BAB III

ANALISIS MASALAH DAN RANCANGAN PROGRAM

III.1. Analisis Sistem yang Sedang Berjalan

Proses analisa sistem merupakan langkah kedua pada pengembangan sistem. Analisa sistem dilakukan untuk memahami informasi-informasi yang didapat dan dikeluarkan oleh sistem itu sendiri. Sistem perbandingan belum begitu banyak diketahui oleh seseorang, seseorang yang ingin melakukan pengamanan data yang ada pada aplikasi atau program yang digunakan lebih aman dari gangguan yang ingin merusak data tersebut. Berkembangnya teknologi informasi secara otomatis akan menambah jumlah data pribadi. Hal ini secara otomatis dapat lupa terhadap keamanan data tersebut.

Untuk itu, sistem yang penulis rancang adalah sistem yang melakukan perbandingan terhadap data dengan menggunakan dua metode yaitu RSA dan MD5 untuk mengetahui tentang keamanan yang dilakukan oleh kedua algoritma tersebut mana yang lebih aman, cepat dan akurat. Dalam tahap pengembangan sistem perbandingan ini, analisa sistem merupakan hal yang harus dilakukan sebelum proses perancangan sistem.

III.2. Strategi Pemecahan Masalah

Adapun strategi pemecahan masalah dari sistem perbandingan yang dirancang adalah sebagai berikut :

a. Data yang dibuat sangat penting dan rahasia, apabila dicuri orang data tersebut bisa di salah gunakan, oleh karena itu perlu untuk mengetahui langkah-langkah dan proses dalam melakukan *hash* data tersebut mana yang lebih aman dari kedua metode tersebut yaitu RSA dan MD5.
b. Suatu data dapat memiliki nilai kerahasiaan, karena data tersebut merupakan sumber daya yang strategis, maka pada kasus ini data tersebut harus diamankan dengan memberikan hak keamanan dengan mengubah data asli ke data yang tidak dimengerti seseorang dengan algoritma MD5 dan RSA.

III.3. Evaluasi Sistem Yang Berjalan

Sistem pemberian pengamanan data saat ini seperti pemberian password pada aplikasi *office* sudah aman tetapi masih dapat di bobol *password* tersebut. Kelemahan dari sistem ini, apabila *file* tersebut sudah diketahui maka isi *file* aslinya dapat dibaca oleh orang yang tidak bertanggung jawab tersebut.

Maka solusi yang penulis buat untuk mengatasi masalah tersebut adalah membuat suatu sistem perbandingan dengan menggunakan algoritma artinya mengubah data asli ke data yang tidak dapat dibaca.

III.4. Desain Sistem

Setelah tahapan analisis sistem, maka selanjutnya dibuat suatu rancangan sistem. Perancangan sistem adalah tahapan yang berguna untuk memperbaiki efisiensi kerja suatu sistem yang telah ada. Pada perancangan sistem ini terdiri dari tahap perancangan yaitu :

1. Perancangan *Use Case Diagram*
2. Perancangan *Sequence Diagram*
3. Perancangan *Activity Diagram*

III.5. Perbandingan RSA dan MD5

Message Digest 5 (MD5) yang utama beroperasi pada kondisi 128-bit, dibagi menjadi empat *word* 32 bit, yaitu A,B,C, dan D. *Register* A,B,C dan D diinisialisasi dengan bilangan *hexadecimal*, sedangkan RSA adalah Algoritmanya mengambil pesan yang panjangnya kurang dari 264 bit dan menghasilkan *message digest* 160-bit.

III.5.1. RSA

RSA sendiri memiliki proses operasi bit shift sebesar 1. Pada tahap preprocessing ini terdiri dari beberapa langkah, yaitu :

1. Penambahan *bit-bit* pengganjal (*padding bits*). Pesan ditambah dengan sejumlah *bit* pengganjal sedemikian sehingga panjang pesan dalam satuan bit *kongruen* dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 *bit* kurang dari kelipatan 512. Panjang *bit-bit* pengganjal haruslah berada antara 1 hingga 512 *bit*. Hal tersebut menyebabkan pesan dengan panjang 448 tetap harus ditambahkan *bit* penyangga sehingga panjangnya akan menjadi 960 *bit*. *Bit-bit* pengganjal sendiri terdiri dari sebuah *bit* 1 diikuti sisanya dengan *bit* 0.
2. Penambahan nilai panjang pesan semula. Pesan yang telah diberi *padding bits* selanjutnya ditambah lagi dengan 64 *bit* yang menyatakan panjang pesan

semula. Setelah ditambah dengan 64 *bit*, panjang pesan akan menjadi kelipatan 512 *bit*.

