BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Untuk membuat tampilan yang menarik memang tidak mudah dilakukan. Seorang perancang tampilan selain harus mempunyai jiwa seni yang memadai, ia juga harus mengerti selera pengguna secara umum. Hal lain yang pelu disadari oleh seorang perancang tampilan adalah bahwa ia harus bisa meyakinkan pemrogramnya bahwa apa yang ia bayangkan dapat diwujudkan dengan peranti bantu yang tersedia (Insap Santoso ; 2009 : 185).

Perancangan merupakan proses pengolahan hasil analisis perangkat lunak menjadi rencana pengembangan perangkat lunak dan batasan-batasan perangkat lunak atau masalah yang mungkin dihadapi dalam pengembangan perangkat lunak. Perancangan yang dilakukan meliputi perancangan arsitektur, perancangan modul, dan perancangan antarmuka. (Y. Yohakim Marwanta ; 2010 : 5).

Bagi perancang antarmuka, hal yang sangat penting untuk ia perhatikan adalah mendokumentasikan semua pekerjaan yang dilakukan. Dokumentasi rancangan dapat dikerjakan atau dilakukan dengan beberapa cara :

1. Membuat sketsa pada kertas
2. Menggunakan peranti purwarupa GUI
3. Menuliskan keterangan yang menjelaskan tentang kaitan antara jendela.
4. Menggunakan peranti bantu CASE (*Computer Aided Software Engineering*).

Cara kedua dan keempat tidak selalu dapat diterapkan, karena peranti tersebut biasanya harus dibeli dan seringkali cukup mahal. Cara ini kebanyakan diterapkan pada pembuatan antarmuka grafis untuk suatu jenis pekerjaan berskala besar.

II.1.1. Cara Pendekatan

Sebuah program aplikasi pastilah ditujukan kepada pengguna, yang utama, bukan perancangan program aplikasi tersebut. Program aplikasi pada dasarnya dapat dikelompokkan dalam dua kategori besar, yakni program aplikasi untuk keperluan khusus dengan pengguna yang khusus pula dan program aplikasi yang akan digunakan oleh pengguna umum, yang juga sering dikenal dengan sebutan public software. Karena perbedaan pada calon pengguna, maka perancang program antarmuka perlu memperhatikan hal ini (Insap Santoso ; 2009 : 186).

Pada kelompok pertama, yakni pada program aplikasi untuk keperluan khusus, misalnya program aplikasi untuk inventori gudang, pengelolaan data akademis mahasiswa, pelayanan reservasi hotel, dan program-program aplikasi yang serupa, kelompok calon pengguna yang akan memanfaatkan program aplikasi tersebut dapat dengan mudah diperkirakan, baik dalam hal keahlian pengguna maupun ragam antarmuka yang akan digunakan. Untuk kelompok ini ada satu pendekatan yang dapat dilakukan, yakni pendekatan yang disebut dengan pendekatan perancangan berpusat ke pengguna (*user centered design approach*). Cara pendekatan ini berbeda pendekatan perancangan oleh pengguna (*user design approach*).

Pendekatan perancangan berpusat ke pengguna adalah perancangan antarmuka yang melibatkan pengguna. Pelibatan pengguna di sini tidak diartikan bahwa pengguna harus ikut memikirkan bagaimana implementasinya nanti, tetapi pengguna diajak untuk aktif berpendapat ketika perancangan antarmuka sedang menggambar wajah antarmuka yang mereka inginkan. Dengan kata lain, perancangan dan pengguna duduk bersama-sama untuk merancang wajah antarmuka yang diinginkan pengguna. Pengguna menyampaikan keinginannya. Sementara perancangan menggambar keinginan pengguna tersebut sambil menjelaskan keuntungan dan kerugian wajah antarmuka yang diinginkan oleh pengguna, seolah-olah sudah mempunyai gambaran nyata tentang antarmuka yang nanti akan mereka gunakan (Insap Santoso ; 2009 : 187).

Pada perancangan oleh pengguna, pengguna sendirilah yang merancang wajah antarmuka yang diinginkan. Di satu sisi, cara ini akan mempercepat proses pengimplementasian modul antarmuka. Tetapi di sisi yang lain, hal ini justru sangat memberatkan pemrogram karena apa yang diinginkan pengguna belum tentu dapa diimplementasikan dengan mudah, atau bahkan tidak dapat dikerjakan dengan menggunakan peranti bantu yang ada.

Perancang program aplikasi yang dimasukkan dalam kelompok kedua, aau public *software*, perlua menganggap bahwa program aplikasi tersebut akan digunakan oleh pengguna dengan berbagai tingkat kepandaian dan karakteristik yang sangat beragam. Di satu sisi keadaan ini dapat ia gunakan untuk memaksa pengguna menggunakan antarmuka yang ia buat, tetapi pada sisi lain pemaksaan itu akan berakibat bahwa program aplikasinya menjadi tidak banyak

penggunanya. Satu kunci penting dalam pembuatan modul antarmuka untuk program-program aplikasi pada kelompok ini adalah dengan melakukan customization. Dengan customization pengguna dapat menggunakan program aplikasi dengan wajah antarmuka yang sesuai dengan selera masing-masing pengguna.

Salah satu contoh dari adanya kemampuan yang dimiliki oleh sebuah program aplikasi atau sistem operasi yang dapat disesuaikan dengan karakteristik pengguna adalah pengaturan desktop pada OS X versi 10.5 favoritnya, sehingga pengguna dapat mengubahnya sesuai keinginan justru akan membuat mata pengguna itu sakit, dikarenakan mata harus melakukan akomomasi maksimum terus menerus untuk menyesuaikan dengan warna tampilan yang ada.

