BAB III

ANALISIS DAN PERANCANGAN SISTEM

3. 1. Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

Dalam proses pembangunan suatu sistem mutlak dilakukan penelitian dan penganalisaan tentang sistem yang akan dibangun. Berikut ini adalah tahapan-tahapan proses yang dilakukan untuk membangun Perangkat Lunak Steganografi menggunakan Teknik *Dynamic Cell Spreading* (DCS) dan *Digital Signature Algorithm* (DSA) untuk keamanan data, yaitu proses pembentukan sepasang kunci, proses pembentukan *digital signature*, proses *embedding*, proses *extracting*, dan proses verifikasi.

3. 1. 1. Analisis Masalah

Tahapan analisis masalah dilakukan terlebih dahulu sebelum tahapan perancangan sistem. Hal ini dilakukan agar masalah-masalah yang dihadapi dapat diketahui dengan jelas, bagaimana cara kerja algoritma yang digunakan, sampai solusi yang diajukan untuk permasalahan tersebut. Berdasarkan hasil penelitian,

masalah-masalah yang sering terjadi pada saat pengguna melakukan steganografi adalah sebagai berikut :

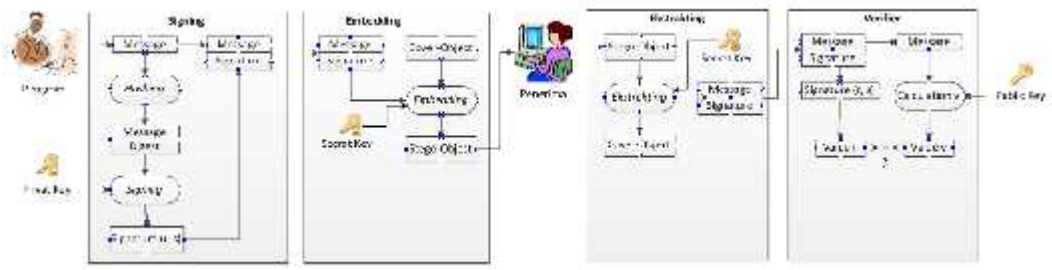
1. *Stego object* yang terkompresi dapat menghancurkan pesan rahasia yang tersembunyi.
2. Kemampuan *steganalisys* yang semakin berkembang dan *modern* mengakibatkan pesan rahasia yang tersembunyi mudah terdeteksi.
3. Steganografi masih memiliki kekurangan dalam hal mengamankan isi dari pesan rahasia.

3. 1. 2. Analisis Algoritma

Analisis algoritma digunakan untuk mengetahui alur proses dari algoritma yang digunakan untuk dapat diterapkan ke dalam perangkat lunak. Pada pembangunan perangkat lunak ini digunakan algoritma *Digital Signature Algorithm* (DSA) untuk membangkitkan nilai *digital signature* pada pesan dan mengautentikasi pesan yang sudah disisipi *digital signature*. Sedangkan teknik *Dynamic Cell Spreading* digunakan untuk menyisipkan pesan ke dalam file *cover object* dan mengekstraknya kembali. Adapun alur proses pada Perangkat Lunak Steganografi menggunakan Teknik *Dynamic Cell Spreading* (DCS) dan *Digital Signature Algorithm* (DSA) untuk keamanan data dapat dilihat pada gambar 3.1. Berikut ini adalah penjelasan mengenai alur proses perangkat lunak yang akan dibangun :

1. Pengirim membuat 2 buah pasang kunci, yaitu kunci privat dan kunci publik.
2. Pengirim memilih *file child* yang akan disisipkan ke dalam *file parent*..

3. Pengirim membuat nilai *digital signature* dari *file child* dengan menggunakan kunci privat.
4. Pengirim memilih *file parent* yang akan digunakan sebagai media pembawa *file child* yang akan disisipkan.
5. *File child, signature,* dan *public key* tersebut kemudian disisipkan ke dalam *file parent*.
6. *File parent* yang telah disisipi *file child* atau disebut *stego object* kemudian dikirim kepada penerima melalui jalur komunikasi.
7. Penerima kemudian mengekstraksi *stego object* yang diterima, untuk mendapatkan *file child* yang telah disembunyikan sebelumnya.
8. Penerima akan memverifikasi keaslian *file child* yang berhasil diekstrak menggunakan kunci publik pengirim yang telah disisipkan sebelumnya.
9. Apabila nilai *signature* pada *file child* yang disembunyikan bernilai sama dengan nilai *signature* yang telah disisipkan pada *file parent*, maka keabsahan *file child* tersebut *valid*, *file child* berasal dari pengirim yang benar dan tidak ada perubahan selama proses pengiriman, tetapi apabila tidak sama, maka *file child* tersebut *invalid*, *file child* bukan berasal dari pengirim yang benar atau ada perubahan isi *file child* pada saat pengiriman.



Gambar 3. 1 Gambaran Umum Sistem

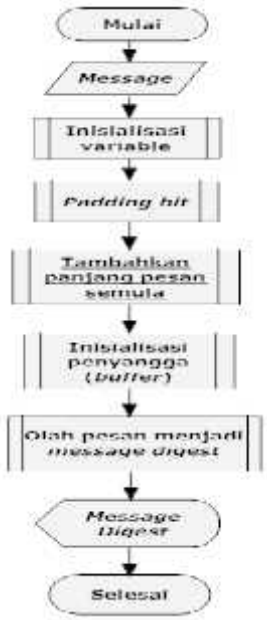
3. 1. 2. 1. Analisis Algoritma SHA-1

Fungsi Hash SHA-1 digunakan pada proses pembentukan *signature* dan verifikasi *signature* DSA. *SHA* adalah fungsi *hash* satu-arah yang dibuat oleh *NIST* dan digunakan bersama *DSS* (*Digital Signature Standard*). Oleh *NSA*, *SHA* dinyatakan sebagai standard fungsi *hash* satu-arah. *SHA* didasarkan pada *MD4* yang dibuat oleh Ronald L. Rivest dari *MIT*.

SHA disebut aman (*secure*) karena ia dirancang sedemikian rupa sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan. Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 264 bit (2.147.483.648 *gigabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh *MD5*. Langkah-langkah pembuatan *message digest* secara garis besar adalah sebagai berikut:

1. Masukkan pesan
2. Inisialisasi variable
3. Penambahan bit-bit pengganjal (*padding bits*).
4. Penambahan nilai panjang pesan semula.
5. Inisialisasi penyangga (*buffer*) *message digest*.
6. Pengolahan pesan dalam blok berukuran 512 bit.

Gambar 3.2 menunjukkan *flowchart* dari langkah-langkah pembuatan *message digest* menggunakan fungsi SHA-1.



Gambar 3. 2 Alur Proses Fungsi SHA-1

Tabel 3. 1 Pseudocode algoritma SHA-1

```
1   /*Catatan : Semua peubah adalah 32-bit tak tertanda dan penjumlahan
    dalam modulo 2^32 */
2   // Inisialisasi penyangga
3   Int A = 0x67452301
4   Int B = 0xEFCDAB89
5   Int C = 0x98BADCFE
6   Int D = 0x10325476
7   Int E = 0xC3D2E1F0
8   // Pemrosesan awal / padding bit
9   Append sebuah bit “1” ke pesan
10  While panjang pesan (dalam bit) kongruen dengan 448 (modulo 512) do
11        Append bit “0” ke pesan
12  Endwhile
13  Append panjang pesan (sebelum padding) as 64-bit big-endian integer
    to pesan
14  //proses pesan  dalam blok-blok yang berukuran 512 bit
15   bagi pesan ke dalam blok-blok berukuran 512-bit
16  For (tiap blok) do
17        Bagi tiap blok ke dalam 16 buah sub-blok 32-bit
18        //ekspansi 16 buah sub-blok 32-bit menjadi 80 buah sub-blok
    32-bit
19        For (i=16; i<=79; i++) do
20              W[i] = (w[i-3] xor w[i-14] xor w[i-16]) leftrotate 1
21        Endfor
22       // inisialisasi peubah penyangga
23       Int a = A
24       Int b = B
25       Int c = C
26       Int d = D
27       Int e = E
28  //proses utama
```

```
29      For (i=0; i<=79; i++) do
30            If (0<=i<=19) then
31                  f = (b and c) or ((not b) and d)
32                  k = 0x5A827999
33            Else If (20<=i<=39) do
34                  f = b xor c xor d
35                  k = 0x6ED9EBA1
36            Else If (40<=i<=59) do
37                  f = (b and c) or (b and d) or (c and d)
38                  k = 0x8F1BBCDC
39            Else If (60<=i<=79) do
40                  f = b xor c xor d
41                  k = 0xCA62C1D6
42            Endif
43            Temp = (a leftrotate 5) + f +e +k +w[i]
44            e = d
45            d = c
46            c = b leftrotate 30
47            b = a
48            a = temp
49      Endfor
50  //jumlahkan blok-blok ini
51      A = A + a
52      B = B + b
53      C = C + c
54      D = D +d
55      E = E + e
56  Endfor
57  Int digest = A append B append C append D append E
```

Berdasarkan *pseudocode* pada tabel 3.1, dapat dihitung nilai kompleksitas dari algoritma tersebut. Untuk menghitung kompleksitas suatu algoritma digunakan notasi Big O. Berikut perhitungan kompleksitas algoritma menggunakan notasi Big O [Tabel 3.2]

Tabel 3. 2 Notasi Big O pada *pseudocode* fungsi SHA-1

line	sintax	notasi
1	/*Catatan : Semua peubah adalah 32-bit tak tertanda dan penjumlahan dalam modulo 2^32 */	
2	// Inisialisasi penyangga	
3	Int A = 0x67452301	$O_{(1)}$
4	Int B = 0xEFCDAB89	$O_{(1)}$
5	Int C = 0x98BADCFE	$O_{(1)}$
6	Int D = 0x10325476	$O_{(1)}$
7	Int E = 0xC3D2E1F0	$O_{(1)}$
8	// Pemrosesan awal / *padding bit*	
9	Append sebuah bit “1” ke pesan	$O_{(1)}$
10	While panjang pesan (dalam bit) kongruen dengan 448 (modulo 512) do	$O_{(n)}$

line	sintax	notasi
11	Append bit “0” ke pesan	$O_{(1)}$
12	Endwhile	
13	Append panjang pesan (sebelum *padding*) as 64-bit big-endian integer to pesan	$O_{(1)}$
14	//proses pesan dalam blok-blok yang berukuran 512 bit	
15	bagi pesan ke dalam blok-blok berukuran 512-bit	$O_{(1)}$
16	For (tiap blok) do	$O_{(n)}$
17	Bagi tiap blok ke dalam 16 buah sub-blok 32-bit	$O_{(1)}$
18	//ekspansi 16 buah sub-blok 32-bit menjadi 80 buah sub-blok 32-bit	
19	For (i=16; i<=79; i++) do	$O_{(n)}$
20	W[i] = (w[i-3] xor w[i-14] xor w[i-16]) leftrotate 1	$O_{(1)}$
21	Endfor	
22	// inisialisasi peubah penyangga	
23	Int a = A	$O_{(1)}$
24	Int b = B	$O_{(1)}$
25	Int c = C	$O_{(1)}$
26	Int d = D	$O_{(1)}$
27	Int e = E	$O_{(1)}$
28	//proses utama	
29	For (i=0; i<=79; i++) do	$O_{(n)}$
30	If (0<=i<=19) then	$O_{(1)}$
31	f = (b and c) or ((not b) and d)	$O_{(1)}$
32	k = 0x5A827999	$O_{(1)}$
33	Else If (20<=i<=39) do	$O_{(1)}$
34	f = b xor c xor d	$O_{(1)}$
35	k = 0x6ED9EBA1	$O_{(1)}$
36	Else If (40<=i<=59) do	$O_{(1)}$
37	f = (b and c) or (b and d) or (c and d)	$O_{(1)}$
38	k = 0x8F1BBCDC	$O_{(1)}$
39	Else If (60<=i<=79) do	$O_{(1)}$
40	f = b xor c xor d	$O_{(1)}$
41	k = 0xCA62C1D6	$O_{(1)}$
42	Endif	
43	Temp = (a leftrotate 5) + f +e +k +w[i]	$O_{(1)}$
44	e = d	$O_{(1)}$
45	d = c	$O_{(1)}$
46	c = b leftrotate 30	$O_{(1)}$
47	b = a	$O_{(1)}$
48	a = temp	$O_{(1)}$
49	Endfor	
50	//jumlahkan blok-blok ini	
51	A = A + a	$O_{(1)}$
52	B = B + b	$O_{(1)}$
53	C = C + c	$O_{(1)}$
54	D = D +d	$O_{(1)}$
55	E = E + e	$O_{(1)}$
56	Endfor	
57	Int digest = A append B append C append D append E	$O_{(1)}$

$$F(n) = O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + (O_{(n)} \text{ x } O_{(1)}) + O_{(1)} + O_{(1)} + (O_{(n)} \text{ x } ($$
$$(O_{(1)} + (O_{(n)} \text{ x } O_{(1)}) + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + (O_{(n)} \text{ x } ((O_{(1)} + (O_{(1)}$$

$+ O_{(1)}))+ (O_{(1)} + (O_{(1)} + O_{(1)})) + (O_{(1)} + (O_{(1)} + O_{(1)}))+ (O_{(1)} + (O_{(1)} + O_{(1)}))$
$+ O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)})) + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)})$
$)+ O_{(1)}$

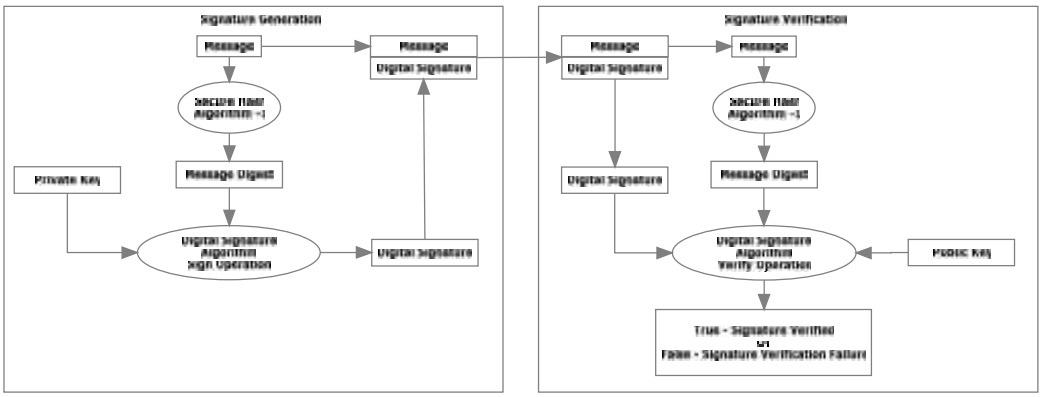
$= O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + O_{(1)}$

$= O_{(\max(1,1,1,1,1,1,n,1,1,n^2,1,))}$

$F(n) = O_{(n)}^2$

3. 1. 2. 2. Analisis Digital Signature Algorithm (DSA)

Seperti yang terlihat pada Gambar 3.3, algoritma DSA digunakan pada proses pemberian nilai *digital signature* dan proses verifikasi. Tanda tangan digital DSA berbentuk sepasang besar angka yang ditampilkan komputer sebagai string dari digit biner. Tanda-tangan digital dihitung dengan menggunakan sejumlah aturan dan sejumlah parameter sehingga identitas pemilik dan integritas data dapat diverifikasi. Pembuat tanda tangan menggunakan kunci privat untuk membuat tanda tangan, sedangkan kunci publik yang berkorespondensi dengan kunci privat namun tidak sama digunakan untuk memverifikasi tanda tangan. Setiap user memiliki sepasang kunci privat dan kunci publik. Kunci publik dapat disebar secara umum, sedangkan kunci privat tidak pernah disebarkan. DSA menggunakan fungsi *hash SHA (Secure Hash Algorithm)* untuk mengubah pesan menjadi *message digest* yang berukuran 160 bit. Gambar 3.3 menunjukkan gambaran konsep *digital signature algorithm* yang menggunakan fungsi *SHA-1*.