3. Inisialisasi penyangga (*buffer*) MD. Algoritma RSA dalam operasinya membutuhkan lima buah *buffer* yang masingmasing besarnya 32 *bit* sehingga nantinya hasil akhirnya akan menjadi 160 *bit*. Kelima *buffer* tersebut dalam operasi RSA ini akan berperan untuk menyimpan hasil antar putaran sekaligus untuk menyimpan hasil akhir. Kelima *buffer* tersebut memiliki nama A, B, C, D ,dan E. *Buffer-buffer* tersebut harus diinisialisasi dengan nilai-nilai sebagai berikut dalam notasi heksadesimal

   A = 67452301

   B = EFCDAB89

   C = 98BADCFE

   D = 10325476

   E = C3D2E1F0

Proses *hashing* dilakukan per blok dengan besar 512 *bit* tiap bloknya. Dalam pengolahan ini terdapat 4 putaran yang tiap putarannya dilakukan sebanyak 20 kali. Tiap putaran memiliki proses yang berbeda – beda. Sebelum putaran pertama dilakukan inisialisasi 5 buah *variable* dengan besar 32 *bit* yang menampung *buffer* inisialisasi.

Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 *bit*. Setiap blok 512 *bit* diproses bersama dengan *buffer* MD menjadi keluaran 128 *bit.* Operasi dasar RSA dapat ditulis dengan persamaan sebagai berikut:

$a, b, c, d, e = (CLS_5(a) + f_t(b, c, d) + e + W_t + K_t), a, CLS_{30}(b), c, d$

keterangan dari rumus diatas sebagai berikut :

a, b, c, d, e = lima buah peubah penyangga 32 *bit* (berisi A, B, C, D, E)

t = putaran, $0 \leq t \leq 79$

f_t = fungsi logika

CLS_s = *circular left shift* sebanyak s *bit*

W_t = *word* 32 *bit* yang diturunkan dari blok 512 *bit*

K_t = konstanta penambah

\+ = operasi penjumlahan modulo 232

Simbol <<< menyatakan operasi pergeseran *circular left shift*. Konstanta penambah dalam operasi RSA memiliki nilai berikut untuk masing-masing putaran, yaitu :

Putaran $0 \leq t \leq 19$ Kt = 5A827999

Putaran $20 \leq t \leq 39$ Kt = 6ED9EBA1

Putaran $40 \leq t \leq 59$ Kt = 8F1BBCDC

Putaran $60 \leq t \leq 79$ Kt = CA62C1D6

III.5.2. MD5

Message Digest 5 (MD5) yang utama beroperasi pada kondisi 128 *bit*, dibagi menjadi empat *word* 32 *bit*, yaitu A, B, C, dan D. *Register* A, B, C dan D diinisialisasi dengan bilangan *hexadecimal*. *Preprocessing* dimulai dengan penambahan *padding bits* sebagai berikut :

1. Pesan tersebut akan diberikan tambahan *bit* sehingga panjangnya dalam *bit* sama dengan 448 modulo 512. Penambahan selalu dilakukan walaupun

panjang dari pesan sudah sama dengan 448 modulo 512. Penambahan *bit* yaitu sebuah *bit* 1 ditambahkan diikuti dengan *bit* 0 sebanyak yang diperlukan sehinggan panjangnya menjadi 448 modulo 512. Minimal dilakukan penambahan sebanyak satu *bit* dan maksimal sebanyak 512 *bit*.

2. 64 *bit* ditambahkan untuk merepresentasikan panjang sebenarnya dari pesan b pada hasil dari penambahan sebelumnya. Jika panjang pesan asli lebih dari 264 *bit* maka hanya 64 *lower order bit* yang dimasukkan. *Lower order word* untuk panjang pesan asli dimasukkan sebelum *high-order word*. Setelah langkah kedua ini, maka panjang dari hasilnya adalah kelipatan dari 512 *bit* atau jika dalam *word* maka panjangnya akan sama dengan kelipatan dari 16 *word* (32 *bit*). Maka hasil dari dua langkah di atas dinotasikan sebagai M[0 …. N-1] dengan N adalah kelipatan dari 16.