Selain cara pendekatan yang dijelaskan di atas, Anda yang terbiasa menulis program-program aplikasi mungkin mempunyai cara khusus untuk berhadapan dengan pengguna. Tetapi perlu Anda ingat bahwa apapun cara yang Anda gunakan, Anda tetap harus mempunyai pedoman bahwa pada akhirnya program itu bukan untuk Anda sendiri, tetapi akan digunakan oleh orang lain. Dengan kata lain, jangan pernah mengabaikan pendapat (calon) pengguna program aplikasi Anda. (Insap Santoso ; 2009 : 188).

II.1.2. Prinsip Dan Petunjuk Perancangan

Antarmuka pengguna secara alamiah terbagi menjadi empat komponen model pengguna, bahasa perintah, umpan balik, dan penampilan informasi. Model pengguna merupakan dasar dari tiga komponen yang lain (Insap Santoso ; 2009 : 188).

Model mental pengguna merupakan model konseptual yang dimiliki oleh pengguna ketika ia menggunakan sebuah sistem atau program aplikasi. Model ini memungkinkan seorang pengguna untuk mengembangkan pemahaman mendasar tentang bagian yang yang dikerjakan oleh program, bahkan oleh pengguna yang sama sekali tidak mengetahui teknologi komputer. Dengan pertolongan model itu pengguna dapat mengantisipasi pengaruh suatu tindakan yang dilakukan dan dapat memilih strategi yang cocok untuk mengoperasikan program tersebut. Model pengguna dapat berupa suatu simulasi tentang keadaan yang sebenarnya dalam dunia nyata, sehingga ia tidak perlu mengembangkannya sendiri dari awal.

Setelah pengguna mengetahui dan memahami model yang di inginkan, dia memerlukan peranti untuk memanipulasi model itu. Peranti pemanipulasian model ini sering disebut dengan bahasa perintah (command language), yang sekaligus merupakan komponen kedua dari antarmuka pengguna. Idealnya program komputer kita mempunyai bahasa perintah yang alami, sehingga model pengguna dengan cepat dapat dioperasionalkan. (Insap Santoso ; 2009 : 189).

Komponen ketiga adalah umpan balik. Umpan balik di sini diartikan sebagai kemampuan sebuah program yang membantu pengguna untuk mengoperasikan program itu sendiri. Umpan balik dapat berbentuk pesan penjelasan, pesan penerimaan perintah, indikasi adanya obyek terpilih, dan penampilan karakter yang diketikkan lewat papan ketik. Beberapa bentuk umpan balik terutama ditujukan kepada pengguna pengguna yang belum berpengalaman dalam menjalankan program sebuah aplikasi. Umpan balik dapat digunakan untuk

member keyakinan bahwa program telah menerima perintah pengguna dan dapat memahami maksud perintah tersebut.

Komponen keempa adalah tampilan informasi. Komponen ini digunakan untuk menunjukkan status informasi atau program ketika pengguna melakukan suatu tindakan. Pada bagian ini perancang harus menampilkan pesan-pesan tersebut seefektif mungkin sehingga mudah dipahami oleh pengguna. Setelah memahami beberapa prinsip dalam perancangan antarmuka pengguna. Pada bagian berikut ini akan diberikan petunjuk singkat tentang perancangan anarmuka yang akan Anda lakukan sebagai seorang perancang tampilan.

II.1.3. Urutan Perancangan

Perancangan dialog, seperti halnya perancangan sistem yang lain, harus dikerjakan secara aas ke bawah. Proses perancangannya dapat dikerjakan secara bertahap sampai rancangan yang diinginkan terbentuk, yaitu sebagai berikut (Insap Santoso ; 2009 : 190).

1. Pemilihan ragam dialog

Untuk suatu tugas tertentu, pilihlah ragam dialog yang menurut perkiraan cocok untuk tugas tersebut. Ragam dialog dapat dipilih dari sejumlah ragam dialog yang telah dijelaskan pada bab-bab sebelumnya. Pemilihan ragam dialog dipengaruhi oleh karakterisiktik populasi pengguna, tipa dialog yang diperlukan, dan kendala teknologi yang ada untuk mengimplementasikan ragam dialog tersebut. Ragam dialog yang terpilih dapat berupa sebuah ragam tunggal, atau sekumpulan ragam dialog yang satu sama lain saling mendukung.

2. Perancangan Struktur Dialog

Tahap kedua adalah melakukan analisis tugas dan menentukan model pengguna dari tugas tersebut untuk membentuk struktur dialog yang sesuai. Dalam tahap ini pengguna sebaiknya banyak dilibatkan, sehingga pengguna langsung mendapatkan umpan balik dari diskusi yang terjadi. Pada tahap ini suatu purwarupa dialog seringkali dibuat untuk memberik gambarang yang lebih jelas kepada calon pengguna.

3. Perancangan format pesan

Pada tahap ini tata letak tampilan dan keterangan tekstual secara terinci harus mendapat perhatian lebih. Selain itu, kebutuhan data masukan yang mengharuskan pengguna untuk memasukkan data ke dalam komputer juga harus dipertimbangkan dari segi efisiensinya. Salah satu contohnya adalah dengan mengurangi pengetikan yang tidak perlu dengan cara mengefektifkan pengguna tombol.

II.2. *Encryption*

Untuk dapat membahas arti penting dari jumlah karakter pada sebuah *password,* harus pertama-tama mengenal apa yang namanya sistem enkripsi. Singkatnya enkripsi itu sendiri adalah sebuah teknik yang melakukan pengubahan informasi (berupa tulisan) menjadi sesuatu yang tidak bisa dimengerti orang lain, dengan menggunakan sebuah kunci matematis. Orang lain tidak akan pernah mengerti apa arti dari kata yang sudah diubah (diacak) oleh proses enkripsi

apabila mereka tidak memiliki kuncinya. Kunci daripada proses enkripsi bisa berupa aturan, angka, arahan, kata maupun kalimat. (ThOR ; 2010 : 27)

Analogi mudah, misalnya saya ingin melakukan enkripsi kalimat ”saya ingin makan” dan kuncinya adalah sebuah perhitungan matematis Matris 8x8, mungkin kalimat tersebut akan menjadi seperti 9xKKL^%ASNK 40lv. Beberapa contoh enkripsi yang paling banyak digunakan di dunia keamanan informasi adalah RSA, MD5 dan SHA1. (ThOR ; 2010 : 28)

Banyak orang salah mengartikan *Encoding* dan Enkripsi (*Encryption*) sebagai suatu hal yang sama. Kenyataannya kedua hal ini berbeda jauh. Walaupun memiliki fungsi yang sama, yakni melakukan pengacakan informasi dan membuat sebuah informasi menjadi tidak bisa dimengerti oleh orang yang tidak memiliki hak untuk memilikinya, namun ada sebuah perbedaan besar antara *Encoding* dan Enkripsi.