Gambar 3. 3 Konsep Digital Signature Algorithm

DSA didasarkan pada algoritma ElGamal. DSA mempunyai properti parameter sebagai berikut :

a. *p* adalah bilangan prima, parameter *p* bersifat publik dan dapat digunakan bersama-sama oleh orang di dalam kelompok. Parameter *p* digunakan untuk membuat kunci, membuat *signature* dan verifikasi.

b. *q* adalah bilangan prima, *q* merupakan faktor dari $p - 1$. Dengan kata lain, $(p - 1) \bmod q = 0$. Parameter *q* bersifat publik karena digunakan untuk membuat kunci, membuat *signature* dan verifikasi.

c. $g = h^{(p-1)/q} \bmod p$, yang dalam hal ini $h < p - 1$ sedemikian sehingga $h^{(p-1)/q} \bmod p > 1$. Parameter *g* bersifat publik dan digunakan untuk membuat kunci, membuat *signature* dan verifikasi.

d. *x* adalah bilangan bulat kurang dari *q*. Parameter *x* merupakan kunci privat yang digunakan untuk membuat *signature* pada pesan.

e. $y = g^x \bmod p$ adalah kunci publik. Parameter *y* berkorespondensi dengan *x* dan digunakan untuk memverifikasi pesan.

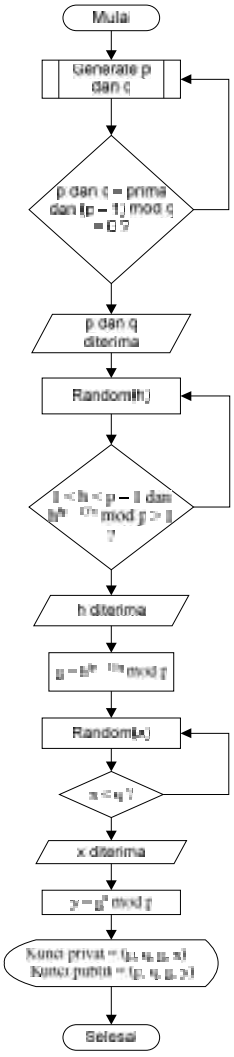
f. *m* adalah pesan yang akan diberi tanda tangan.

3. 1. 2. 2. 1. **Analisis Prosedur Pembangkitan Sepasang Kunci**

Sepasang kunci diperlukan untuk membangkitkan nilai *digital signature* pada pesan dan memverifikasi keabsahan pesan. Kunci-kunci tersebut adalah kunci privat dan kunci publik. Tanpa kunci-kunci tersebut pembentukan dan verifikasi *digital signature* tidak dapat dilakukan. Prosedur pembangkitan sepasang kunci pada algoritma DSA akan dijelaskan sebagai berikut :

1. *Generate* bilangan prima *p* dan *q*, yang dalam hal ini $(p - 1) \bmod q = 0$.
2. Hitung $g = h^{(p-1)/q} \bmod p$, yang dalam hal ini $1 < h < p - 1$ dan $h^{(p-1)/q} \bmod p > 1$.
3. *Generate* kunci privat *x*, yang dalam hal ini $x < q$.
4. Hitung kunci publik $y = g^x \bmod p$.
5. Kunci privat = (*p*, *q*, *g*, *x*) dan kunci publik = (*p*, *q*, *g*, *y*)

   Berikut *flowchart* prosedur pembangkitan sepasang kunci :



Gambar 3. 4 *Flowchart* Prosedur Pembangkitan Sepasang Kunci

Tabel 3. 3 *Pseudocode* Pembangkitan Sepasang Kunci

```
1   Procedure BuatKunci(input p, q, x : integer,
    output Kv, Kb : string)
2   {Membuat kunci publik dan kunci privat
3    Masukan : p, q, h, x
4    Keluaran : Kv, Kb
5   }
6   Deklarasi
7   h, g : integer
8   Algoritma
9   Repeat {
10    p ← Probablyprime(p)  //fungsi pembangkit nilai acak bilangan
11    q ← Probablyprime(q)} //prima
12  Until(((p – 1) mod q) = 0)
13  Endrepeat
```

```
14  Repeat {
15    h ← random() }  //fungsi pembangkit bilangan acak
16  Until ((1 < h < p) and ((h ^ (( p – 1 )/ q ) mod p) > 1)
17  Endrepeat
18  g ← h ^ (( p – 1 ) / q ) mod p
19  Repeat {
20    x ← random()}   //fungsi pembangkit bilangan acak
21  Until (x < q)
22  Endrepeat
23  y ← (g ^ x) mod p
24  Kv ← p, q, g, x
25  Kb ← p, q, g, y
```

Jadi, pada prosedur pembangkitan sepasang kunci [Tabel 3.3] menghasilkan :

Kunci publik dinyatakan sebagai (*p, q, g, y*);

Kunci privat dinyatakan sebagai (*p, q, g, x*);

Berdasarkan *pseudocode* pada tabel 3.3, dapat dihitung nilai kompleksitas dari algoritma tersebut. Untuk menghitung kompleksitas suatu algoritma digunakan notasi Big O. Berikut perhitungan kompleksitas algoritma menggunakan notasi Big O :

Tabel 3. 4 Notasi Big O *Pseudocode* Pembangkit Sepasang Kunci

line	syntax	notasi
9	Repeat {	$O_{(1)}$
10	p ← Probablyprime(p) //fungsi pembangkit nilai acak bilangan prima	$O_{(1)}$
11	q ← Probablyprime(q)}	$O_{(1)}$
12	Until(((p – 1) mod q) = 0)	$O_{(n)}$
13	Endrepeat	
14	Repeat {	$O_{(1)}$
15	h ← random() } //fungsi pembangkit bilangan acak	$O_{(1)}$
16	Until ((1 < h < p) and ((h ^ ((p – 1)/ q) mod p) > 1)	$O_{(n)}$
17	Endrepeat	
18	g ← h ^ ((p – 1) / q) mod p	$O_{(1)}$
19	Repeat {	$O_{(1)}$
20	x ← random()} //fungsi pembangkit bilangan acak	$O_{(1)}$
21	Until (x < q)	$O_{(n)}$
22	Endrepeat	
23	y ← (g ^ x) mod p	$O_{(1)}$
24	Kv ← p, q, g, x	$O_{(1)}$
25	Kb ← p, q, g, y	$O_{(1)}$

$F(n) = (O_{(1)} \text{ x } (O_{(1)} + O_{(1)}) \text{ x } O_{(n)}) + (O_{(1)} \text{ x } O_{(1)} \text{ x } O_{(n)}) + O_{(1)} + (O_{(1)} \text{ x } O_{(1)} \text{ x } O_{(n)}) + O_{(1)} + O_{(1)} + O_{(1)}$

$= (O_{(1)} \text{ x } O_{(\max(1,1,))} \text{ x } O_{(n)}) + (O_{(1)} \text{ x } O_{(1)} \text{ x } O_{(n)}) + O_{(1)} + (O_{(1)} \text{ x } O_{(1)} \text{ x } O_{(n)}) + O_{(1)} + O_{(1)} + O_{(1)}$

$= O_{(n)} + O_{(n)} + O_{(1)} + O_{(n)} + O_{(1)} + O_{(1)} + O_{(1)}$

$= O_{(\max(n,n,1,n,1,1,1))}$

$F(n) = O_{(n)}$

3. 1. 2. 2. 2. **Analisis Pembentukan *Digital Signature***

Tanda tangan digital (*digital signature*) merupakan protokol yang dijalankan untuk dapat memverifikasi keaslian dan keutuhan dari pesan yang akan dikirimkan dalam suatu saluran komunikasi.

Prosedur pembentukan *digital signature* pada algoritma DSA meliputi :

1. Masukkan pesan dan kunci privat.
2. Ubah pesan *m* menjadi *message digest* dengan menggunakan fungsi *hash SHA-1, H*.
3. Tentukan bilangan acak *k* dimana $k < q$.
4. Tanda-tangan dari pesan *m* adalah bilangan *r* dan *s*. Hitung *r* dan *s* sebagai berikut :

   $r = (g^k \bmod p) \bmod q$

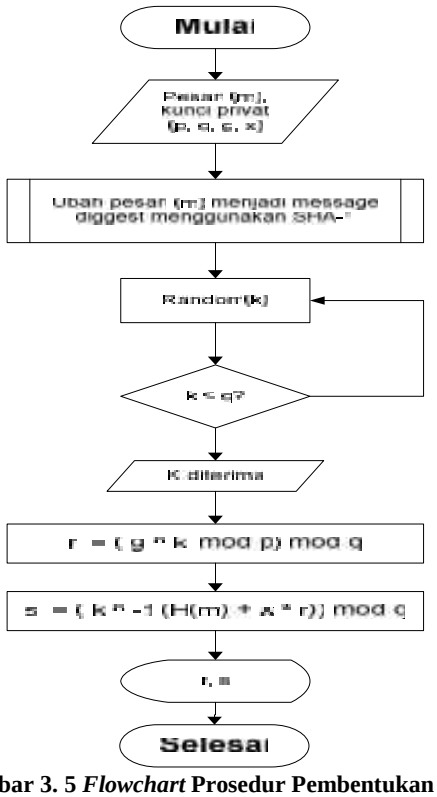
   $s = (k^{-1} (H(m) + x * r)) \bmod q$
5. Kirim pesan *m* beserta tanda-tangan *r* dan *s*.

Berikut *pseudocode* dan *flowchart* dari prosedur pembentukan *digital signature* :

Tabel 3. 5 *Pseudocode* Pembentukan *Digital Signature*

```
1   Procedure PembentukanDigitalSignature (Input m : file; k, r
    :integer; p, q, g, x : string; Output  s : string, m' : file)

2   { Membuat nilai digital signature dari pesan (m)
3     Masukan : m, k
4     Keluaran : r, s
5   }
6   Deklarasi
7   H(m) : string
8   Algoritma
9   m ← m
10  p ← p
11  q ← q
12  g ← g
13  x ← x
14  H(m) ← SHA-1(m)          {Ubah pesan menjadi message digest
                             menggunakan fungsi SHA-1}
15  Repeat
16     k ← random()
17  Until (k < q) { cek k < q }
18  Endrepeat
19  r ← ( g ^ k  mod p) mod q
20  s ← ( k ^ -1 (H(m) + x * r)) mod q
```



Gambar 3. 5 *Flowchart* Prosedur Pembentukan *Digital Signature*

Jadi pada prosedure pembentukan *digital signature* menghasilkan nilai *signature r* dan *s*. Nilai *signature* tersebut disisipkan ke dalam pesan (*m*) sehingga menghasilkan pesan *m, r, s.*

Berikut perhitungan kompleksitas algoritmanya :

Tabel 3. 6 Notasi Big O *Pseudocode* Pembentukan Digital Signature

line	syntax	notasi
9	m ← m	$O_{(1)}$
10	p ← p	$O_{(1)}$
11	q ← q	$O_{(1)}$
12	g ← g	$O_{(1)}$
13	x ← x	$O_{(1)}$
14	H(m) ← SHA-1(m)	$O_{(n)}^2$
15	Repeat	$O_{(1)}$
16	k ← random()	$O_{(1)}$
17	Until (k < q) { *cek k < q* }	$O_{(n)}$
18	Endrepeat	
19	r ← (g ^ k mod p) mod q	$O_{(1)}$
20	s ← (k ^ -1 (H(m) + x * r)) mod q	$O_{(1)}$
21	m ← m	$O_{(1)}$
22	p ← p	

$$F(n) = O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + (O_{(1)} \text{ x } O_{(1)} \text{ x } O_{(n)}) + O_{(1)} + O_{(1)} + O_{(1)}$$

$$= O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + O_{(n)} + O_{(1)} + O_{(1)} + O_{(1)}$$

$$= O_{(\max(1,1,1,1,1,n^2,n,1,1,1))}$$

$$F(n) = O_{(n)}^2$$

3. 1. 2. 2. 3. **Analisis Verifikasi Digital Signature**

Proses verifikasi adalah proses memastikan keaslian si pengirim. Artinya apakah yang mengirimkan itu asli dari orang yang bersangkutan. Prosedur verifikasi keabsahan *digital signature* pada algoritma DSA meliputi :

1. Masukkan pesan yang sudah disisipkan tanda tangan dan kunci publik.
2. Hitung

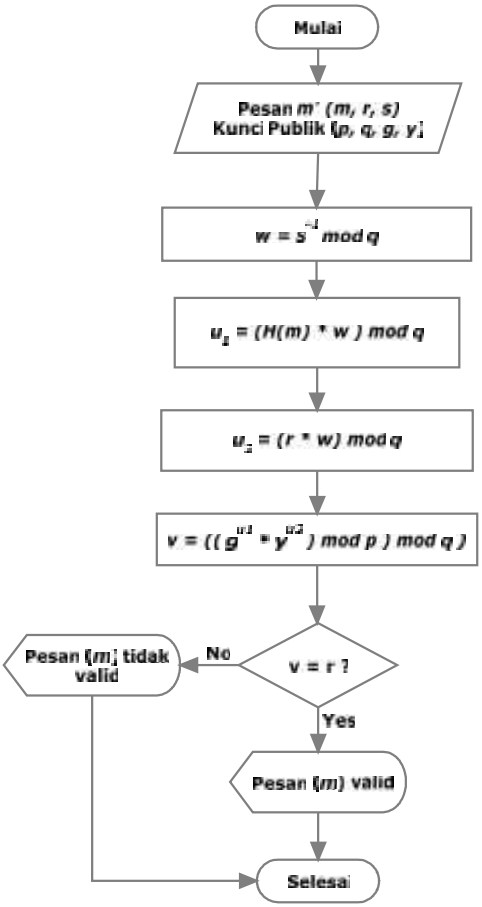
   $w = s^{-1} \bmod q$

   $u_1 = (H(m) * w) \bmod q$

   $u_2 = (r * w) \bmod q$

   $v = ((g^{u1} * y^{u2}) \bmod p) \bmod q)$
3. Jika $v = r$, maka tanda-tangan sah, yang berarti bahwa pesan masih asli dan dikirim oleh pengirim yang benar.