Setelah *padding* pesan terdiri dari n word M[0 … n - 1] dimana n adalah kelipatan 16. Langkah berikutnya dalam *preprocessing* adalah menyiapkan MD buffer sebesar 4 word yaitu A, B, C, D. Dimana A merupakan *lower order word*. *Buffer* diberi nilai awal sebagai berikut nilai dalam *hexadecimal* dimulai dengan *lower order byte*.

A: 01 23 45 67

B : 89 ab cd ef

C : fe dc ba 98

D : 76 54 32 10

Proses *hashing* dilakukan per blok, dengan setiap blok melalui 4 putaran. Proses *hashing* menggunakan 4 fungsi F, G, H, dan I yang masing-masing mempunyai *input* 3 *word* dan *output* 1 *word* :

F(X,Y,Z) = (X ^ Y) v (~X ^ Z)

G(X,Y,Z) = (X ^ Z) v (Y ^ ~Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X ^ ~Z)

Adapun keterangan dari rumus diatas adalah

^ = *bitwise* and

v = *bitwise* or

xor = *bitwise exclusive* or, dan

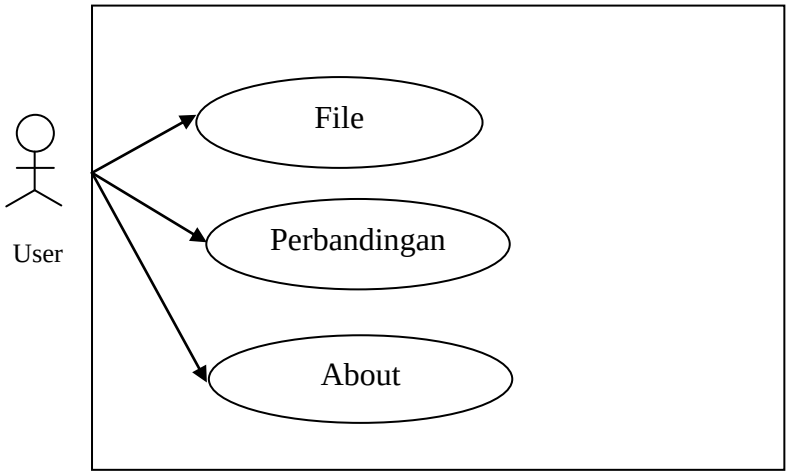
~ = *bitwise* not.

Pada setiap posis *bit*, F berperilaku sebagai if X then Y else Z. Jika setiap *bit* dari X, Y dan Z independent dan tidak bisa maka setiap *bit* dari F akan independen dan tidak bisa juga. Fungsi G, H dan I juga serupa dengan fungsi F, beroperasi pada *bitwise parallel* untuk menghasilkan *output* dari *input* X, Y dan Z. Serta dalam hal jika X, Y dan Z independent dan tidak bisa, maka G, H dan I juga independent dan tidak bisa.

Tahap ini menggunakan tabel T[1 …. 64] yang dibentuk dari fungsi sinus. Misal T[i] adalah elemen ke-i dari tabel, maka T[i] didefinisikan sebagai 4294967296 * abs(sin(i)), dimana I dalam radian.

III.6. *Use Case Diagram*

Use case menjelaskan urutan kegiatan yang dilakukan aktor dan sistem untuk mencapai suatu tujuan tertentu. Sebuah *Use Case* mempresentasikan sebuah interaksi antara aktor dengan sistem dan menggambarkan fungsionalitas yang diharapkan dari sebuah sistem perbandingan. Diagram *Use Case* tersebut dapat dilihat pada gambar III.1.