Di saat Enkripsi membutuhkan sebuah kunci untuk melakukan tugasnya, yang dibutuhkan encoding hanyalah sebuah ketentuan yang harus dia jalankan. Ketentuan tersebut bisa saja berupa kalimat suruhan atau arahan seperti ”majukan seluruh huruf 3 urutan ke dapan”. Sehingga sebuah kata ”saya” akan berubah menjadi ”Vdbd”.

Memang terkesan lucu, namun perusahaan besar seperti *Yahoo Inc*. Menggunakan teknik *decode* yang mudah untuk melindungi *cookie* para pengguna layanannya. Berikut adalah gambaran singkatnya seperti contoh *cookie* yang pernah didapatkan dari proses *hacking account yahoo*. Ini adalah contoh salah satu *cookies* yang didapatkan :

Y=v=1εn=8jnd0gsjo11f8εl=h4d_meh3b854_rv/oεp=m2occph01300000
0εjb=16\47\εiz=1016εr=csεlg=usεintl=usεnp=1;

Apabila ingin mencoba membacanya, silakan membaca dengan urutan sebagai berikut :

1) sk c&s *session*
2) n *is password*
3) l *is username*
4) p *is country, year of birth, gender and more*
5) b *is cookies created*
6) lg *is language*
7) intl *is internationl language*
8) iz *is zipcode*
9) jb *is industry and title*

Dan apabila melakukan *decoding* (proses pengembalian informasi yang sudah di *encode* ke text normal lagi) caranya seperti ini :

0123456789abcdefghijklmnopqrstuvwxyz

abcdifghijklmnopqrstuvwxyz0123456789

Maka *username*, atau *value* l (*login name*) pada *cookie* tersebut bisa dibaca dengan mudah menjadi :

H4d_meh3b854_rv/o

Ren_wordlife_15/o

Dimana *variable* /o akan selalu tetap pada setiap *username*. Tapi tentu saja *yahoo* sudah tidak menggunakan teknik seperti ini lagi. (ThOR ; 2010 : 29)

II.3. RSA

Dari sekian banyak algoritma kriptografi kunci publik yang pernah dibuat, algoritma yang paling populer adalah algoritma RSA. Algoritma ini melakukan pemfaktoran bilangan yang sangat besar. Oleh karena alasan tersebut RSA dianggap aman. Untuk membangkitkan dua kunci, dipilih dua bilangan prima acak yang bear. Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (Massachussets Institute of Technology) pada tahun 1976, yaitu : Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman. RSA mengekspresikan teks asli yang dienkripsi menjadi blok-blok yang mana setiap blok memiliki nilai bilangan biner yang diberi simbol n, blok teks asli M dan blok teks kode C. Untuk melakukan enkripsi pesan M, pesan dibagi ke dalam blok-blok numerik yang lebih kecil daripada n (data biner dengan pangkat terbesar). Jika bilangan prima yang panjangnya 200 digit, dapat ditambah beberapa bit 0 di kiri bilangan untuk menjaga agar pesan tetap kurang dari nilai n. (Dony Ariyus ; 2010 : 148)

RSA adalah sebuah algoritma pada *enkripsi public key* RSA merupakan algoritma pertama yang cocok untuk digital *signature* seperti halnya enkripsi, dan salah satu yang paling maju dalam bidang kriptografi public key. RSA masih digunakan secara luas dalam protokol perdagangan elektronik (*electronic commerce*), dan dipercaya dalam mengamankan dengan menggunakan kunci yang cukup panjang. (Wahana Komputer ; 2010 : 16)

Algoritma RSA dijabarkan pada tahun 1977 oleh tiga orang, yaitu Ron Rivest, Adi Shamir dan Len Adleman dari *Masachusetts Institute of Technology*.

Huruf RSA itu sendiri berasal dari inisial nama mereka (Rivest, Shamir, Adleman).

Clifford Cocks, seorang matematikawan Inggris yang bekerja untuk GCHQ, menjabarkan tentang sistem *equivalen* pada dokumen internal di tahun 1973. Penemuan *Clifford Cocks* tidak terungkap hingga tahun 1997 karena alasan *top secret classification*.

Algoritma tersebut dipatenkan oleh *Massachusetts Institute of Technology* pada tahun 1983 di Amerika Serikat sebagai U.S. Patent 4405829. Paten tersebut berlaku hingga 21 September 2000. Semenjak algoritma RSA dipublikasikan sebagai aplikasi paten, regulasi di sebagian besar negara-negara lain tidak memungkinkan penggunaan paten. Hal ini menyebabkan hasil temuan *Clifford Cocks* dikenal secara umum dan tidak dapat dipatenkan di Amerika Serikat.