   Berikut *flowchart* dari prosedure verifikasi *digital signature* :



Gambar 3. 6 *Flowchart* Prosedur Verifikasi Keabsahan *Digital Signature*

Tabel 3. 7 *Pseudocode* Prosedure Verifikasi Keabsahan *Digital Signature*

```
1  Procedure verifikasi (input m, r, s : file; p, q, g, y :
   string; output valid : boolean)
2  {Memverifikasi keaslian pesan menggunakan kunci publik
   pengirim
3  Masukan : m, r, s, p, q, g, y
4  Keluaran : true or false
5  }
6  Deklarasi
7  w, u1, u2, v : integer
8  Algoritma
9  m ← m
10 r ← r
11 s ← s
12 p ← p
13 q ← q
14 g ← g
15 y ← y
16 w ← s ^ -1 mod q
17 u1 ← (SHA-1(m) * w) mod q     {hashing pesan m menggunakan
   fungsi SHA-1}
18 u2 ← (r * w) mod q
19 v ← (((g ^ u1) * (y ^ u2)) mod p) mod q)
20 if v = r then
21 valid ← true
22 Else
23 valid ← false
24 Endif
```

Berikut perhitungan kompleksitas algoritmanya :

Tabel 3. 8 Notasi Bi O *Pseudocode* Prosedure Verifikasi Keabsahan *Digital Signature*

line	syntax	notasi
9	m ← m	$O_{(1)}$
10	r ← r	$O_{(1)}$
11	s ← s	$O_{(1)}$
12	p ← p	$O_{(1)}$
13	q ← q	$O_{(1)}$
14	g ← g	$O_{(1)}$
15	y ← y	$O_{(1)}$
16	w ← s ^ -1 mod q	$O_{(1)}$
17	u1 ← (SHA-1(m) * w) mod q *{hashing pesan m menggunakan fungsi SHA-1}*	$O_{(n)^2}$
18	u2 ← (r * w) mod q	$O_{(1)}$
19	v ← (((g ^ u1) * (y ^ u2)) mod p) mod q)	$O_{(1)}$
20	if v = r then	$O_{(1)}$
21	valid ← true	$O_{(1)}$
22	Else	
23	valid ← false	$O_{(1)}$
24	Endif	

$$F(n) = O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + O_{(1)} + O_{(1)} + (O_{(1)} + O_{(1)} + O_{(1)})$$

$$= O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + O_{(1)} + O_{(1)} + (O_{(max(1,1,1))})$$

$$= O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(1)} + O_{(n)}^2 + O_{(1)} + O_{(1)} + O_{(1)}$$

$$= O_{(max(1,1,1,1,1,1,1,1,n^2,1,1,1))}$$

$$F(n) = O_{(n)}^2$$

3. 1. 2. 2. 4. **Contoh Perhitungan Algoritma DSA**

Berikut adalah contoh perhitungan pada algoritma DSA :

a. Prosedure Pembangkitan Sepasang Kunci

1. Pilih bilangan prima *p* dan *q*, yang dalam hal ini $(p - 1) \bmod q = 0$

   $p = 59419$

   $q = 3301$ (memenuhi $3301 * 18 = 59419 - 1$)

2. $g = h^{(p-1)/q} \bmod p$, yang dalam hal ini $1 < h < p - 1$ dan $h^{(p-1)/q} \bmod p > 1$.

   $g = 18870$ (dengan $h = 100$)

3. Tentukan kunci privat *x*, yang dalam hal ini $x < q$.

   $x = 3223$

4. Hitung kunci publik $y = g^x \bmod p$.

   $y = 29245$

b. Prosedur Pembentukan *Digital Signature*

1. Hitung nilai *hash* dari pesan menggunakan fungsi *SHA-1*, misalkan *H(m)* = 4321
2. Tentukan bilangan acak $k < q$.

   $k = 997$

   $k^{-1} = 2907 \pmod{3301}$
3. Hitung *r* dan *s* sebagai berikut :

   $r = (g^k \bmod p) \bmod q = 848$

   $s = (k^{-1} (H(m) + x * r)) \bmod q$

   $= 7957694475 \bmod 3301 = 183$
4. Kirim pesan *m* beserta tanda-tangan *r* dan *s*.

c. Prosedure Verifikasi Keabsahan *Digital Signature*

1. Hitung

   $s^{-1} = 469 \pmod{3301}$

   $w = s^{-1} \bmod q = 469$

   $u_1 = (H(m) * w) \bmod q = 2026549 \bmod 3301 = 3036$

   $u_2 = (r * w) \bmod q = 397712 \bmod 3301 = 1592$

   $v = ((g^{u1} * y^{u2}) \bmod p) \bmod q) = 848 \bmod 3301 = 848$
2. Karena $v = r$, maka tanda-tangan sah.

3. 1. 2. 3. Analisis Teknik *Dynamic Cell Spreading* (DCS)

Teknik *Dynamic Cell Spreading* (DCS) merupakan steganografi dengan menggunakan model proteksi terhadap deteksi yang dikembangkan oleh *Holger Ohmacht* dengan konsep dasar yaitu menyembunyikan file pesan (semua data

elektronik) ke dalam media *digital*. Penyembunyian pesan dilakukan dengan cara menyisipkan pada bit rendah LSB (*Least Significant Bit*) dari data *digital* yang menyusun file tersebut menggunakan *buffer memory* sebagai media penyimpanan sementara. Teknik *Dynamic Cell Spreading* memiliki 2 proses utama, yaitu proses *embedding* dan *extracting*.

Proses penggabungan *file child* dengan *file parent* membutuhkan media memori sebagai perantara untuk menghitung jumlah keseluruhan bit yang terdapat di dalam *file child* maupun *file parent* sehingga memudahkan dalam proses *embedding* itu sendiri.

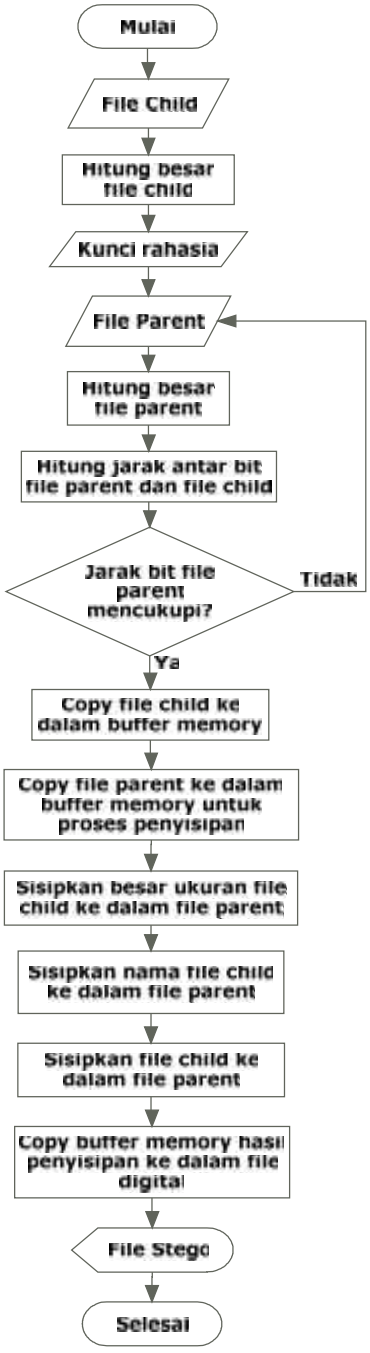
3. 1. 2. 3. 1. **Analisis Proses Embedding**

Proses *embedding* adalah proses menyisipkan *file child* ke dalam *file parent* sehingga menghasilkan *file stego object* yang berisi *file child* yang disembunyikan. Proses *embedding* dalam teknik DCS mempunyai beberapa tahapan yaitu :

a. Masukkan *file child, file parent* dan kunci rahasia.
b. Mengkalkulasi besar dan jarak antar bit yang ada pada *file child* dan *file parent* agar mempermudah penghitungan dan penyisipan bit data yang akan dimasukkan, apabila ukuran *file parent* tidak mencukupi maka pengguna harus mengganti *file parent* dengan ukuran yang lebih besar.
c. Mengkopi *file child* dan *file parent* ke dalam *buffer memory.*
d. Menyisipkan ukuran *file child* ke dalam *file parent*.
e. Menyisipkan nama *file child* ke dalam *file parent.*

f. Menyisipkan bit *file child* ke dalam bit rendah *Least Significant Bit (LSB) file parent* dengan jarak sesuai dengan kunci rahasia yang diberikan.

g. Mengkopi *buffer memory* ke bentuk media *digital*, mengubah kembali dari memori menjadi file *digital* atau disebut *file stego*.



Gambar 3. 7 *Flowchart* proses *embedding*

3. 1. 2. 3. 2. **Analisis Proses Extracting**

Proses ekstraksi adalah proses memisahkan *file child* yang ada pada *file stego object*. Proses ekstraksi pesan pada teknik *Dynamic Cell Spreading* memiliki beberapa tahapan sebagai berikut :

a. Masukkan file *stego object* dan kunci rahasia yang diberikan oleh pengirim.
b. Meng-*copy* file *stego object* ke dalam *buffer memory*.
c. Alokasi *buffer memory* untuk menampung file hasil ekstrak.
d. Ekstrak ukuran *file child* yang sebelumnya telah disisipkan guna mengetahui besar ukuran file yang disisipkan.
e. Ekstrak nama *file child* yang sebelumnya telah disisipkan guna mengetahui nama dan ekstensi file yang disisipkan.
f. Ekstrak file dengan menggunakan kunci rahasia yang diberikan oleh pengirim agar file bisa kembali utuh.
g. Copy hasil ekstrak dari *buffer memory* ke dalam *file digital* untuk mendapatkan file secara utuh.



Gambar 3. 8 *Flowchart* Proses Ekstraksi

3. 1. 3. Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak yang akan dibangun berdasarkan kebutuhan pengguna pada jurnal dan artikel yang ada. Spesifikasi kebutuhan perangkat lunak akan dibagi ke dalam dua bagian yaitu SKPL-F (spesifikasi kebutuhan perangkat lunak fungsional) [Tabel 3.9] dan SKPL-NF (spesifikasi kebutuhan perangkat lunak non-fungsional) [Tabel 3.10]. Berikut ini adalah tabel dari spesifikasi kebutuhan perangkat lunak dalam penelitian ini.

Tabel 3. 9 Spesifikasi Kebutuhan Perangkat Lunak Fungsional

Kode	Kebutuhan
SKPL-F01	Sistem dapat membuat 2 buah pasang kunci, yaitu kunci privat dan kunci publik.
SKPL-F02	Sistem dapat mengambil file dari media penyimpanan.
SKPL-F03	Sistem dapat membuat *signature* dari pesan.
SKPL-F04	Sistem dapat menyisipkan file ke dalam file *digital*.
SKPL-F05	Sistem dapat menyimpan file yang sudah disisipkan file.
SKPL-F06	Sistem dapat mengekstraksi pesan tersembunyi dari media *digital.*
SKPL-F07	Sistem dapat menyimpan file hasil ekstraksi.
SKPL-F08	Sistem dapat mengautentikasi file hasil ekstraksi.

Tabel 3. 10 Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional

Kode	Kebutuhan
SKPL-NF01	Pengguna adalah orang yang memahami konsep steganografi dan *digital signature.*
SKPL-NF02	Perangkat keras yang digunakan adalah komputer dengan spesifikasi minimal Processor Dual Core 2.0 GHz dan memori RAM minimal 512 Mb.
SKPL-NF03	Perangkat Lunak ini akan dibangun menggunakan Java.
SKPL-NF04	Perangkat Lunak ini membutuhkan Java SDK (*Software Development Kit*) untuk dapat dijalankan di berbagai platform *Operating System* (OS).
SKPL-NF05	Ukuran file *parent* minimal 8x lebih besar dari file *child.*
SKPL-NF06	Format file *parent* yang digunakan terbagi menjadi : a. Dokumen Gambar : bmp, gif, jpg dan jpeg. b. Dokumen Audio : mp3 dan wav. c. Dokumen Video : mp4, avi, flv, mkv dan 3gp.
SKPL-NF07	Format file *child* yang akan disisipkan terbagi menjadi : a. Dokumen Teks : txt, rtf, doc, docx, ppt, pptx, xls, xlsx dan pdf. b. Dokumen Gambar : bmp, gif, jpg dan jpeg. c. Dokumen Audio : mp3 dan wav. d. Dokumen Video : mp4, avi, flv, mkv dan 3gp.

3. 1. 4. Analisis Kebutuhan Non Fungsional

Analisis kebutuhan fungsional digunakan untuk memastikan bahwa perangkat lunak yang dibuat sesuai dengan kebutuhan pengguna dan layak untuk dikembangkan. Tahapan analisis kebutuhan fungsional dibagi menjadi beberapa

tahap yaitu analisis perangkat keras, perangkat lunak, dan perangkat pikir. Tahapan-tahapan ini sangat penting untuk membangun suatu perangkat lunak.

3. 1. 4. 1. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam membangun perangkat lunak ini memiliki spesifikasi sebagai berikut :

1. Processor Intel Core i7 2.00 GHz,
2. Memori RAM 4 Gb,
3. Harddisk 600 Gb,
4. Resolusi layar monitor 1366 x 768 pixel.

Sedangkan spesifikasi minimum yang dibutuhkan untuk menjalankan perangkat lunak ini adalah sebagai berikut :

1. Processor Dual Core 2.0 GHz atau setara,
2. Memori RAM 1 Gb,
3. Harddisk 80 Gb,
4. Resolusi layar monitor 1024 x 768 pixel.

3. 1. 4. 2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak pendukung untuk dapat menjalankan perangkat lunak Steganografi dan *Digital Signature* ini adalah JSDK (*Java Software Development Kit*) yang di dalamnya terdapat *Java Sources* dan JRE (*Java Runtime Environment*) agar dapat digunakan di berbagai macam platform *Operating System* (OS).

3. 1. 4. 3. Analisis Pemakai

Untuk menggunakan perangkat lunak ini, dibutuhkan 2 pengguna dengan karakteristik seperti pada tabel 3.11.

Tabel 3. 11 Karakteristik pengguna

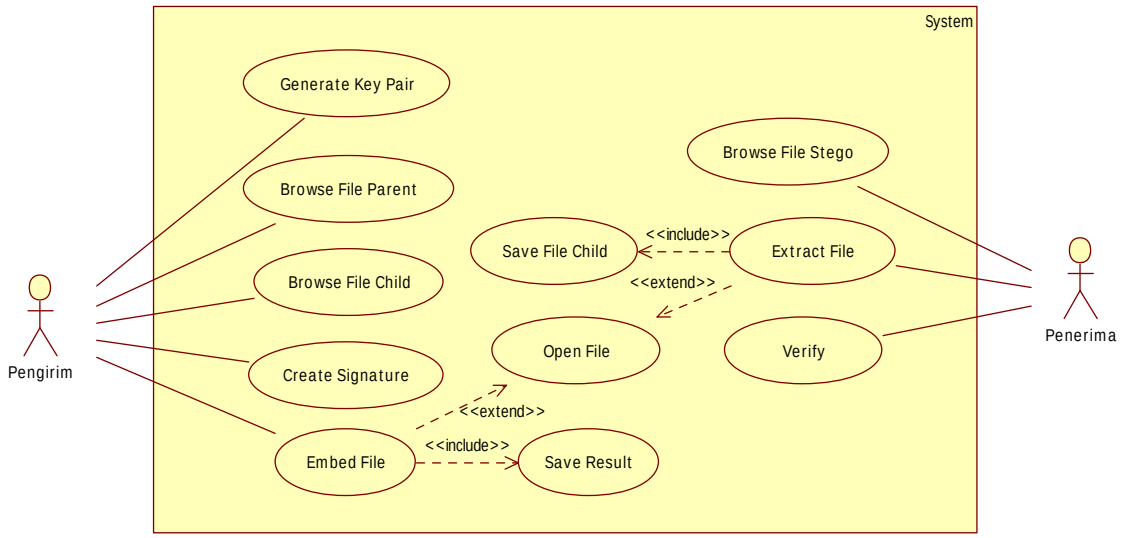
Pengguna	Hak Akses	Tingkat Pendidikan	Tingkat Keterampilan	Pengalaman
Pengirim	Membuat kunci,membuat *digital signature* pada pesan, menyisipkan pesan	Minimal SMA atau setara	Memahami konsep steganografi dan *Digital Signature*	Minimal 6 bulan dalam bidang *security data*
Penerima	Deteksi pesan, ekstraksi pesan, otentikasi pesan	Minimal SMA atau setara	Memahami konsep steganografi dan *Digital Signature*	Minimal 6 bulan dalam bidang *security data*

3. 1. 5. Analisis Kebutuhan Fungsional

Kebutuhan fungsional dianalisis dengan memodelkan sistem. Pemodelan yang digunakan untuk memodelkan perangkat lunak ini adalah pemodelan berorientasi objek. Perangkat lunak ini dimodelkan menggunakan UML (*Unified Modeling Language*). *Tools* yang digunakan adalah *Use Case Diagram, Sequence Diagram,* dan *Activity Diagram.*

3. 1. 5. 1. **Use Case Diagram**

Use case diagram perangkat lunak steganografi dan digital signature ini dapat dilihat pada gambar 3.9.