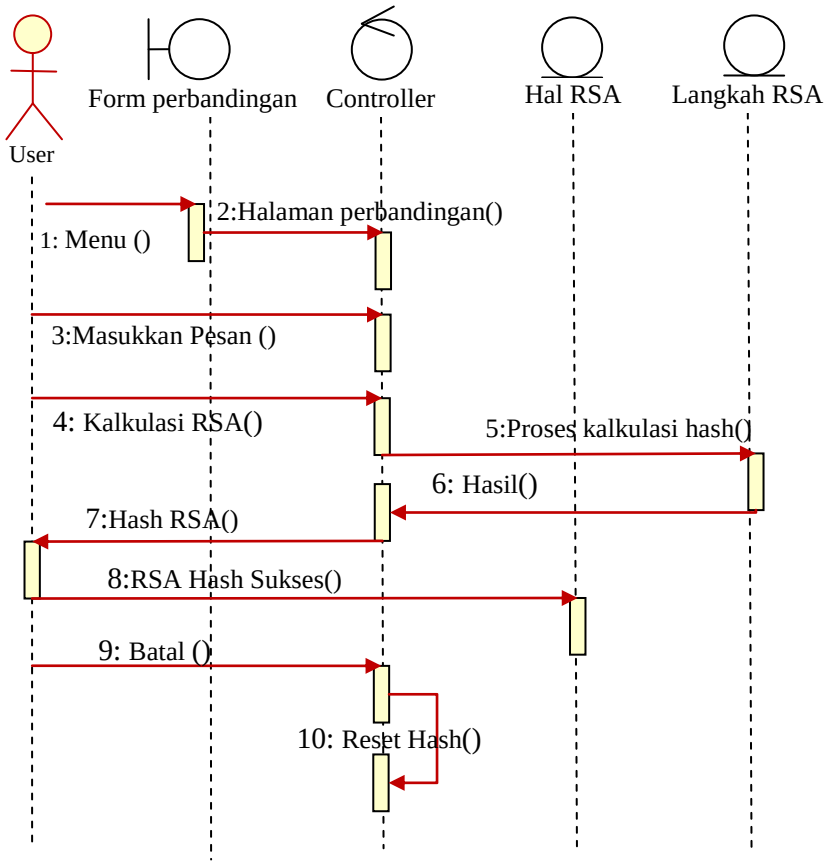
Gambar III.1. *Use Case Diagram* Menu Utama

III.7. *Sequence* Diagram

Sequence diagram menunjukkan bagaimana operasi yang dilakukan secara detail. *Sequence* diagram menjelaskan interaksi obyek yang disusun dalam suatu urutan waktu. Urutan waktu yang dimaksud adalah urutan kejadian yang dilakukan oleh seorang *actor* dalam menjalankan sistem, adapun *sequence* yang dilakukan terdiri dari perbandingan RSA dan MD5.

1. *Sequence Hash* RSA

Kalkulasi hash RSA digunakan untuk mengetahui langkah-langkah yang terjadi dalam pemrosesan setiap bloknya sampai perubahan data asli ke data yang tidak dimengerti dengan metode RSA, untuk lebih jelasnya dapat dilihat pada gambar III.2.

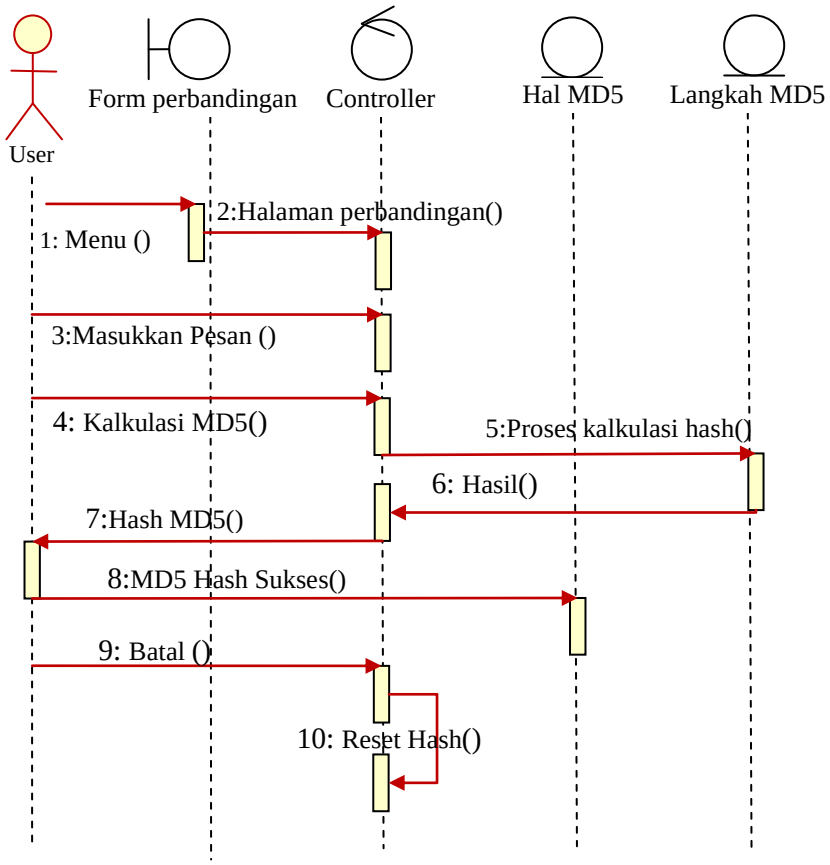


Gambar III.2. *Sequence RSA*

Dari gambar III.2 menunjukkan bahwa seorang *user* jika ingin melakukan *preprocessing* RSA harus terlebih dahulu masuk kedalam tampilan menu utama, selanjutnya masuk kedalam perbandingan RSA.