RSA dapat juga digunakan untuk mengesahkan sebuah pesan. Misalkan Alice ingin mengirim pesan kepada Bob. Alice membuat sebuah hash value dari pesan tersebut, dipangkatkan dengan bilangan d dibagi N seperti halnya pada deskripsi pesan, dan melampirkannya sebagai tanda tangan pada pesan tersebut. Saat bob menerima pesan yang telah ditandatangani, Bob memangkatkan tanda tangan tersebut dengan bilangan e dibagi N (seperti halnya pada enkripsi pesan), dan membandingkannya dengan nilai hasil dan hash value dengan hash value pada pesan tersebut. Jika kedua cocok, maka bob dapat mengetahui bahwa pemilik dari pesan tersebut adalah Alice, dan pesan pun tidk pernah diubah sepanjang pengiriman.

Harap dicatat bahwa skema padding merupakan hal yang esensial untuk mengamankan pengesahan pesan seperti halnya pada enkripsi pesan, oleh karena itu kunci yang sama tidak digunakan pada proses enkripsi dan pengesahan. Penyerangan yang paling umum pada RSA ialah pada penanganan masalah faktorisasi pada bilangan yang sangat besar. Apabila terdapat faktorisasi metode yang baru dan cepat telah dikembangkan, ada kemungkinan untuk membongkar RSA. (Wahana Komputer ; 2010 : 17)

Pada tahun 2005, bilangan faktorisasi terbesar yang digunakan secara umum ialah sepanjang 663 *bit*, menggunakan metode distribusi mutahir. Kunci RSA pada umumnya sepanjang 1024-2048 *bit*. Beberapa pakar meyakini bahwa kunci 1024 bit ada kemungkinan dipecahkan pada waktu dekat (hal ini masih dalam perdebatan), tetapi tidak ada seorangpun yang berpendapat kunci 2048 *bit* akan pecah pada masa depan.

Metode RSA digagas oleh Ron Rivest, Adi Shamir dan Leonard Adleman dari MIT tahun 1977. Walaupun metode RSA sudah berumur 30 tahun, tetapi metode ini termasuk metode pengaman dokumen yang cukup handal. (Supriyono, jurnal.sttn-batan.ac.id ; 2008 : 181).

Adapun rumus mamtematika beserta prosedur metode RSA adalah sebagai berikut :

1. Ambil secara random dua bilangan prima p dan q yang besar dan berbeda, namun ukuran keduanya (jumlah *digit* dalam basis bilangan yang digunakan harus sama.

ϕ

2. Hitung modulus n dan fungsi Euler's Totient (n) dengan rumus :

   (1) n = p q

   a) (n) ϕ(p – 1) [q – 1]

   b) dengan :

      n = modulus (*public key*)

      p dan q = dua bilangan prima yang dimunculkan secara random.

3. Pilih suatu bilangan *integer* e sedemikian hingga

   I < e < ϕ(n) dan gc d(e, ϕ(n)) = 1

   (2) dengan:

      I = bilangan *integer*

      e = *public key* (kunci enkripsi)

      gcd = persekutuan pembagi terbesar (*greatest common divisor*)

4. Hitung nilai *integer* d dimana I < d < ϕ) sedemikian hingga

      d = e-1 mode ϕ(n) atau ed = I(mod ϕ(n))

      (3) dengan :

      d = *private key* (kunci dekripsi)

5. Membangun tabel untuk mempresentasikan tiap karakter.

6. Plainteks (teks yang akan dienkripsi) disandikan dengan angka-angka sesuai dengan tabel yang terbentuk oleh proses 5 dan akan diperoleh suatu nilai M yang merupakan kumpulan angka-angka dari *plaintext*, kemudian kumpulan angka-angka tersebut diblok tiap 4 angka menjadi m_1, m_2,..., m_n. Proses enkripsi dilakukan per blok dan masing-masing blok rumus enkripsinya adalah $c_1 = m_1^e$ (modn), $c_2 = m_2^e$ (modn), ... dst, sehingga menghasilkan nilai

C dimana C merupakan kumpulan angka-angka dari c_1, c_2, ... , c_n. (Supriyono, jurnal.sttn-batan.ac.id ; 2008 : 182).

7. Proses dekripsi dilakukan dengan menggunakan logika seperti langkah 6 dengan melakukan perhitungan terbalik, yaitu $m_1 = c_1^d$ (modn), $m_2 = c_2^d$ (modn) ,dst, sehingga menghasilkan nilai M dimana M = m_1 m_2 m_3 Nilai akhir M tersebut dipresentasikan balik dengan *table* yang dibangun seperti pada proses 5 di atas. (Supriyono ; 2008 : 183).

Implementasi RSA akan dilakukan enkripsi menggunakan algoritma RSA terhadap plaintext M = FISIKA. Pertama-tama plaintext tersebut diubah menjadi format ASCII sebagai berikut : (Tri Rahajoeningroem ; 2011: 80).

Tabel II.1. Format ASCII

Text (karakter)	F	I	S	I	K	A
ASCII (heksa)	46	49	53	49	4B	41
ASCII (desimal)	70	73	83	73	75	65

(sumber : Tri Rahajoeningroem ; 2011)

Plaintext dalam format ASCII desimal tersebut kemudian dipecah menjadi blok-blok tiga digit berikut :

$m_1 = 707$ $m_3 = 737$

$m_2 = 383$ $m_4 = 565$

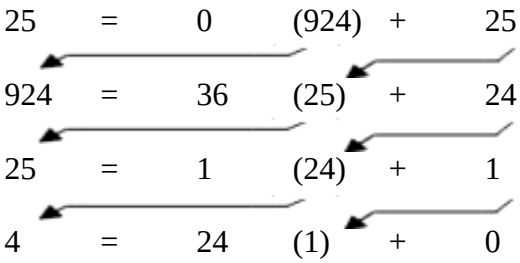
Dalam membuat kunci RSA, perlu dirancang agar nilai m_i masih terletak di dalam rentang antara 0 sampai $n - 1$. Maka ditentukan bahwa nilai n minimal adalah 909. Nilai ini diambil berdasarkan pertimbangan bahwa karakter huruf kapital dengan nilai terbesar adalah Z dengan nilai ASCII yaitu 5Ah atau 90. Kombinasi ZZ akan dapat dipecah menjadi blok 909 atau 090. Misalkan dipilih p

= 23 dan *q = 43* (keduanya prima), maka dapat dihitung : (Tri Rahajoeningroem, jurnal.unikom.ac.id ; 2011 : 81).