Gambar 3. 9 *Use Case* Diagram

3. 1. 5. 2. **Skenario *Use Case***

Skenario *Use Case* berfungsi untuk menjelaskan lebih detail mengenai modul-modul yang terdapat dalam perangkat lunak.

Berikut ini adalah skenario *use case* yang terdapat terjadi pada sistem steganografi dan *digital signature*.

Tabel 3. 12 Skenario *Use Case Generate Key*

Identifikasi	
Use Case Name	Generate Key
Actor	Pengirim
Use Case Description	Membuat sepasang kunci yaitu kunci privat dan kunci publik
Trigger	Pengirim memilih menu embed file dan menekan tombol Generate Key
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Pengirim memilih menu embed file	
2. Pengirim menekan tombol Generate Key	
3. Pengirim mendapatkan sepasang kunci	
Extension	
a. Jika gagal generate key a.1. Pengirim mendapatkan pesan gagal membuat kunci. a.2. Pengirim mengulangi langkah skenario no. 2.	
Assumptions	Pengirim belum memiliki atau ingin merubah kunci

Tabel 3. 13 Skenario *Use Case Browse File Parent*

Identifikasi	
Use Case Name	Browse File Parent
Actor	Pengirim
Use Case Description	Pengirim memilih file parent yang digunakan sebagai media penyisipan
Trigger	Pengirim berada di menu embed file dan menekan tombol Browse File Parent
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Pengirim berada di form embed file	
2. Pengirim menekan tombol browse file parent	
3. Pengirim memilih file parent	
Extension	
a. Jika Pengirim menekan tombol cancel a.1. Pengirim membatalkan memilih file parent	
Assumptions	Pengirim belum memilih atau ingin merubah file parent

Tabel 3. 14 Skenario *Use Case Browse File Child*

Identifikasi	
Use Case Name	Browse File Child
Actor	Pengirim
Use Case Description	Pengirim memilih file child yang akan ditandatangani dan disisipkan ke dalam file parent
Trigger	Pengirim berada di menu embed file dan menekan tombol Browse File Child
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Pengirim berada di form embed file	
2. Pengirim menekan tombol browse file child	
3. Pengirim memilih file child	
Extension	
a. Jika Pengirim menekan tombol cancel a.1. Pengirim membatalkan memilih file child	
Assumptions	Pengirim belum memilih atau ingin merubah file child

Tabel 3. 15 Skenario Use Case Create Signature

Identifikasi	
Use Case Name	Create Signature
Actor	Pengirim
Use Case Description	Pengirim membuat signature dari file child yang sudah dipilih
Trigger	Pengirim sudah memilih file child dan memiliki sepasang kunci lalu menekan tombol create signature
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Pengirim berada di form embed file	
2. Pengirim menekan tombol create signature	
3. Pengirim mendapatkan nilai signature dari file child	
Extension	
a. Jika gagal membuat signature a.1. Pengirim mendapatkan pesan gagal create signature. a.2. Pengirim menekan tombol create signature. b. Jika file child tidak ditemukan b.1. Pengirim mendapatkan pesan file child tidak ditemukan. b.2. Pengirim memilih file child. b.3. Pengirim menekan tombol create signature.	
Assumptions	Pengirim memiliki sepasang kunci dan sudah memilih file child

Tabel 3. 16 Skenario Use Case Embedding

Identifikasi	
Use Case Name	Embed File
Actor	Pengirim
Use Case Description	Pengirim menyisipkan file child ke dalam file parent
Trigger	Pengirim memilih file child, file parent, memiliki sepasang kunci dan sudah membuat signature file child, lalu menekan tombol proses.
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Pengirim berada di form embed file	
2. Pengirim menekan tombol proses	
3. Pengirim mendapatkan file stego	
Extension	
a. Jika sepasang kunci tidak ditemukan a.1. Pengirim menekan tombol generate key a.2. Pengirim menekan kembali tombol proses b. Jika file child tidak ditemukan b.1. Pengirim memilih file child b.2. Pengirim menekan kembali tombol proses c. Jika file parent tidak ditemukan c.1. Pengirim memilih file parent c.2. Pengirim menekan kembali tombol proses d. Jika signature tidak ditemukan d.1. Pengirim menekan tombol create signature d.2. Pengirim menekan kembali tombol proses e. Jika gagal embed file e.1. Pengirim menekan kembali tombol proses	
Assumptions	Pengirim sudah memiliki sepasang kunci, file child, file parent dan signature.

Tabel 3. 17 Skenario Use Case Browse File Stego

Identifikasi	
Use Case Name	Browse File Stego
Actor	Penerima
Use Case Description	Penerima memilih file stego yang akan diekstrak
Trigger	Penerima berada di form extract file dan menekan tombol browse
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Penerima berada di form extract file	
2. Penerima menekan tombol browse	
3. Penerima memilih file stego	
Extension	
a. Jika Pengirim menekan tombol cancel a.1. Pengirim membatalkan memilih file stego	
Assumptions	Penerima belum memilih atau ingin merubah file stego

Tabel 3. 18 Skenario Use Case Extracting

Identifikasi	
Use Case Name	Extract File
Actor	Penerima
Use Case Description	Penerima mengekstrak file child yang disisipkan ke dalam file parent.
Trigger	Penerima memilih file stego yang akan diekstrak dan menekan tombol proses.
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Penerima berada di form extract file	
2. Penerima menekan tombol proses	
3. Penerima mendapatkan file hasil ekstrak	
Extension	
a. Jika file stego tidak ditemukan a.1. Penerima memilih file stego a.2. Penerima menekan tombol proses kembali. b. Jika tidak ditemukan parameter file yang disembunyikan b.1. File tidak mengandung file yang disisipkan b.2. Penerima mendapatkan pesan gagal extract c. Jika gagal extract file c.1. Penerima mendapatkan pesan gagal extract c.2. Penerima menekan tombol proses kembali	
Assumptions	Penerima sudah memiliki file stego.

Tabel 3. 19 Skenario Use Case Verification

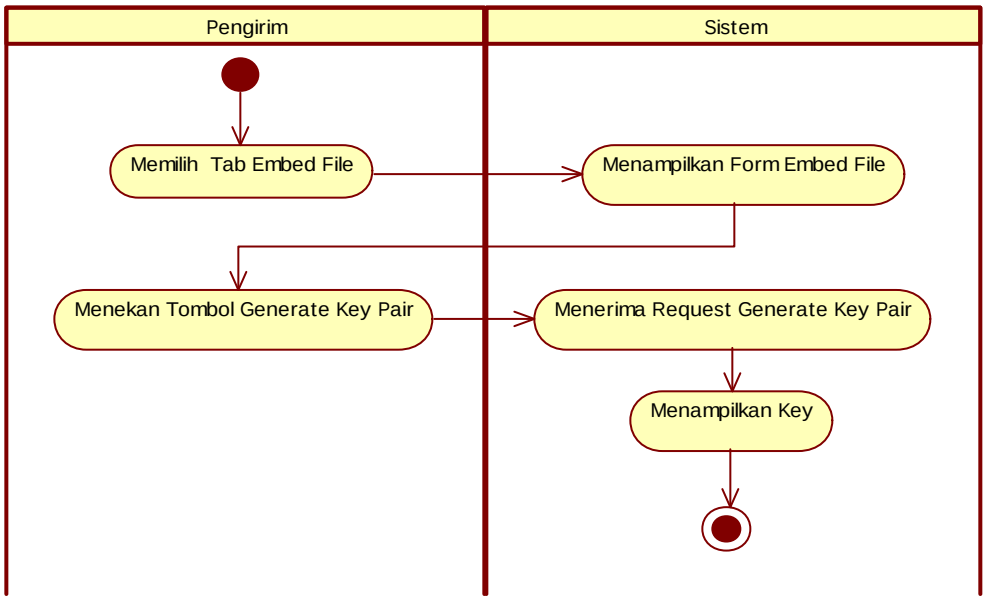
Identifikasi	
Use Case Name	Verify
Actor	Penerima
Use Case Description	Penerima akan memverifikasi keaslian file child yang disisipkan
Trigger	File child berhasil di ekstrak dan penerima menekan tombol verify
Trigger Type	☐ Internal ☑ External
Main Success Scenario	
1. Penerima sudah mendapatkan file child yang diekstrak dari file stego	
2. Penerima menekan tombol verify	
3. Penerima mendapatkan hasil verifikasi	
Extension	
a. Jika file child tidak ditemukan a.1. Penerima mengekstrak file stego terlebih dahulu a.2. Penerima menekan tombol verify kembali b. Jika signature tidak valid b.1. File child telah ada perubahan atau file tersebut bukan berasal dari orang yang mengirim	
Assumptions	Penerima berhasil mengekstrak file child

3. 1. 5. 3. ***Activity Diagram***

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

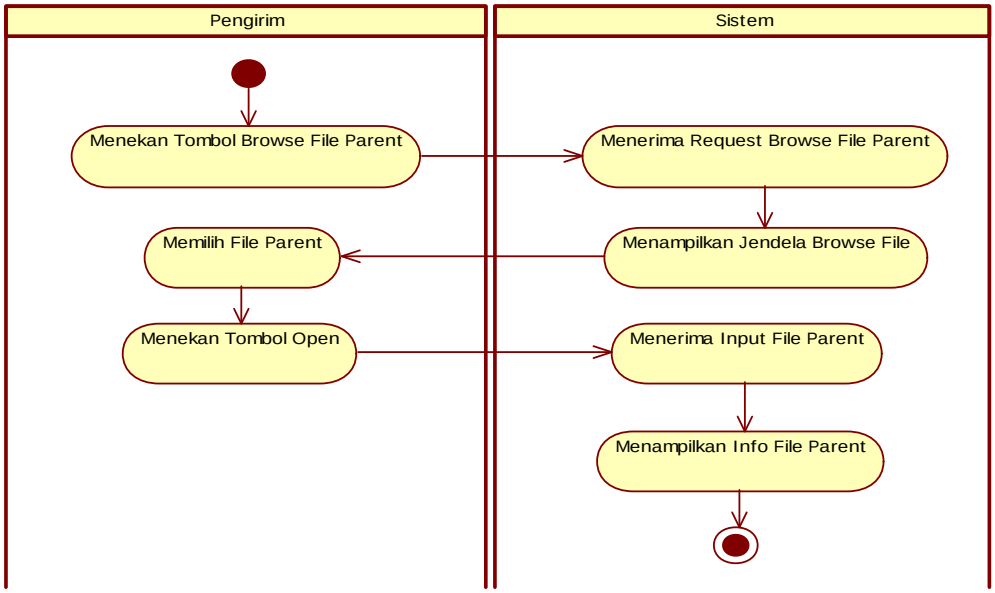
Berikut *activity diagram* yang ada pada perangkat lunak steganografi dan *signature* ini :

1. Generate Key



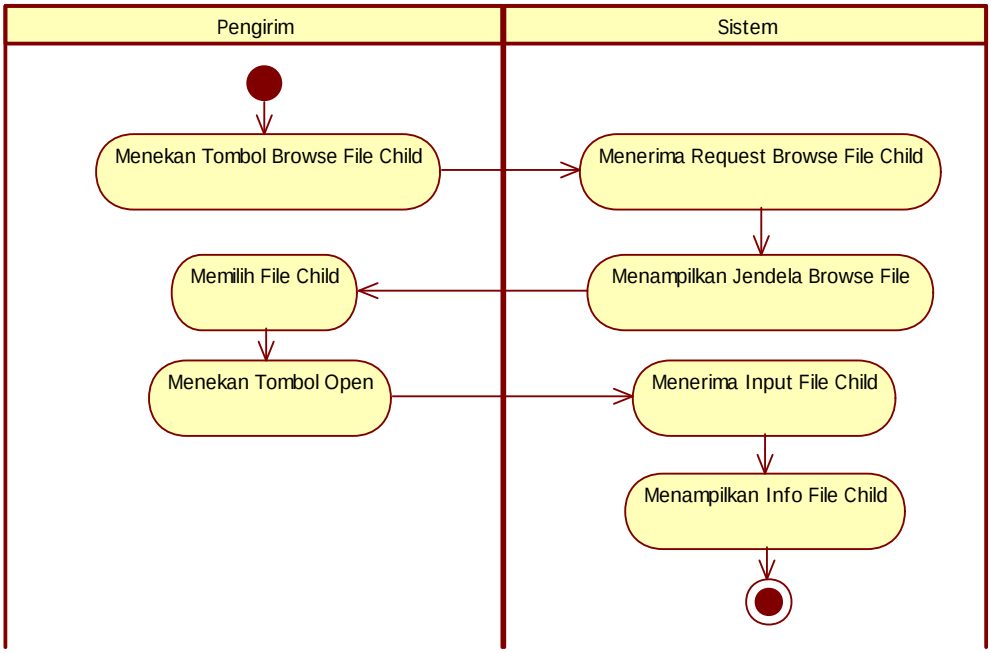
Gambar 3. 10 ***Activity Diagram Generate Key***