2. *Sequence Hash MD5*

Kalkulasi hash MD5 digunakan untuk mengetahui langkah-langkah yang terjadi dalam pemrosesan setiap bloknya sampai perubahan data asli ke data yang tidak dimengerti dengan metode MD5, untuk lebih jelasnya dapat dilihat pada gambar III.3.



Gambar III.3. *Sequence Hash MD5*

Dari gambar III.3 menunjukkan bahwa seorang *user* jika ingin melakukan *preprocessing* MD5 harus terlebih dahulu masuk kedalam tampilan menu utama, selanjutnya masuk kedalam perbandingan MD5.

III.8. Desain Sistem Detail

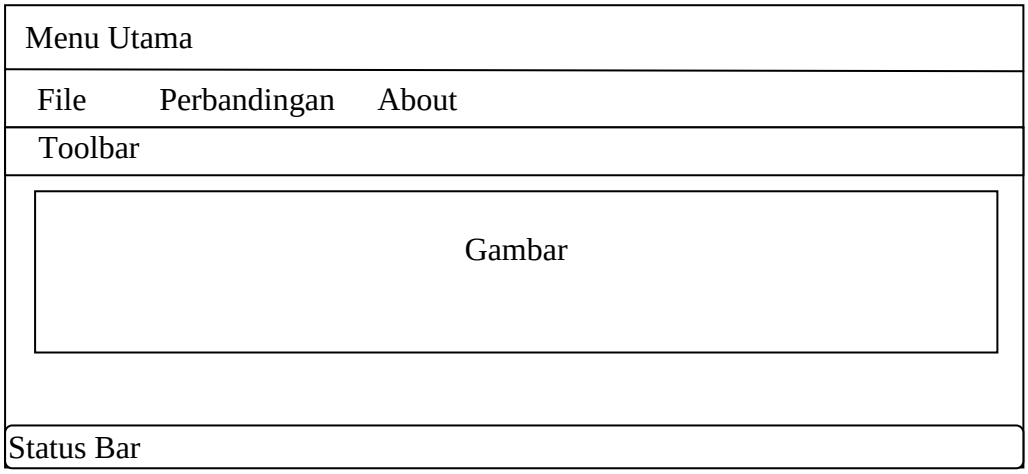
Perancangan terinci yang disebut juga desain teknis sistem secara fisik (*phisycal system design*) atau disebut juga desain internal (*internal design*), yaitu perancangan bentuk fisik atau bagan arsitektur sistem yang diusulkan. Dalam merancang suatu sistem perlu diketahui hal yang akan menunjang sistem, agar dapat mempermudah pengolahan data nantinya. Pengolahan data ini diharapakan dapat mempermudah dalam hal penyajian, pelayanan dan pembuatan berbagai laporan data yang dibutuhkan.

III.8.1 Desain *User Interface*

Desain sistem ini berisikan pemilihan menu dan hasil pencarian yang telah dilakukan. Adapun bentuk rancangan output dari sistem *enkripsi file* ini adalah sebagai berikut :