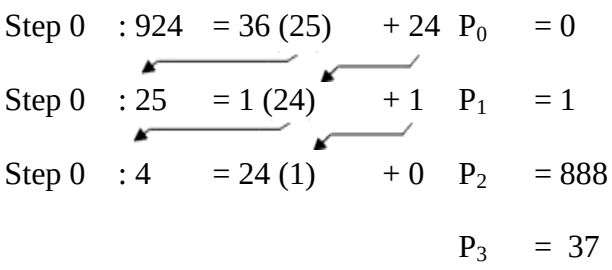
n = p x q = 989

m = (p – 1) x (q – 1) = 924

Dipilih kunci publik *e = 25* (yang relatif prima dengan 924 karena pembagi bersama terbesarnya adalah 1). Bahwa 25 relatif prima terhadap 924 dapat dibuktikan dengan mencari nilai gcd (25,924) melalui algoritma Euclid seperti berikut.



Hasil gcd pada algoritma ini adalah hasil sisa bagi terakhir sebelum 0. Maka pada perhitungan diatas terlihat bahwa sisa bagi sebelum nol adalah 1. Maka gcd (25, 924) = 1. Selanjutnya untuk menghitung kunci privat *d* algoritma *Extended Euclid* sebagai berikut :



P2 dan P3 dihitung melalui persamaan berikut

P2 = (pi-2 – pi-1q-i-2-) mod n

= (0 – 1(36)) mod 924 = 888

P3 = (1 – 888(1)) mod 924 = 37

Maka diperoleh kunci publik adalah 25 dan 989. Sedangkan kunci *privat* adalah 37 dan 989. Enkripsi setiap blok diperoleh menggunakan kunci *public* 25 dan 989 dengan cara sebagai berikut :

$c1 = 707^{25} \bmod 989$

Untuk menghitung $707^{25} \bmod 989$ dapat menggunakan teknik *divide and conquer* untuk membagi pemangkatnya sampai berukuran kecil. Ilustarinya adalah sebagai berikut :

$707^{25} = 707^{16} \cdot 707^{8} \cdot 707^{1}$

$707^{2} \bmod 989 = 499849 \bmod 989 = 404$

$707^{4} \bmod 989 = (707^{2} \cdot 707^{2}) \bmod 989$

$= [(707^{2} \bmod 989)\ (707^{2} \bmod 989)]$

Mod 989

$= (404 \cdot 404) \bmod 989$

$= 163216 \bmod 989 = 31$

$707^{8} \bmod 989 = (707^{4} \cdot 707^{4}) \bmod 989$

$= [(707^{4} \bmod 989)(707^{4} \bmod 989)]$

Mod 989

$= (31 \cdot 31) \bmod 989$

$= 961 \bmod 989 = 961$

$707^{16} \bmod 989 = (707^{8} \cdot 707^{8}) \bmod 989$

$= [(707^{8} \bmod 989)(707^{8} \bmod 989)]$

Mod 989

= (961 . 961) mod 989

= 923521 mod 989 = 784

707^{25} mod 989 = (707^{16} . 707^{8} . 707^{1}) mod 989

= [(707^{16} mod 989 . 707^{8} mod 989) mod 989 . 707^{1} mod 989) mod 989

= ((784 . 961) Mod 989 . 707) mod 989

= ((753424) mod 989 . 707) mod 989

= (795 . 707) mod 989 = 313

Jadi c1 = 707^{25} mod 989 = 313. Dengan cara yang sama dapat diperoleh :

$c_2 = 383^{25}$ mod 989 = 776

$c_3 = 737^{25}$ mod 989 = 737

$c_4 = 565^{25}$ mod 989 = 909

Maka *ciphertext* adalah C = 313 776 737 909. Perlu diingat, bahwa *ciphertext* ini dalam format ASCII desimal. Jika diubah kembali menjadi format karakter maka dapat diperoleh:

Tabel II.2. Format Pengembalian Karakter

ASCII (desimal)	31	37	76	73	79	09
ASCII (heksa)	1F	24	4C	49	4F	09
Text (karakter)		%	L	I	O	

(sumber : Tri Rahajoeningroem ; 2011)

Heksadesimal 1F dan 09 adalah *non-printing characters*.

Untuk melakukan dekripsi (mengubah *ciphertext* menjadi *plaintext*) maka digunakan kunci privat 37 dan 989 dengan cara seba-gai berikut :

m_1 = 31337 mod 989 = 707

m_2 = 77637 mod 989 = 383

m_3 = 73737 mod 989 = 737

m_4 = 90937 mod 989 = 565

Maka diperoleh *plaintext* 707 383 737 565. (Tri Rahajoeningroem, jurnal.unikom.ac.id ; 2011 : 82).

II.4. MD5

MD5 atau yang juga dikenal dengan nama Message Digest Algorithm 5, merupakan sebuah teknik enkripsi yang terbilang cukup muda. Ditemukan pada April 1992, oleh seorang ahli matematikan bernama Ron Rivest. Teknik ini memiliki saudara-saudara yang sebelumnya juga sudah pernah tampil, yakni MD, MD2, MD3, dan MD4. Namun keempat saudara MD5 itu kandas dimakan waktu. Kelemahan demi kelemahan yang ditemukan membuat mereka tidak lagi populer untuk dipergunakan (Whindy Yoevestian;2008:29).

MD5 memang dikenal sebagai enkripsi yang paling mudah dipergunakan dan terbilang sangat aman. Tak jarang para pecinta kriptografi menyatakan bahwa MD5 adalah teknik enkripsi satu arah, yang juga berarti apabila Anda sudah mengacak sebuah informasi tertulis yang sebenarnya, Bayangkan sebuah pintu kenop yang hanya ada pada bagian depan pintu.