2. Browse File Parent



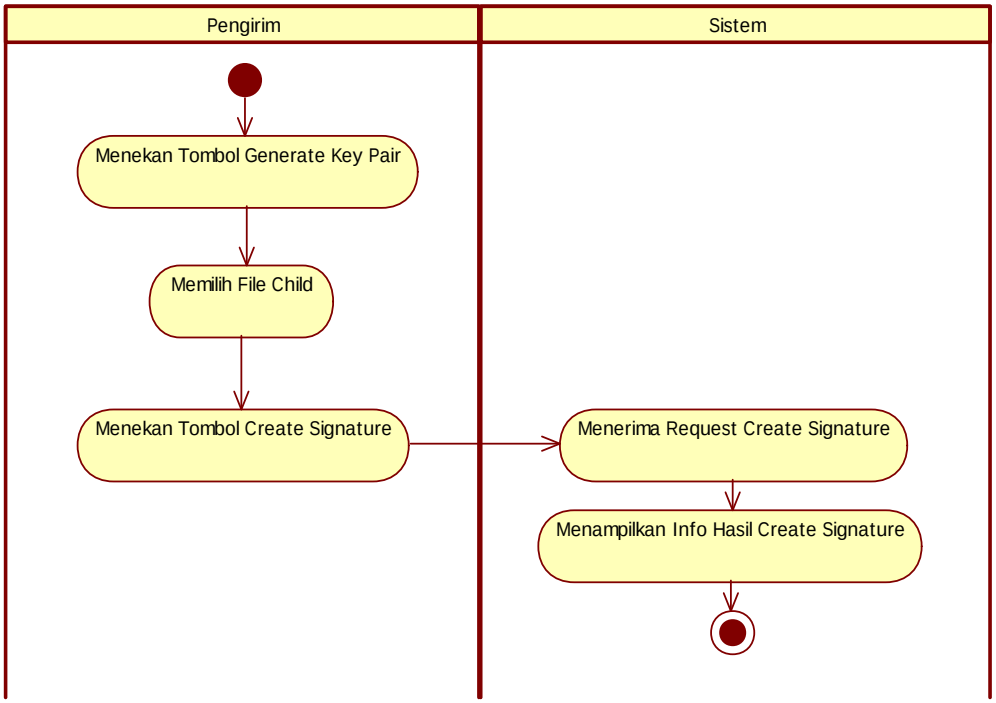
Gambar 3. 11 ***Activity Diagram Browse File Parent***

3. Browse File Child



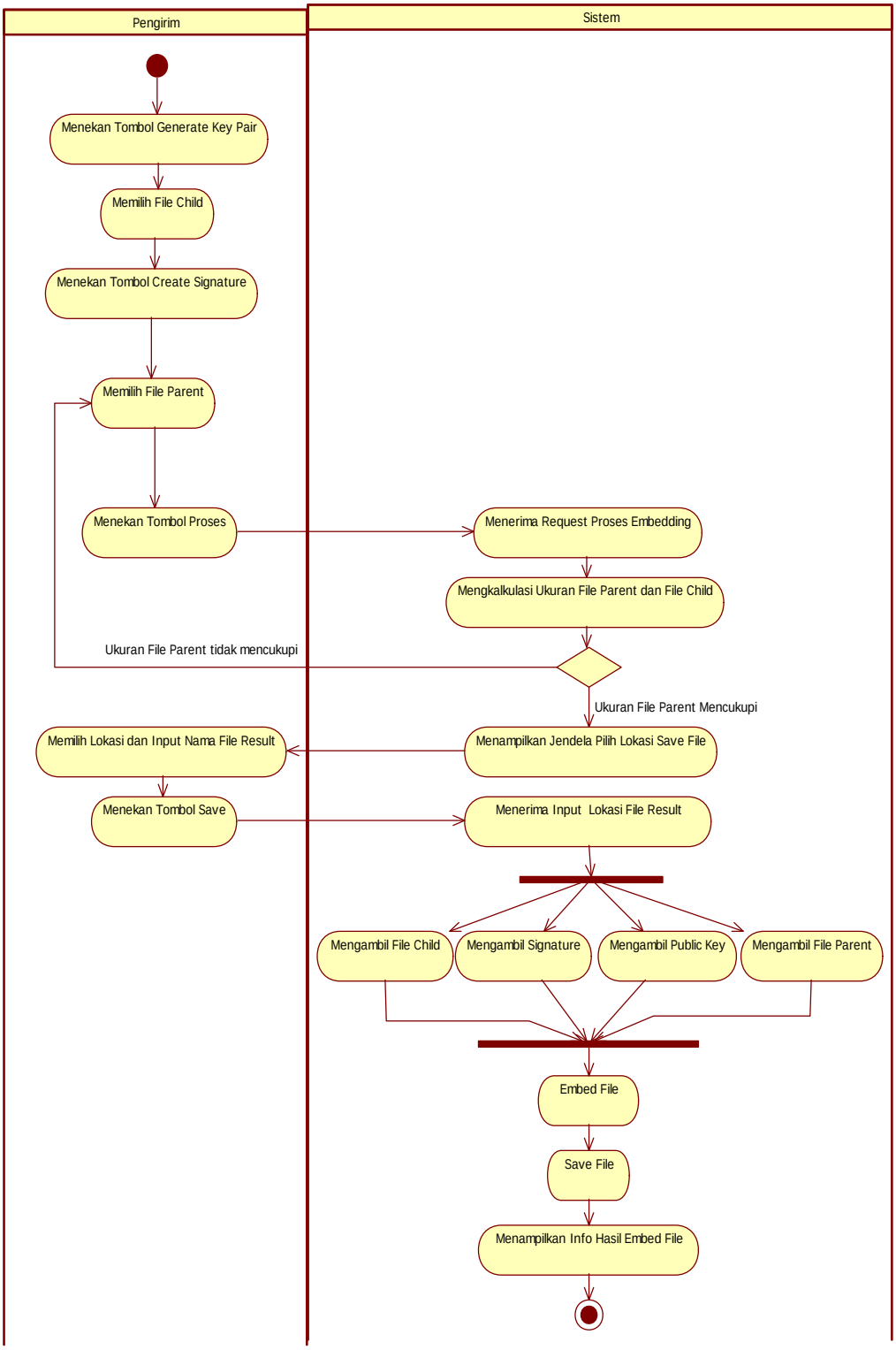
Gambar 3. 12 ***Activity Diagram Browse File Child***

4. Create Signature



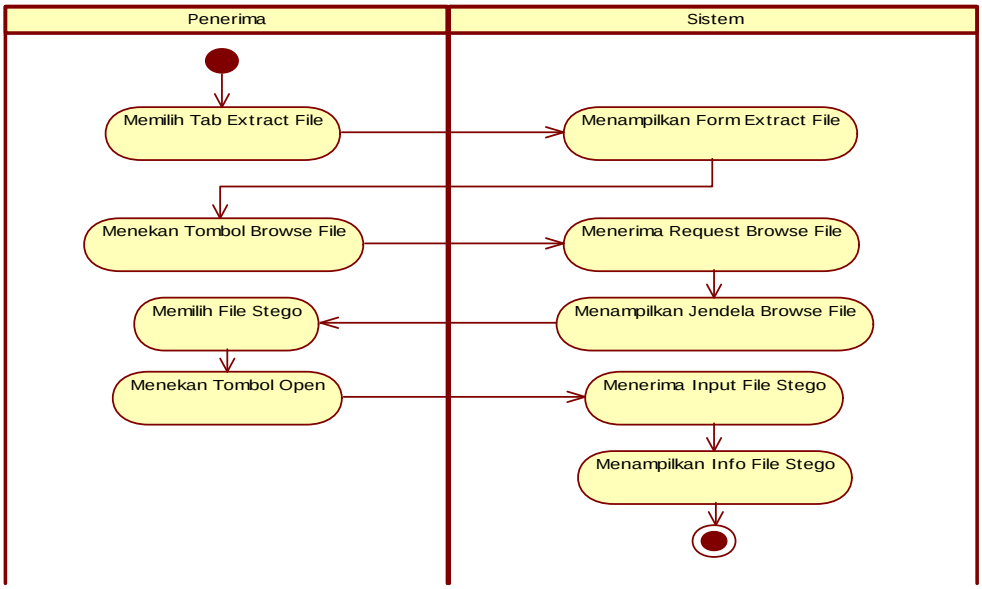
Gambar 3. 13 ***Activity Diagram Create Signature***

5. Embed File



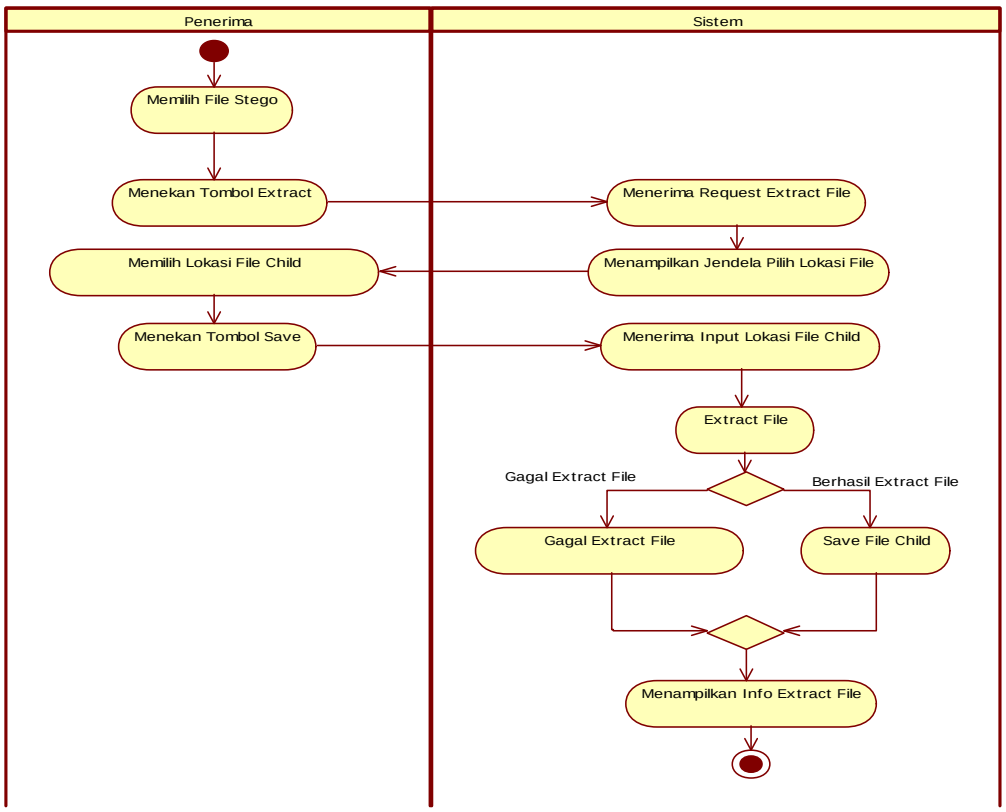
Gambar 3. 14 ***Activity Diagram Embed File***

6. Browse File Stego



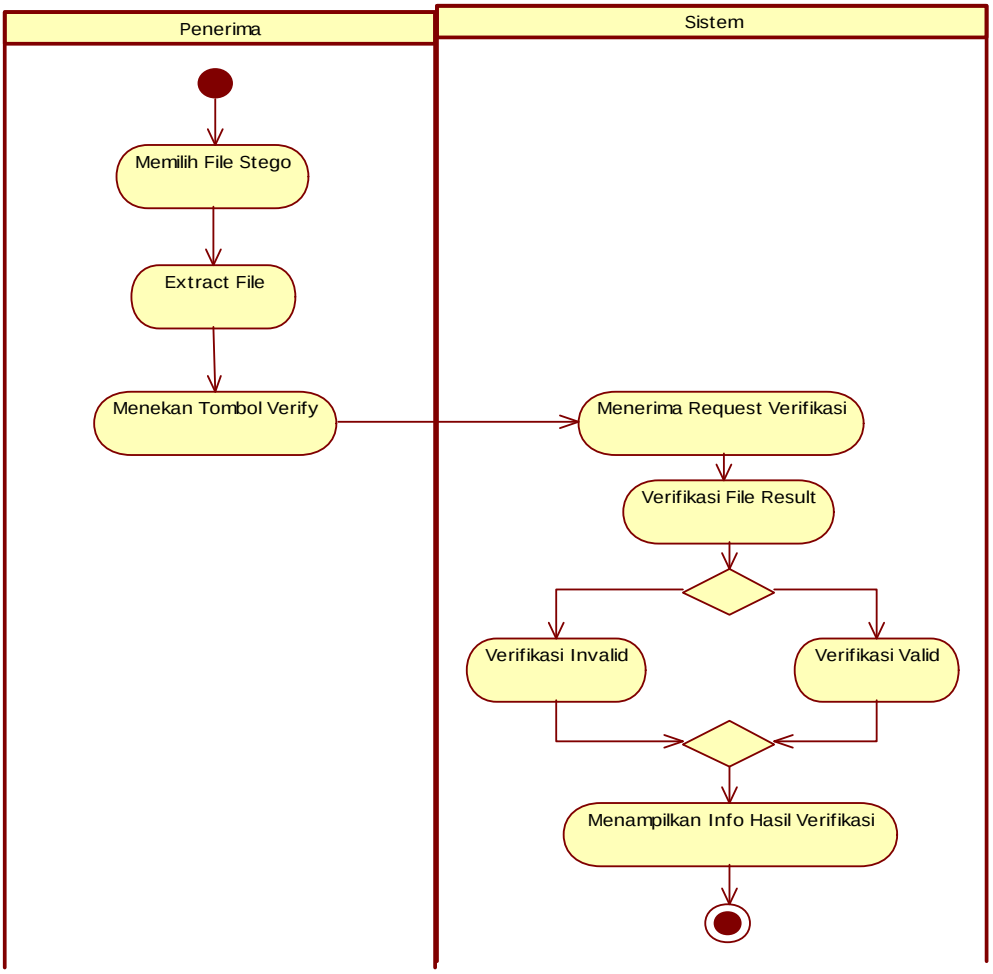
Gambar 3. 15 ***Activity Diagram Browse File Stego***

7. Extract File



Gambar 3. 16 ***Activity Diagram Extract File***

8. Verify



Gambar 3. 17 ***Activity Diagram Verify***

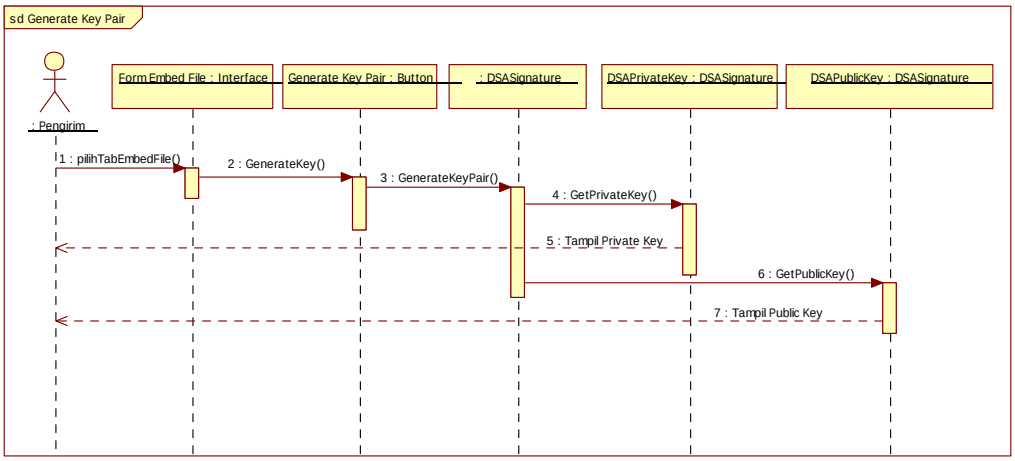
3. 1. 5. 4. ***Sequence Diagram***

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output*

tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

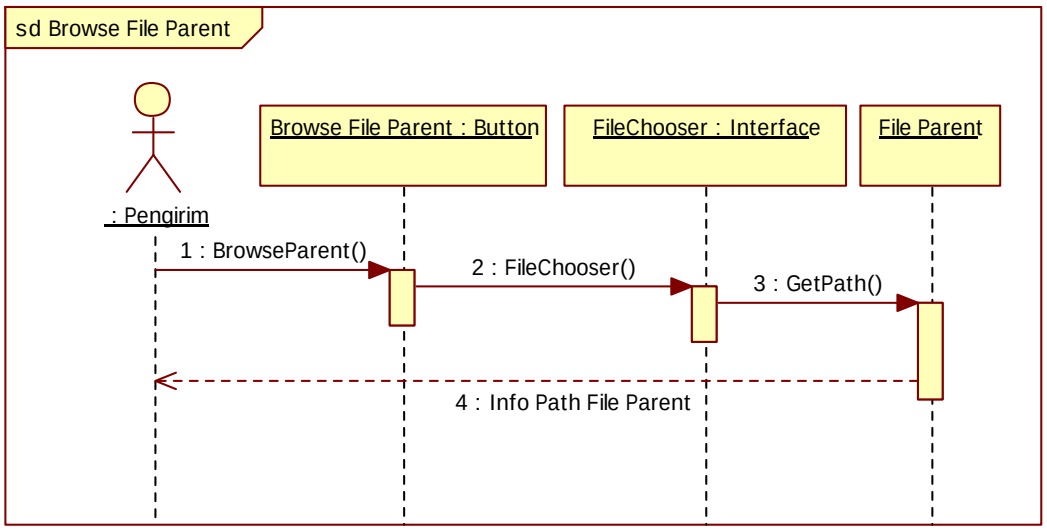
Sequence diagram pada perangkat lunak steganografi dan *digital signature* ini terbagi menjadi beberapa bagian, yaitu :