1. Menu Utama

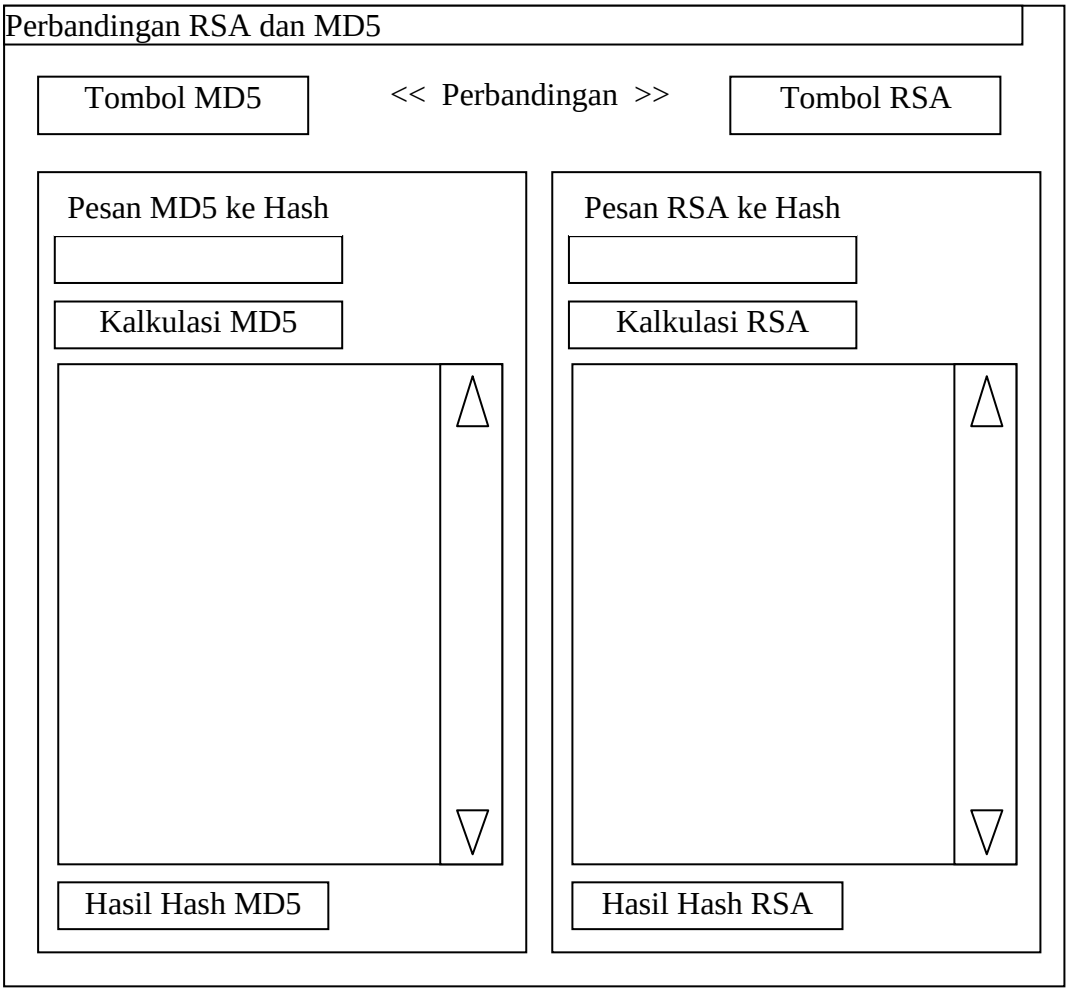
   Menu utama akan tampil dilayar monitor, adapun bentuk daripada menu utama dapat dilihat pada gambar III.4.



Gambar III.4. Desain Menu Utama

2. Form Perbandingan

Bentuk form perbandingan yang dirancang dapat dilihat pada gambar III.5.



Gambar III.5. Desain *Form* Perbandingan RSA dan MD5

III.9. Logika Program

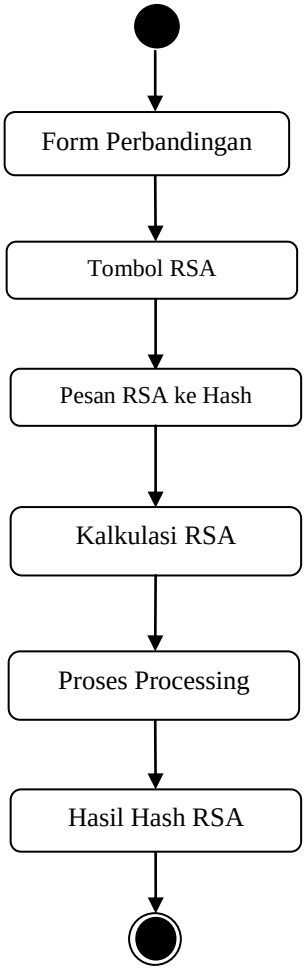
Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Perancangan algoritma dalam skripsi ini dituangkan ke dalam *flowchart*. Berikut perancangan algoritma-algoritma yang dibahas dalam perancangan sistem.