Dengan menggunakan 128 bit enkripsi, MD5 selalu menghasilkan 32 buah karakter yang terdiri dari huruf dan angka sebagai hasil akhirnya. Walupun Anda memasukkan informasi yang sangat panjang atau sanga pendek, hasil yang akan keluar tetap berupa kombinasi angka dan huruf sebanyak 32 buah karakter.

Hasil yang akan didapat melalui enkripsi MD5 terhadap kalimat “Nama saya adalah Th0R dan saya seorang pencinta ilmu komputer dan teknologi” adalah 88817b3e8ac59fe429333ef1dd3b459e (32 buah karakter kombinasi angka dan huruf).

Sedangkan hasil yang didapatkan dari proses enkripsi angka 1 adalah c4ca4238a0b823820dcc509a6f75849b (tetap jumlah 32 karakter). Walaupun Anda hanya mengubah sedikit saja bagian daripada kalimat yang ingin Anda enkripsi, misalnya kalimat di atas menjadi “Bana saya Th0R dan saya seorang pencinta ilmu komputer dan teknologi” (Anda perhatikan huruf m dalam kata yang seharusnya “dan”), hasilnya adalah 8f66121c3197e935cd35cb34bf3310d2. Tetap berupa kombinasi angka dan huruf dengan jumlah karakter 32 buah, namun tidak memiliki kesamaan, sistem enkripsi MD5 tidak memiliki pattern (atau pola) dalam melakukan pekerjaan.

II.5. Perancangan *File* MD5

Algoritma *enkripsi* dengan MD5 ini adalah misalnya ingin mengenkripsi teks menjadi bentuk *enkripsi* MD5. Teks tersebut akan dienkripsi dengan teknik *hash* dari MD5 sehingga akan menghasilkan *Chiper* teks. Inti dari enkripsi adalah algoritmanya, langkah-langkah yang dilakukan *Message Digest* 5 (MD5) adalah sebagai berikut :

1. Inisialisasi

   Message Digest 5 (MD5) yang utama beroperasi pada kondisi 128-bit, dibagi menjadi empat *word* 32 bit, yaitu A,B,C, dan D. Operasi tersebut diinisialisasi

dan dijaga untuk tetap konstan. *Register* A,B,C dan D diinisialisasi dengan bilangan *hexadecimal*.

2. Operasi mengupdate string teks masukan.

   Yaitu melakukan update blok MD5, melanjutkan operasi MD5, memproses blok pesan serta melakukan update pada konteks.

   a. Menghitung jumlah *byte* dengan sisa hasil baginya adalah 64. Siapkan sebuah variable index untuk menentukan panjang bytenya. Dimana nilai index tersebut adalah alamat pertama dari count tersebut dilakukan operasi pergeseran bit kearah kanan sebanyak 3 kali serta dilakukan operasi AND dengan nilai Hexa 0x3f.

      Melakukan *update* jumlah *bit*, dimana sudah diketahui teksnya dan panjang teksnya.

   b. Mengubah sesuai dengan jumlah kemungkinan yang ada.

   c. Pengisian *buffer*

3. Pembuatan Hash atau proses Finalisasi yaitu tahap akhir proses MD5, dimana terjadi proses merubah pesan data konteks ke dalam *hash buffer* dan mengubah dari hash buffer menjadi string, Siapkan sebuah variable buffer bertipe array dengan nomor indexnya adalah 16.

   a. Menyimpan dan mengenkode ke dalam bentuk *bit-bit*, sepanjang 8 bit dengan asumsi panjang adalah kelipatan 4.

   b. Padding kedalam 56 dengan sisa hasil bagi 64

   c. Penambahan panjang bit sebanyak 8, dengan proses seperti yang dilakukan pada langkah no.2 diatas

d. Menyimpan ke dalam digest (inti). Dengan melakukan proses *encoding* dari konteks *state* ke dalam digest dengan panjang 16.
e. Pengosongan *buffer* dengan len merupakan panjang konteks.
f. Mengubah dari hash menjadi string. Mengkonversi hash bernilai numerik kedalam bentuk string dengan menggunakan fungsi KonversiString.

II.6. Pemodelan

Pemodelan UML perangkat lunak bekerja dengan cara yang cukup serupa layaknya seorang arsitek atau insinyur teknik sipil yang akan membuat sebuah bangunan / gedung berskala besar. Saat seorang arsitek atau insinyur teknik sipil akan membuat sebuah bangunan / gedung berskala besar, ia biasanya membuat denah-denah atau maket-maket yang menggambarkan bentuk jadi dari bangunan / gedung. Sebagai seorang perancang sistem perangkat lunak juga bertindak dengan cara yang serupa, hanya saja yang kita rancang bukan bangunan, melainkan sistem perangkat lunak. Menggambarkan komponen-komponen sistem perangakt lunak dalam bentuk-bentuk geometri tertentu misalmya untuk menggambarkan suatu kelas (class) dalam aplikasi, menggunakan antarkelas (asosiasi), menggunakan garis lurus (Adi Nugroho ; 2009 : 6).

II.6.1. Diagram *Use Case*

Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan use case diagram yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, use case diagram tidak hanya sangat penting pada tahap analisis,

tetapi juga sangat penting untuk perancangan (design), untuk mencari (mencoba menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian (testing) (Adi Nugroho; 2009:7).

Dengan menggunakan use case diagram, kita akan mendapatkan banyak informasi yang sangat penting yang berkaitan dengan aturan-aturan bisnis yang coba kita tangkap. Dalam hal ini, setiap objek yang berinteraksi dengan sistem perangkat lunak misalnya, orang, suatu perangkat keras, sistem lain, dan sebagainya merupakan actor untuk sistem perangkat lunak, sementara use case merupakan deskripsi lengkap tentang bagaimana sistem perangkat lunak berperilaku untuk para actornya. Dengan demikian, use cara diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para actor dengan sistem perangkat lunak yang sedang kita kembangkan.