1. Generate Key



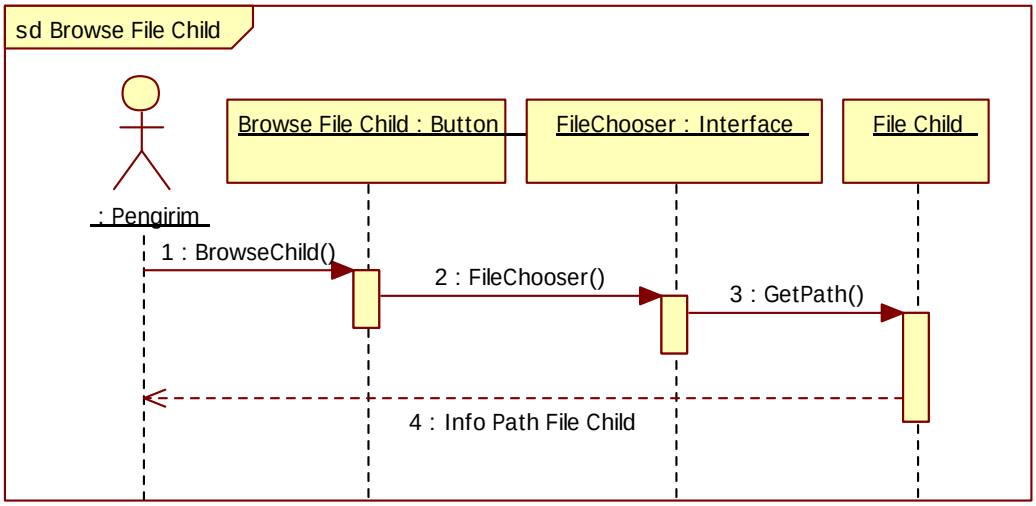
Gambar 3. 18 *Sequence Diagram Generate Key*

2. Browse File Parent



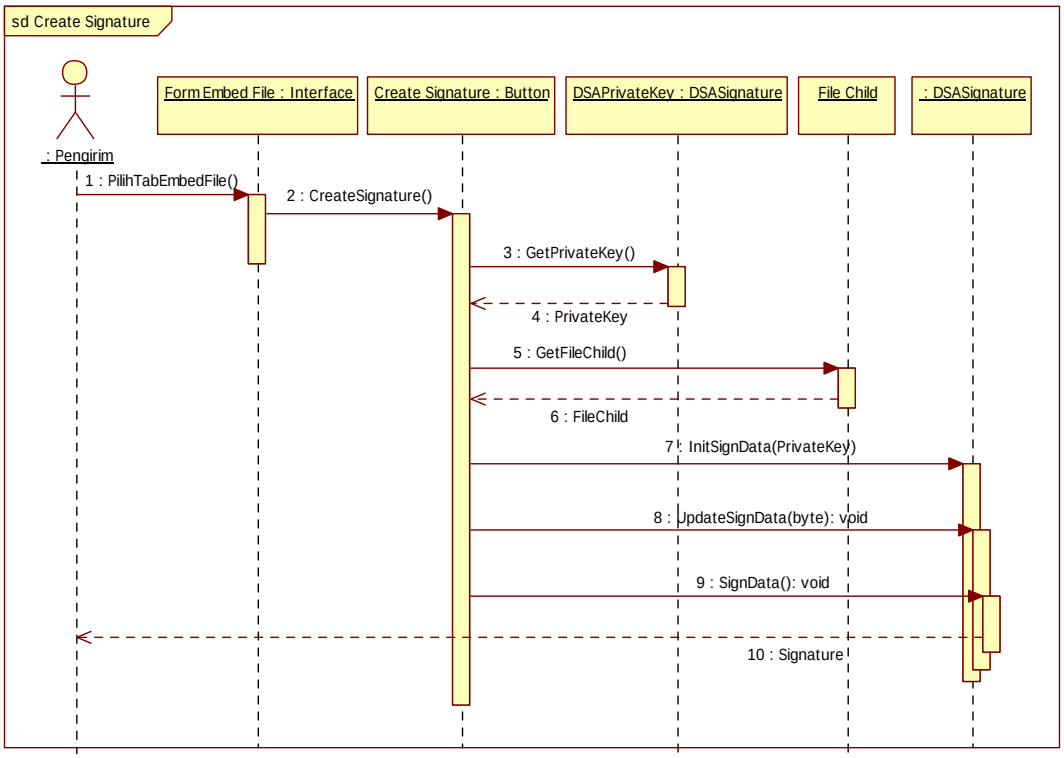
Gambar 3. 19 *Sequence Diagram Browse File Parent*

3. Browse File Child



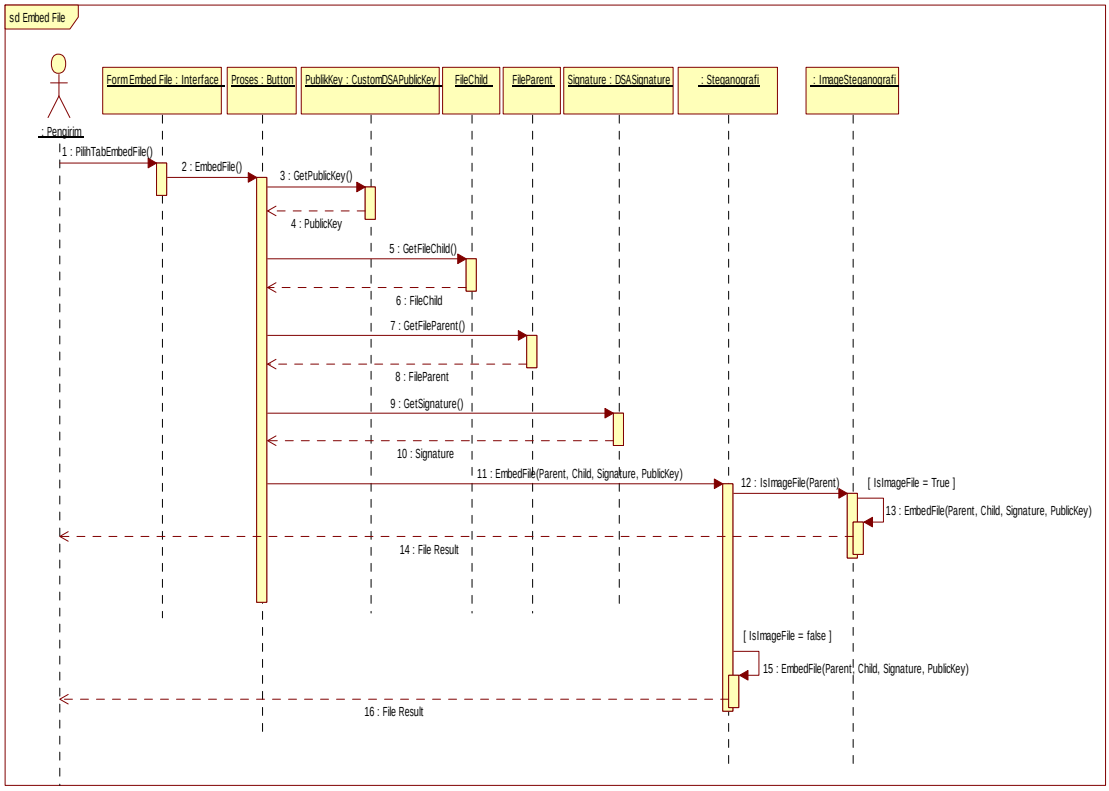
Gambar 3. 20 ***Sequence Diagram Browse File Child***

4. Create Signature



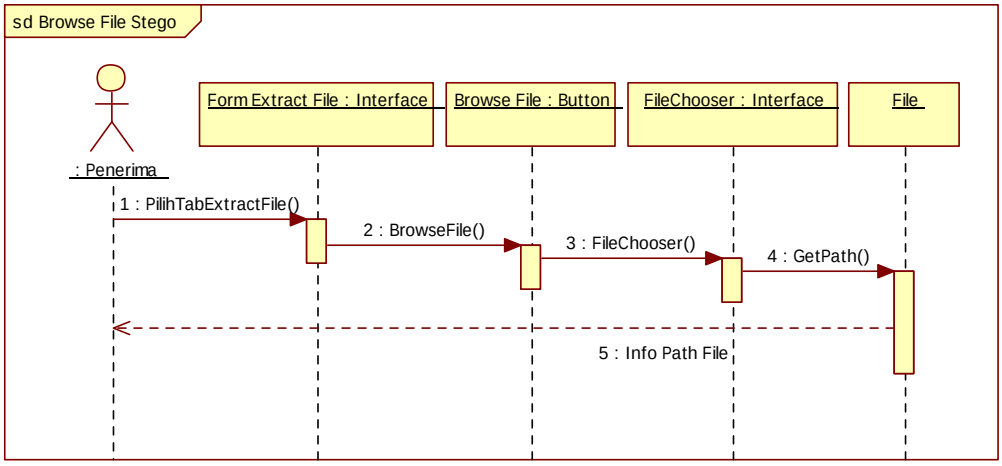
Gambar 3. 21 ***Sequence Diagram Create Signature***

5. Embed File



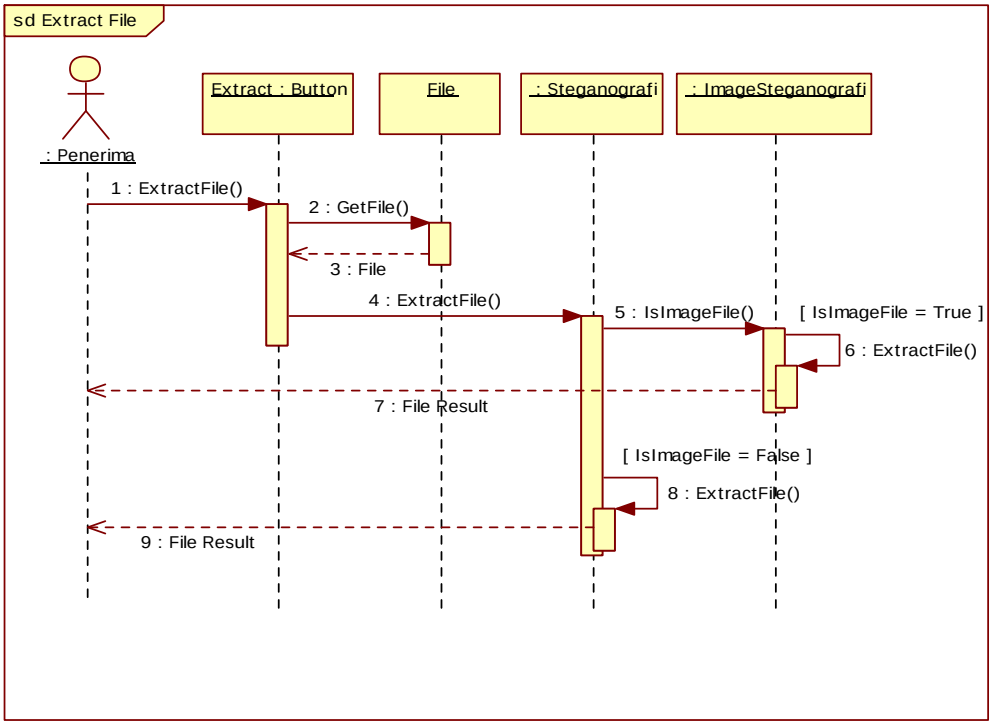
Gambar 3. 22 *Sequence Diagram Embed File*

6. Browse File Stego



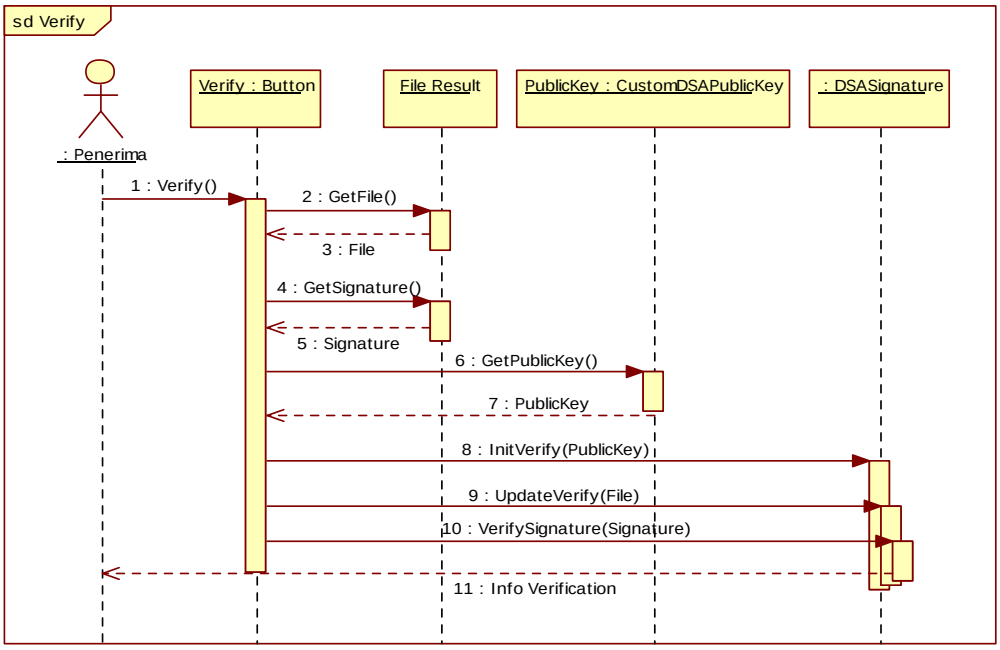
Gambar 3. 23 *Sequence Diagram Browse File Stego*