III.9.1. *Activity* Diagram

Activity diagram merupakan *activity* yang terdiri dari proses *enkripsi file* dan *deskripsi file. Activity* diagram ini ditunjukkan untuk penggambaran dasar aliran daripada *enkripsi* dan *deskripsi file* tersebut.

1. *Activity Diagram Hash* RSA

Activity diagram *hash RSA* merupakan *activity* diagram untuk proses *preprocessing*. *Activity* diagram *hash* RSA dapat dilihat pada gambar III.6.

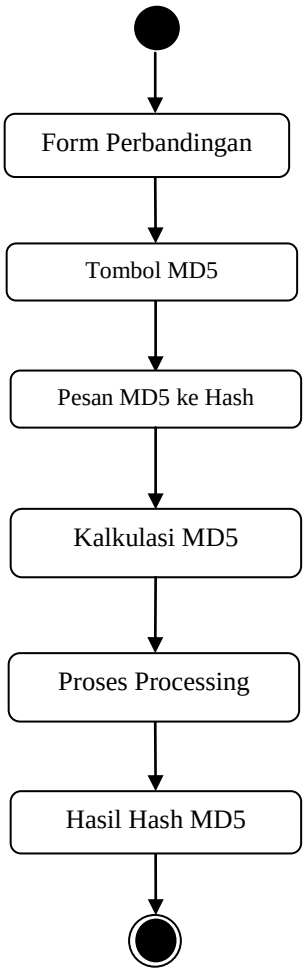


Gambar III.6. *Activity Diagram* RSA

Pada gambar III.6 menjelaskan bahwa pengguna menginputkan sebuah pesan yang ingin di hashkan, jika semuanya sudah diinputkan maka proses kalkulasi *hash* RSA dapat dilakukan, dan menemukan hasil hash RSA tersebut.

2. *Activity Diagram* MD5

Activity diagram *hash* MD5 merupakan *activity* diagram untuk proses *preprocessing*. *Activity* diagram *hash* MD5 dapat dilihat pada gambar III.7.



Gambar III.7. *Activity Diagram* MD5

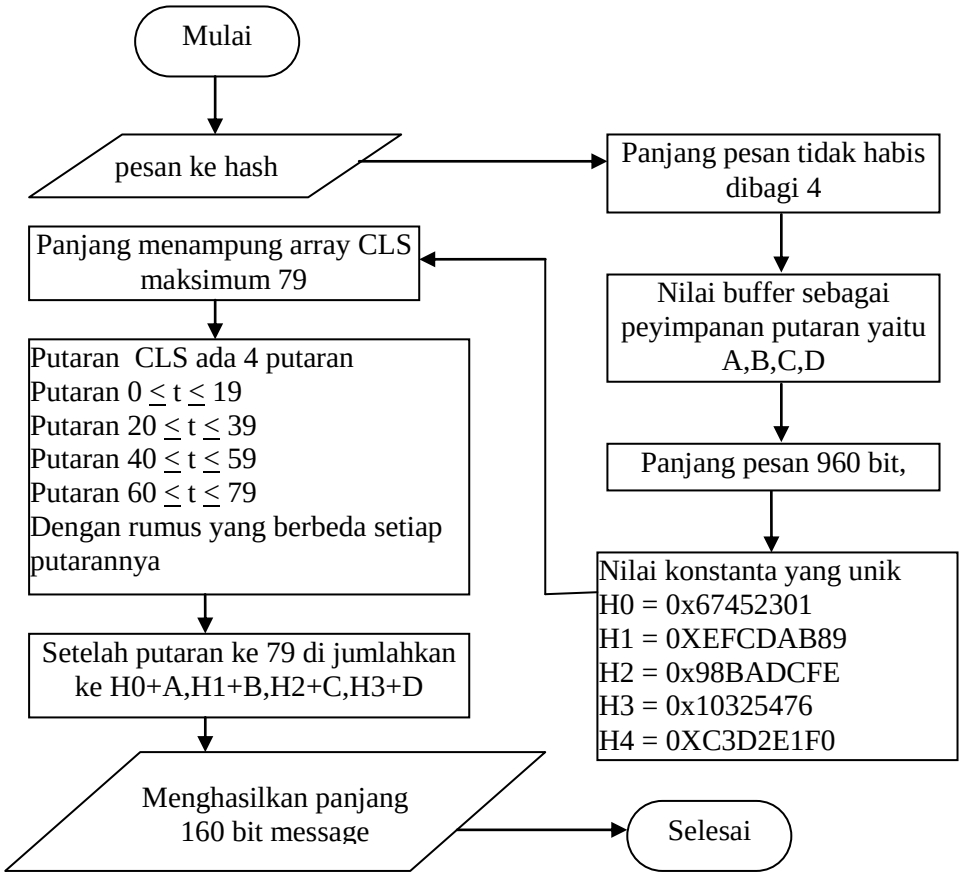
Pada gambar III.7 menjelaskan bahwa pengguna menginputkan sebuah pesan yang ingin di hashkan, jika semuanya sudah diinputkan maka proses kalkulasi *hash* MD5 dapat dilakukan, dan menemukan hasil hash MD5 tersebut.