Actor pada dasarnya ditentukan berdasarkan perannya (role) pada program /aplikasi yang sedang kita kembangkan, bukan sebagai objek-objek secara mandiri. Sebagai contoh, jika mengambil kasus ATM (Anjungan Tunai Mandiri), seseorang (objek tunggal) mungkin bisa dikelompokkan sebagai actor Karyawan Bank serta Nasabah jika oarng tersebut merupakan karyawan bank yang bersangkutan sekaligus sebagai nasabah karena memiliki tabungan di bank tersebut. Sementara itu, Adi, Ana Geuis, dan beberapa orang lainnya dapat dikelompokkan menjadi actor nasabah jika mereka semua masing-masing memiliki tabungan di bank tersebut.

II.6.2. Diagram *Activity*

Activity diagram menggambarkan urutan aktifitas yang digunakan untuk menjelaskan aktifitas dari sebuah operasi. Pada activity diagram terdapat keadaan aksi yang berisi spesifikasi dari aktifitas tertentu. Diagram ini berisi, pilihan keputusan dan kondisi serta spesifikasi message yang dikirim atau diterima sebagai gambaran dari aksi.

II.6.3. *Sequence*

Diagram interaksi yang menekankan pada waktu pengiriman *message*. *Sequence* diagram menunjukkan sekumpulan objek dan pengiriman serta penerimaan *message* antar objek. Objek yang umumnya memiliki nama atau instansiasi dari *class*, tapi dapat pula merupakan turunan dari *things* lain, seperti *collaboration*, *component* dan *node*. Diagram ini digunakan untuk mengilustrasikan *dynamic view* dari sistem.

II.6.4. *State*

Sebuah *state* diagram menggambarkan keadaan mesin, transisi, *event* dan *activity*. Diagram ini adalah pelengkap khusus untuk mendeskripsikan sebuah *class* yang menggambarkan *state* dari objek dari *class* dan *event* yang menyebabkan *state* berubah. *Event* tersebt dapat berasal dari objek yang mengirimkan suatu *message* atau dari kondisi yang terpenuhi.

II.7. Bahasa Pemrograman Visual Basic 2010

Visual Basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi

Windows. Visual Basic 2010 atau Visual Basic 9 adalah versi terbaru yang telah diluncurkan oleh Microsoft bersama C#, visual C++, dan Visual Web Developer dalam satu paket Visual Studio 2010 (Wahana Komputer; 2010: 2).

Visual Basic 2010 merupakan aplikasi pemrograman yang menggunakan teknologi *.NET Framework.* Teknologi *.NET Framework* merupakan komponen Windows yang terintegrasi serta mendukung pembuata, penggunaan aplikasi, dan halaman web. Teknologi *.NET Framework* mempunyai 2 komponen utama, yaitu CLR (*Common Language Runtime*) dan *Class Library*, CLR digunakan untuk menjalan aplikasi yang berbasis .NET, sedangkan *Library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi.

Sebelum menginstall komputer harus memenuhi beberapa persyaratan agar Visual Basic 2010 dapat dijalankan dengan baik. Adapun, persyaratan *(System Requirements*) yang harus dipenuhi dapat Anda lihat pada Tabel II.3.

Tabel II.3. Sistem Requirements Visual Basic 2010

Sistem	Syarat Minimal	Syarat yang direkomendasikan
Arsitektur	X86 dan x64 (WOW)	
Sistem Operasi	Microsoft Windows XP Service Pack 2 Microsoft Windows Server 2003 Windows Vista	
Prosesor	CPU 1.6 GHz (Giga Hertz)	Windows XP dan Windows Server 2003:CPU 2,2 GHz atau yang lebih tinggi. Windows Vista : CPU 2,4 GHz
RAM	Windows XP dan Windows Server 2003 384 MB (Mega byte) Windows Vista : 768 MB	RAM 1024 MB / 1 GB atau yang lebih besar.
Harddisk	Tanpa MSDN Ruang Kosong harddisk pada drive	Kecepatan harddisk 7200 RPM atau yang

	penginstalan 2 GB. Sisa ruang harddisk kosong 1 GB Dengan MSDN Ruang kosong harddisk pada drive penginstalan 3,8 GB (MSDN diinstal full) 2,8 GB untuk menginstal MSDN default. Kecepatan Harddisk 5400 RPM.	lebih tinggi.
Display Layar	1024 x 768 display	1280 x 1024 display

(Sumber : Wahana Komputer ; 2010 : 2)

II.8. Keamanan Data

Keamanan data merupakan bagian dari perkembangan teknologi informasi. Ketika berpikir bahwa data yang dimiliki merupakan data yang sangat penting, semua berusaha untuk melindunginya agar jangan sampai jatuh ke tangan orang yang tidak bertanggung jawab. Tetapi buat sebagian orang, mereka justru tidak mengetahui sepenting apakah data yang mereka miliki. Karena ketidaktahuan tersebut, mereka baru menyadari bahwa data yang mereka miliki sangat penting setelah mengalami kecurian data dan mengalami kerugian. Data di sini bisa bersifat umum tidak terbatas pada data digital saja, tetapi juga seperti data diri (ktp, ijasah, sertifikat, dan lain-lain). Data yang menyangkut informasi pribadi tidak seharusnya diumbar sembarang seperti pada blog, situs jejaring pertemanan, email, selebaran, fotokopi KTP di buang sembarangan dan lain-lain. (Andik Susilo ; 2010 : 59).

Masalah keamanan merupakan salah satu aspek terpenting dari sebuah sistem informasi. Masalah keamanan sering kurang mendapat perhatian dari para perancang dan pengelola sistem informasi. Masalah keamanan sering berada di urutan setelah tampilan, atau bahkan di urutan terakhir dalam daftar hal-hal yang

dianggap penting. Apabile mengganggu performansi sistem, masalah keamanan sering tidak dipedulikan, bahkan ditiadakan. (Doni Ariyus; 2010:6).