7. Extract File



Gambar 3. 24 *Sequence Diagram Extract File*

8. Verification

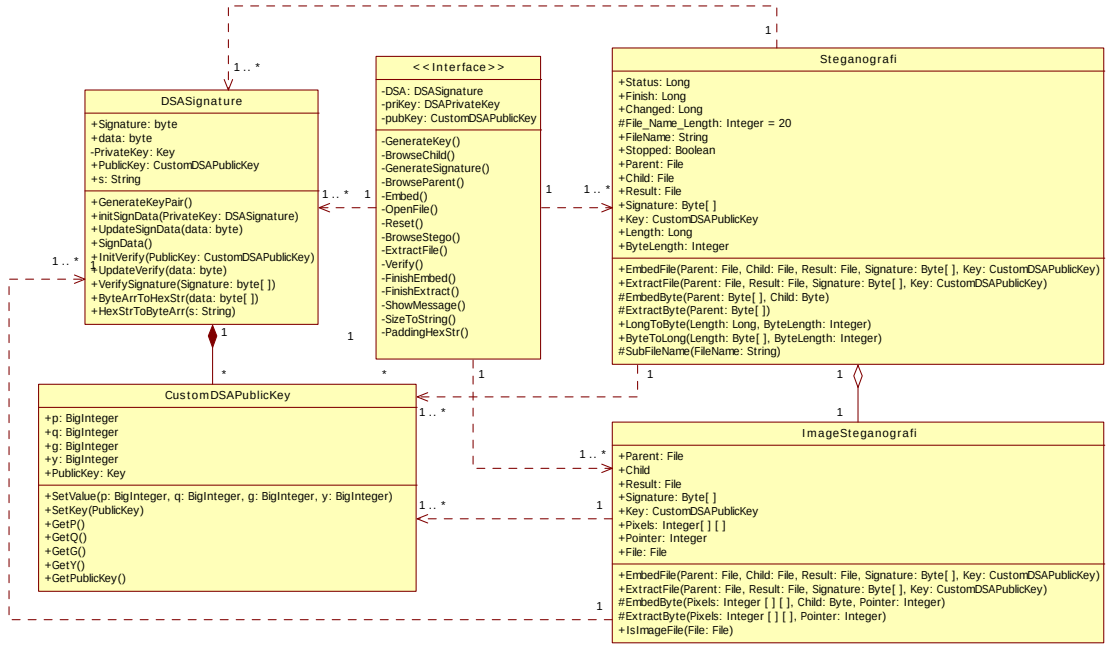


Gambar 3. 25 *Sequence Diagram Verification*

3. 1. 5. 5. ***Class Diagram***

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Berikut *class diagram* yang terbentuk pada perangkat lunak steganografi dan *digital signature*.



Gambar 3. 26 *Class Diagram*

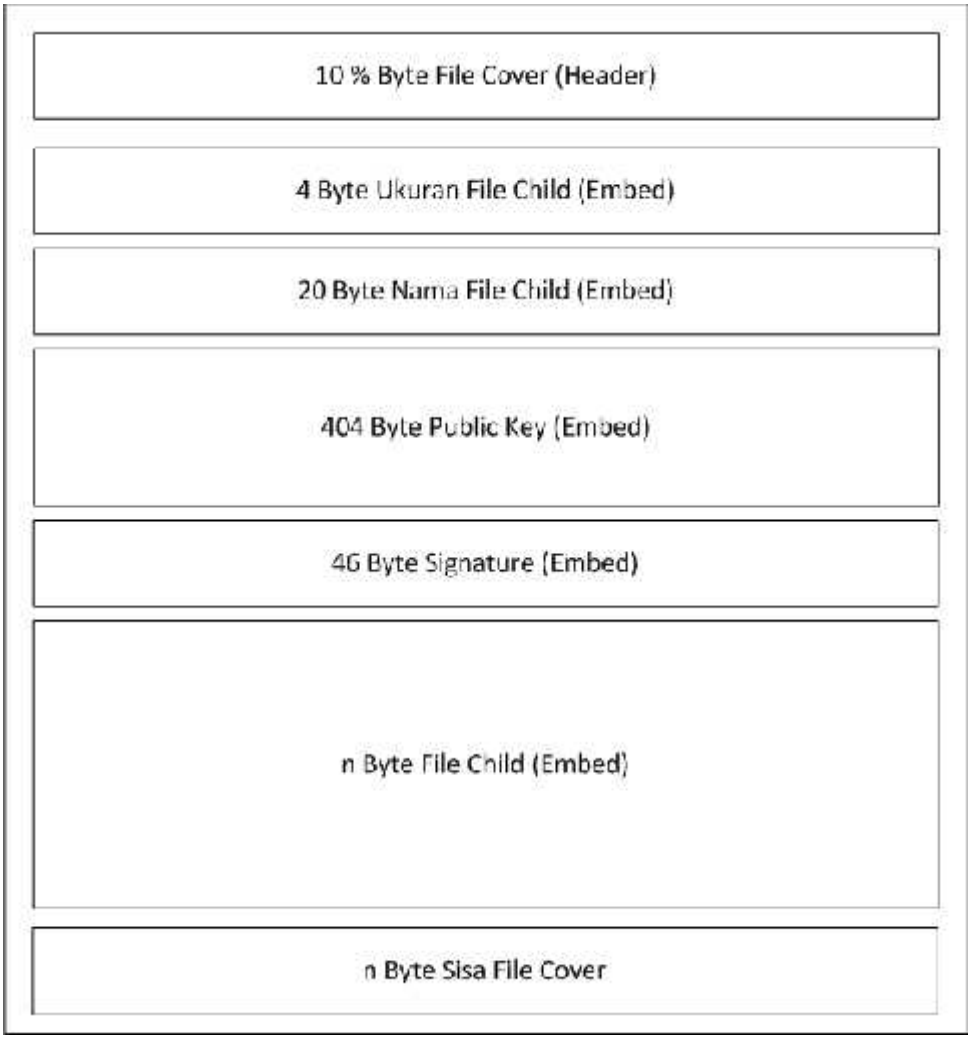
3. 2. Perancangan Sistem

Perancangan merupakan tahap lanjutan dari analisis sistem dimana pada perancangan sistem digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean kedalam suatu bahasa pemograman. Dalam perancangan sistem tidak lepas dari hasil analisis, karena dari hasil analisis sistem baru dapat dibuat suatu perancangan sistem.

Desain umum yang akan diaplikasikan bertujuan untuk memberikan gambaran secara umum kepada penggunaan tentang sistem yang akan dibangun.

3. 2. 1. Perancangan Struktur File

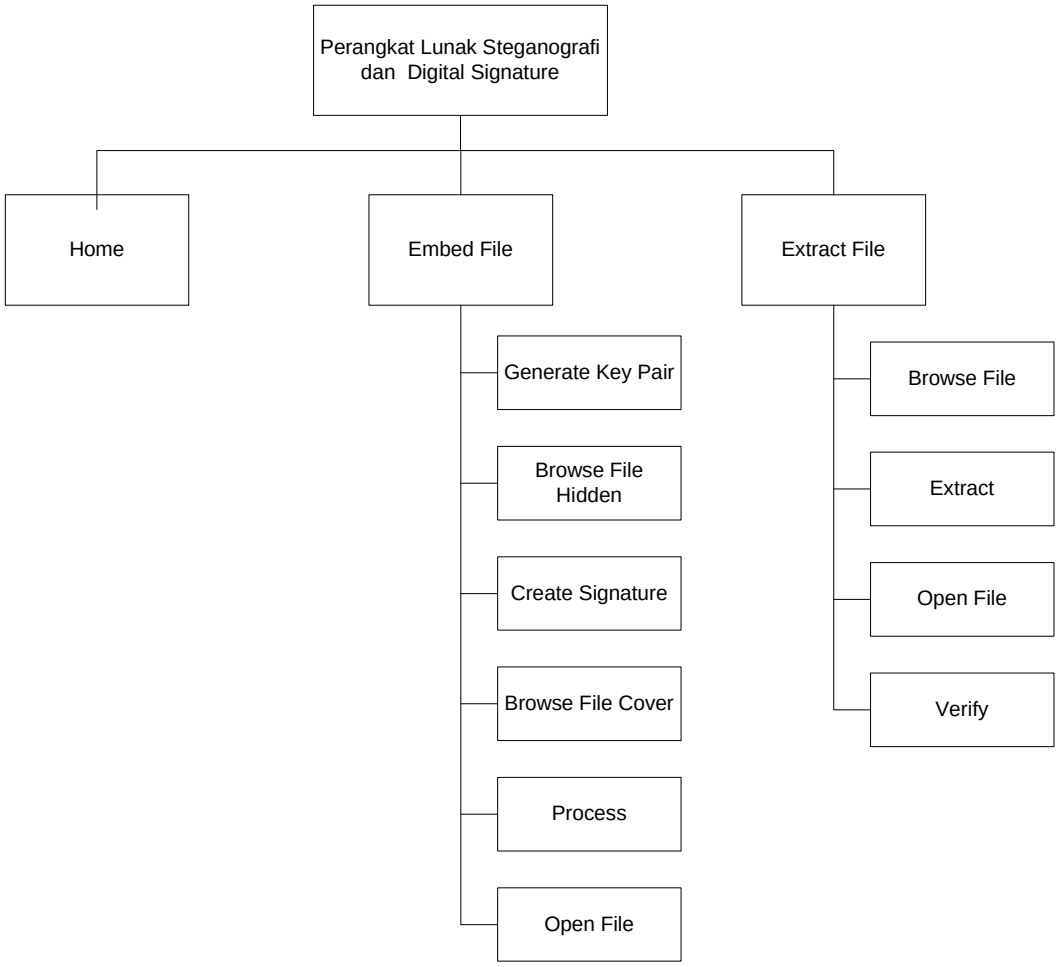
File hasil embed memiliki struktur yang berbeda dengan file aslinya, karena file hasil embed telah disisipkan file dan parameter-parameter yang dibutuhkan pada saat proses ekstraksi nanti. Berikut adalah struktur file yang tercipta setelah proses embed dilakukan.



Gambar 3. 27 Perancangan Struktur File

3. 2. 2. Perancangan Struktur Menu

Struktur Menu pada perangkat lunak steganografi dan *digital signature* yang akan dibangun, digambarkan dengan struktur sebagai berikut :



Gambar 3. 28 Struktur Menu

3. 2. 3. Perancangan Antarmuka Perangkat Lunak

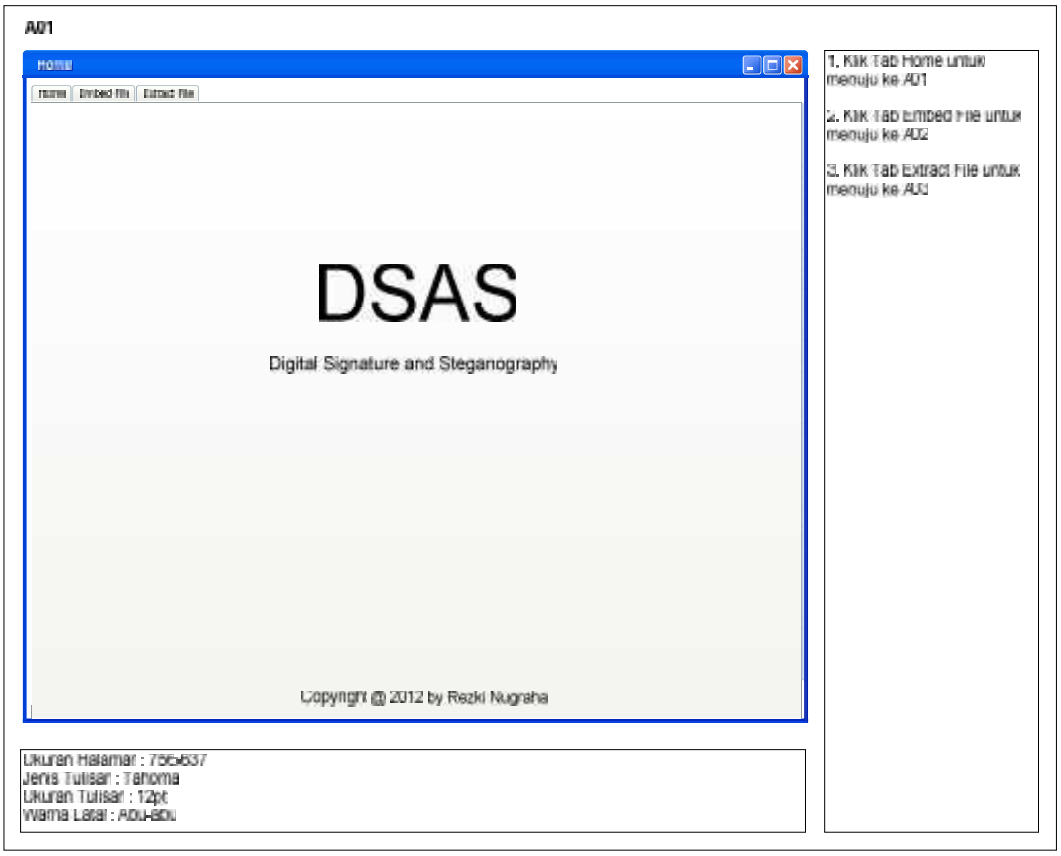
Antarmuka merupakan tampilan dari suatu perangkat lunak yang berperan sebagai media komunikasi yang digunakan sebagai sarana antara perangkat lunak dan pengguna. Perancangan ini merupakan sebuah penggambaran, perencanaan dan pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan utuh.

Sistem yang dibangun diharapkan menyediakan suatu tampilan antarmuka yang mudah dipahami dan digunakan oleh pengguna.

Berikut ini adalah tampilan antarmuka yang digunakan untuk perangkat lunak steganografi dan *digital signature* :

3. 2. 3. 1. Perancangan Antarmuka *Home*

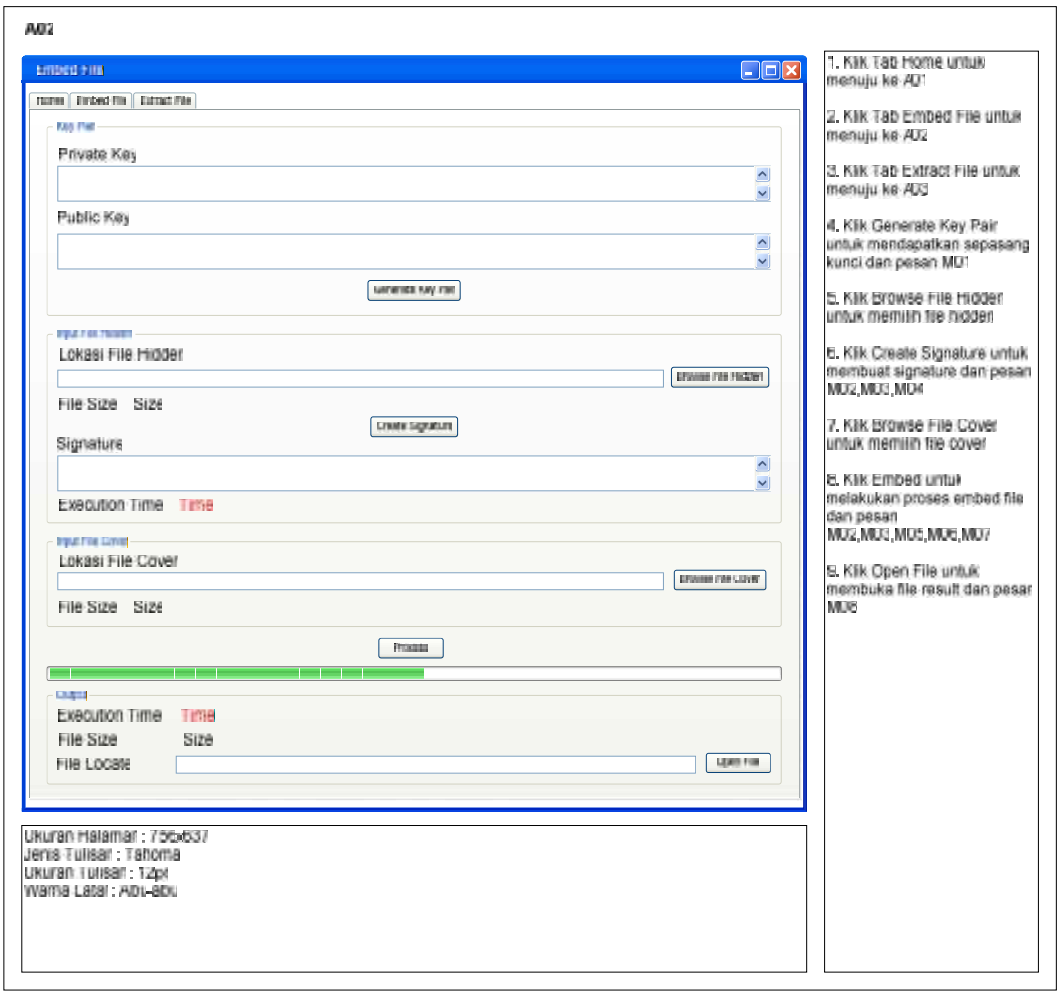
Berikut ini dapat dilihat perancangan antarmuka halaman *Home*. Halaman *Home* merupakan halaman pertama yang akan muncul ketika perangkat lunak dieksekusi oleh pengguna. Perancangan antarmuka halaman *Home* digambarkan seperti pada gambar 3.29.



