- Data Encryption Standard (DES), Triple DES (3DES, 2-key and 3-key)
- Advanced Encryption Standard (AES)
- MD5, MD4, and SHA-1 cryptographic hashes
- MD5 and SHA-1 Message Authentication Code (MAC)

Block cipher modifiers are also provided with DBMS_CRYPTO. You can choose from several padding options, including PKCS (Public Key Cryptographic Standard) #5, and from four block cipher chaining modes, including Cipher Block Chaining (CBC).

Table 24-1 lists the DBMS_CRYPTO package features in comparison to the other PL/SQL encryption package, the DBMS_OBFUSCATION_TOOLKIT.

Table 24-1 DBMS_CRYPTO and DBMS_OBFUSCATION_TOOLKIT Feature Comparison

Package Feature	DBMS_CRYPTO	DBMS_OBFUSCATION_TOOLKIT
Cryptographic algorithms	DES, 3DES, AES, RC4, 3DES_2KEY	DES, 3DES
Padding forms	PKCS5, zeroes	none supported
Block cipher chaining modes	CBC, CFB, ECB, OFB	CBC
Cryptographic hash algorithms	MD5, SHA-1, MD4	MD5
Keyed hash (MAC) algorithms	HMAC_MD5, HMAC_SH1	none supported
Cryptographic pseudo-random number generator	RAW, NUMBER, BINARY_INTEGER	RAW, VARCHAR2
Database types	RAW, CLOB, BLOB	RAW, VARCHAR2

Gambar 2.1

Cryptograph Support List Oracle 10g

Diambil dari http:// docs. oracle. com/ cd/ B19306_01 /appdev. 102/ b14258 /d_crypto.htm.

Dapat terlihat bahwa algoritma kriptografi asimetris RSA tidak ada didalam *cryptograph support list*, sehingga pengguna harus menyusun sintak secara manual jika ingin mengimplementasikan RSA pada program aplikasi yang akan dibangun.

2.5 Oracle Form Developer

Oracle Form Developer (OFD) adalah perangkat lunak yang digunakan untuk membuat antarmuka yang menjembatani basis data oracle dengan pengguna. OFD berbasis IDE yang menyediakan navigasi objek, dan *code editor* untuk menggunakan PL/SQL. Pada awalnya OFD dikembangkan hanya untuk menjembatani basis data oracle dari sisi server, namun pada perkembangannya pada *platform* Windows menjadi berfungsi di kedua sisi *client-server*. Dan pada versi selanjutnya ditambahkan *library* java sehingga dapat terintegreasi dengan Java EE.

2.6 Flowchart

Flowchart adalah representasi grafis peta atau dari suatu proses. Langkah-langkah dalam suatu proses yang ditampilkan dengan bentuk simbolik, dan aliran proses ini ditunjukkan dengan panah menghubungkan simbol. Pemrograman komputer *flowchart* dipopulerkan pada 1960-an yang digunakan untuk memetakan logika program. Dalam pekerjaan peningkatan kualitas, *flowchart*

sangat berguna untuk menampilkan bagaimana proses saat ini berfungsi atau bisa idealnya fungsi.

Flowchart dapat membantu melihat apakah langkah-langkah proses telah logis, mengungkap masalah atau miskomunikasi, menentukan batas-batas proses, dan mengembangkan dasar umum pengetahuan tentang proses. *Flowcharting* proses seringkali membawa kepada redudansi ringan, kebuntuan, tapi *flowchart* tidak akan bekerja jika alur tidak akurat. Tujuan pembuatan *flowchart* adalah untuk menggambarkan tahapan penyelesaian masalah secara sederhana, terurai, rapi dan jelas dengan menggunakan simbol yang terstandarisasi.