Informasi menentukan hampir setiap elemen dari kehidupan manusia. Informasi sangat penting artinya bagi kehidupan karena tanpa informasi maka hampir semuanya tidak dapat dilakukan dengan baik. Contohnya, jika membeli tiket penerbangan dan membayarnya dengan menggunakan kartu kredit, informasi mengenai diri nantinya disimpan dan dikumpulkan serta digunakan oleh bank dan penerbangan. Demikian juga halnya saat membeli obat di apotik. Harus mendapat resep dari dokter dan memberikan resep tersebut ke pelayan apotik. Resep itu merupakan satu informasi yang disampaikan dokter ke pihak apotik tentang obat yang dibutuhkan.

Kemajuan sistem informasi memberikan banyak keuntungan bagi kehidupan manusia. Meski begitu, aspek negatifnya juga banyak, seperti kejahatan komputer yang mencakup pencurian, penipuan, pemerasan, kompetisi, dan banyak lainnya. Jatuhnya informasi ke pihak lain, misalnya lawan bisnis, dapat menimbulkan kerugian bagi pemilik informasi. Sebagai contoh, banyak informasi milik perusahaan yang hanya boleh diketahui oleh orang-orang tertentu di perusahaan tersebut, seperti misalnya informasi tentang produk yang sedang dalam pengembangan. Algoritma dan teknik yang digunakan untuk menghasilkan produk tersebut. Untuk itu keamanan dari sistem informasi yang digunakan harus terjamin dalam batas tertentu.

II.8.1. Ancaman Keamanan

Terjadi banyak pertukaran informasi setiap detiknya di *internet*. Juga banyak terjadi pencurian atas informasi oleh pihak-pihak yang tidak bertanggung jawab. Ancaman keamanan yang terjadi terhadap informasi adalah :

1. *Interruption*

   Merupakan ancaman terhadap *availability* informasi, data yang ada dalam sistem komputer dirusak atau dihapus sehingga jika data atau informasi tersebut dibutuhkan maka pemiliknya akan mengalami kesulitan untuk mengaksesnya, bahkan mungkin informasi itu hilang.

2. *Interception*

   Merupakan ancaman terhadap kerahasiaan (*secrecy*). Informasi disadap sehingga orang yang tidak berhak dapat mengakses komputer di mana informasi tersebut disimpan.

3. *Modification*

   Merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil menyadap lalu lintas informasi yang sedang dikirim dan kemudian mengubahnya sesuai keinginan orang tersebut.

4. *Fabrication*

   Merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil meniru atau memalsukan informasi sehingga orang yang menerima informasi tersebut menyangka bahwa informasi tersebut berasal dari orang yang dikehendaki oleh si penerima informasi. (Doni Ariyus; 2010:6).

II.8.2. Aspek-aspek Keamanan Komputer

Keamanan komputer meliputi empat aspek, antara lain :

1. *Authentication*

   Agar penerima informasi dapat memastikan keaslian pesan, bahwa pesan itu datang dari orang yang dimintai informasi. Dengan kata lain, informasi itu benar-benar datang dari orang yang dikehendaki.

2. *Integrity*

   Keaslian pesan yang dikirim melalui jaringan dan dapat dipastikan bahwa informasi yang dikirm tidak dimodifikasi oleh orang yang tidak berhak.

3. *Non Repudiation*

   Merupakan hal yang berhubungan dengan si pengirim. Pengirim tidak dapat mengelak bahwa dialah yang mengirim informasi tersebut.

4. *Authority*

   Informasi yang berada pada sistem jaringan tidak dapat dimodifikasi oleh pihak yang tidak berhak untuk mengaksesnya.

5. *Confidentiality*

   Merupakan menjaga informasi dari orang yang tidak berhak mengakses.

6. *Privacy*

   Lebih ke arah data-data yang bersifat pribadi.

7. *Availability*

   Aspek availabilitas berhubungan dengan ketersediaan informasi ketika dibutuhkan. Sistem informasi yang diserang atau dijebol dapat menghambat atau meniadakan akses ke informasi

8. *Access Control*

   Aspek ini berhubungan dengan cara pengaturan akses ke informasi. Hal ini biasanya berhubungan dengan masalah otentikasi dan privasi. Kontrol akses seringkali dilakukan dengan menggunakan kombinasi *user id* dan *password* dengan mekanisme lain. (Doni Ariyus ; 2010 : 6).

II.9. Data Teks

Setiap pengguna komputer biasanya punya data pribadi yang disimpan di komputer atau di media penyimpanan mobile macam *flashdisk*. Data itu bisa terkait dengan kegiatan pribadi misalnya, imil, riwayat jelajah di internet, atau terkait dengan urusan kerja. Setiap orang tentulah punyak kriteria sendiri tentang data vital pribadinya.

Bagaimana cara menyimpan dan melindungi data pribadi itu pun mungkin berbeda-beda antara satu pengguna dengan lainnya. Namun, pada umumnya para pengguna menyimpan data pribadinya itu dengan cara menempatkannya dalam *folder* tersendiri di partisi tertentu pada harddisk atau di *flashdisk*. Bahkan tak sedikit pengguna yang melatakkan data pentingnya di *My Documents*.

Tentuk saja cara penempatan *file* penting macam itu bisa aman-aman saja sejauh komputer atau flashdisk yang ditempati data itu digunakan sendiri, dan tak mungkin hilang. Dalam keadaan tertentu terpaksa meminjamkan komputer atau *flashdisk* kepada orang lain. Pada kondisi macam ini, tentulah data penting berpotensi atau dilirik orang yang dipinjami. (Bambang P Putranto ; 2010 : 1).