III.9.2. Perbandingan RSA dan MD5 pada Flowchart

Metode perbandingan yang digunakan untuk perancangan keamanan data teks antara RSA dan MD5 menggunakan flowchart. Flowchart merupakan konsep perancangan untuk mengetahui proses dan jalannya aliran data.

1. *Flowchart* RSA

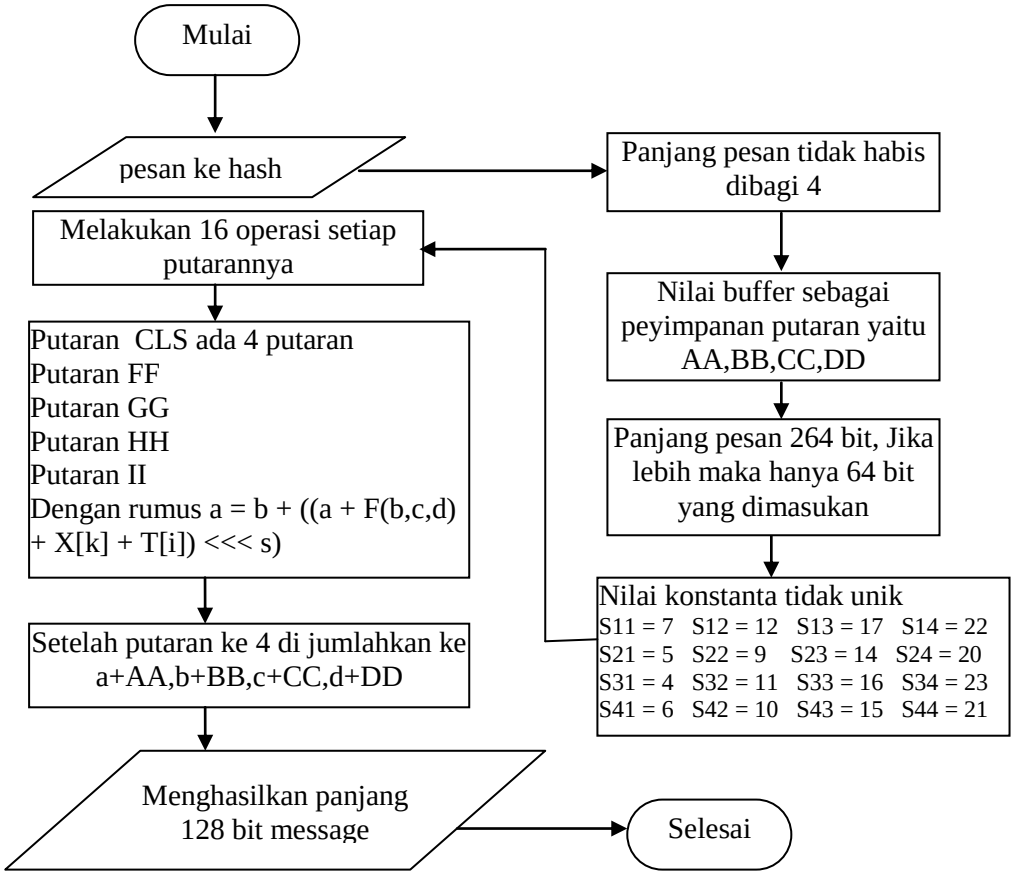
Flowchart hash RSA merupakan aliran data untuk proses *hash* RSA dapat dilihat pada gambar III.8.



Gambar III.8. *Flowchart* RSA

1. *Flowchart* MD5

Flowchart hash MD5 merupakan aliran data untuk proses *hash* MD5 dapat dilihat pada gambar III.9.



Gambar III.9. *Flowchart* MD5