Gambar 3. 29 Perancangan antarmuka halaman *Home*

3. 2. 3. 2. Perancangan Antarmuka *Embed File*

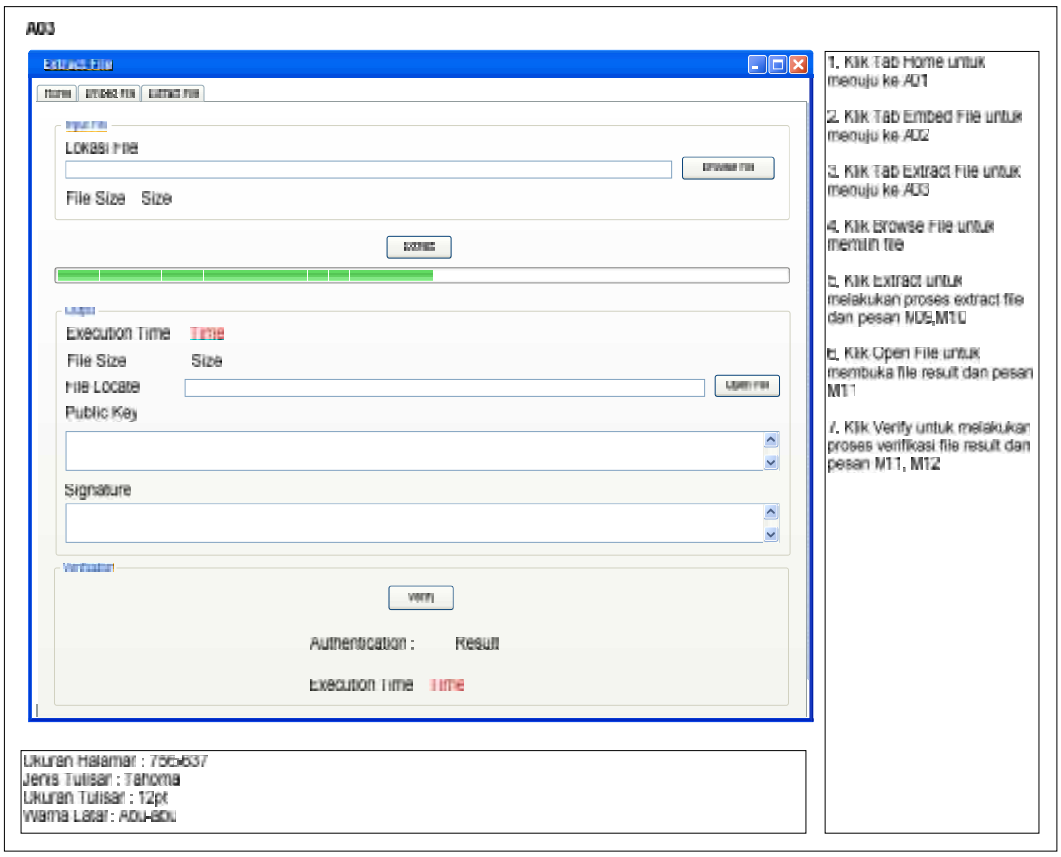
Berikut ini dapat dilihat perancangan antarmuka halaman *Embed File*. Perancangan antarmuka halaman *Embed File* digambarkan seperti pada Gambar 3.30.



Gambar 3. 30 Perancangan Antarmuka Halaman *Embed File*

3. 2. 3. 3. Perancangan Antarmuka *Extract File*

Berikut ini adalah gambar perancangan antarmuka halaman *Extract File*. Halaman *Extract File* akan muncul apabila pengguna memilih tab *Extract File*. Perancangan ditampilkan pada Gambar 3.31.



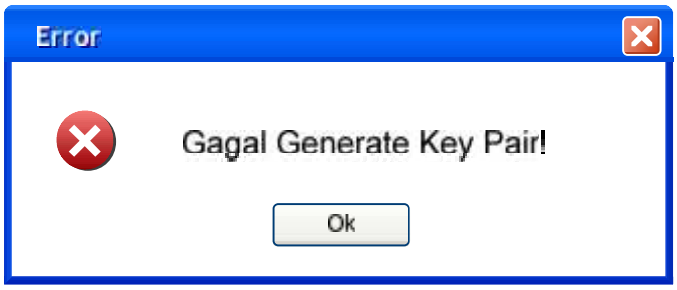
Gambar 3. 31 Perancangan Antarmuka Halaman *Extract File*

3. 2. 4. Perancangan Pesan

Pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi untuk menyampaikan notifikasi dan informasi kepada pengguna agar perangkat lunak lebih interaktif.

Berikut perancangan pesan yang ada pada perangkat lunak steganografi dan *digital signature* :

1) Rancangan pesan yang akan muncul pada saat gagal *Generate Key Pair*.



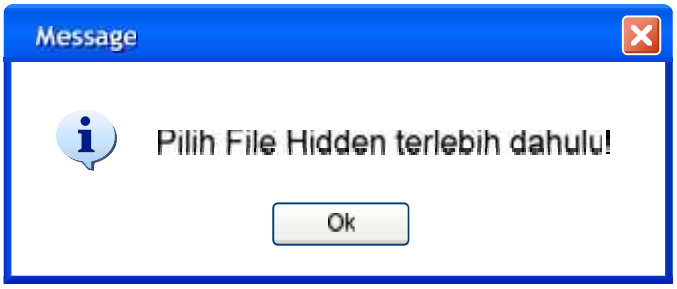
Gambar 3. 32 Pesan M01

2) Rancangan pesan yang akan muncul pada saat akan membuat *signature* dan melakukan proses *Embed File* tetapi belum memiliki sepasang kunci.



Gambar 3. 33 Pesan M02

3) Rancangan pesan yang akan muncul pada saat akan membuat *signature* dan melakukan proses *embed file* tetapi belum memilih *file hidden*.



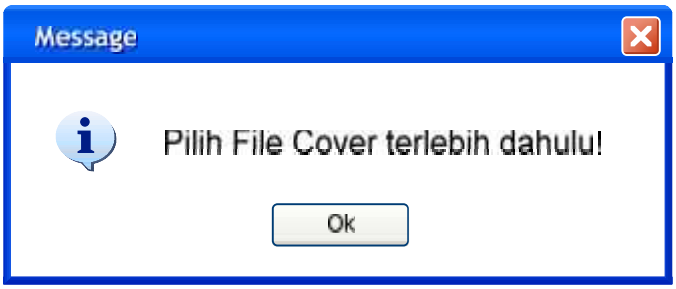
Gambar 3. 34 Pesan M03

4) Rancangan pesan yang akan muncul pada saat gagal membuat *signature*.



Gambar 3. 35 Pesan M04

5) Rancangan pesan yang akan muncul pada saat akan melakukan proses *embed file* tetapi belum memilih *file cover*.



Gambar 3. 36 Pesan M05

6) Rancangan pesan yang akan muncul pada saat akan melakukan proses *embed file* tetapi belum membuat *signature*.



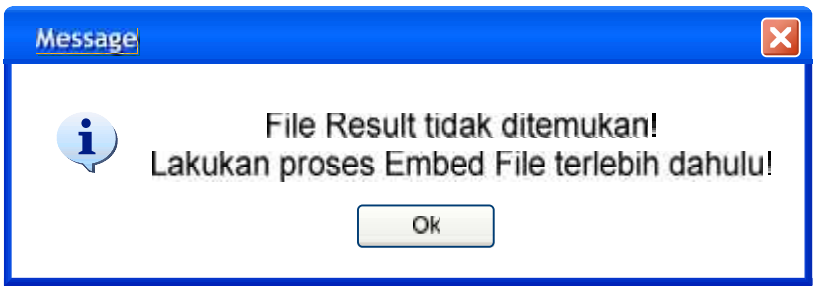
Gambar 3. 37 Pesan M06

7) Rancangan pesan yang akan muncul pada saat gagal melakukan proses *embed file*.



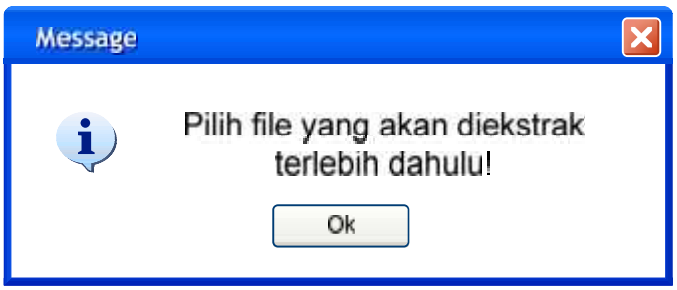
Gambar 3. 38 Pesan M07

8) Rancangan pesan yang akan muncul pada saat akan membuka *file result* tetapi *file result* tidak ditemukan karena belum melakukan proses *embed file*.



Gambar 3. 39 Pesan M08

9) Rancangan pesan yang akan muncul pada saat akan melakukan proses *extract file* tetapi belum memilih *file* yang akan di-*extract.*



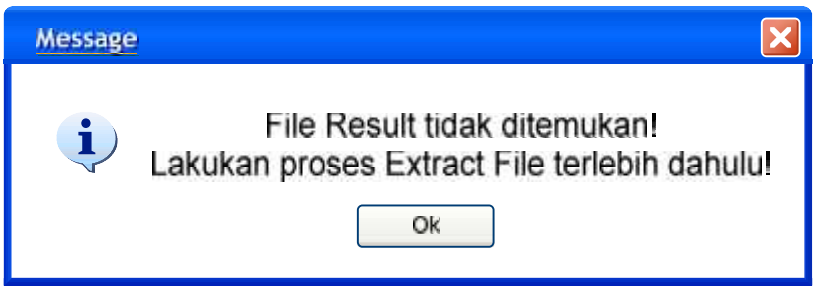
Gambar 3. 40 Pesan M09

10) Rancangan pesan yang akan muncul pada saat gagal melakukan *extract file.*



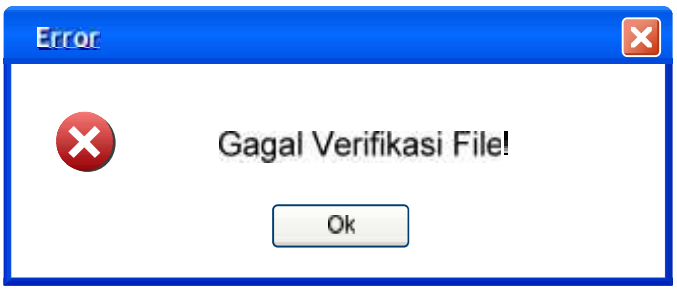
Gambar 3. 41 Pesan M10

11) Rancangan pesan yang akan muncul pada saat akan melakukan proses *verify* dan membuka *file* tetapi *file result* tidak ditemukan karena belum melakukan proses *extract file*.



Gambar 3. 42 Pesan M11

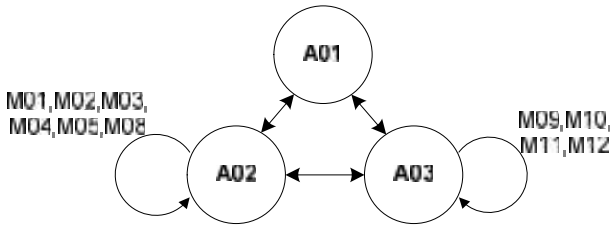
12) Rancangan pesan yang akan muncul pada saat gagal melakukan proses *verify*.



Gambar 3. 43 Pesan M12

3. 2. 5. Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam sebuah program. Jaringan semantik perangkat lunak steganografi dan *digital signature* dapat dilihat pada gambar 3.44.



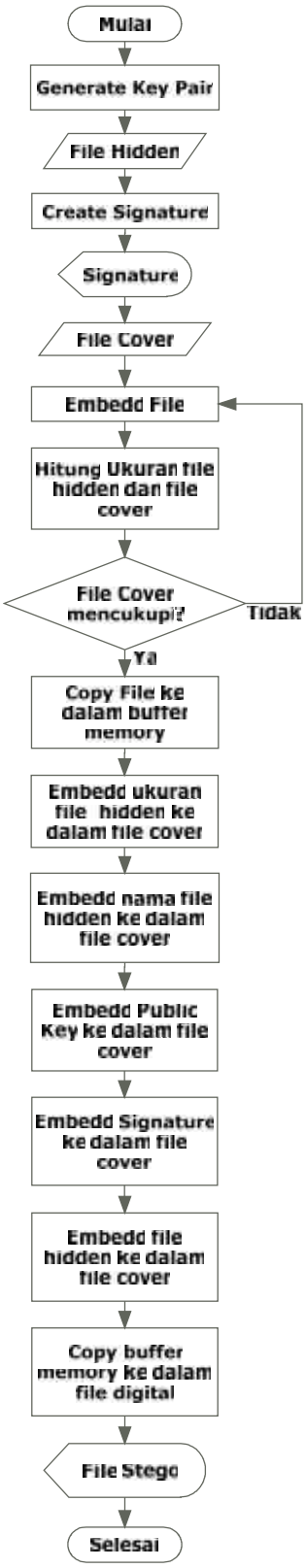
Gambar 3. 44 Jaringan Semantik

3. 2. 6. Perancangan *Method*

Langkah terakhir dalam perancangan yaitu merancang *method*. Perancangan *method* digunakan sebagai algoritma dasar yang akan diimplementasikan ke dalam sistem. Perancangan *method* untuk perangkat lunak steganografi dan *digital signature* adalah sebagai berikut :

3. 2. 6. 1 *Method Embed File*

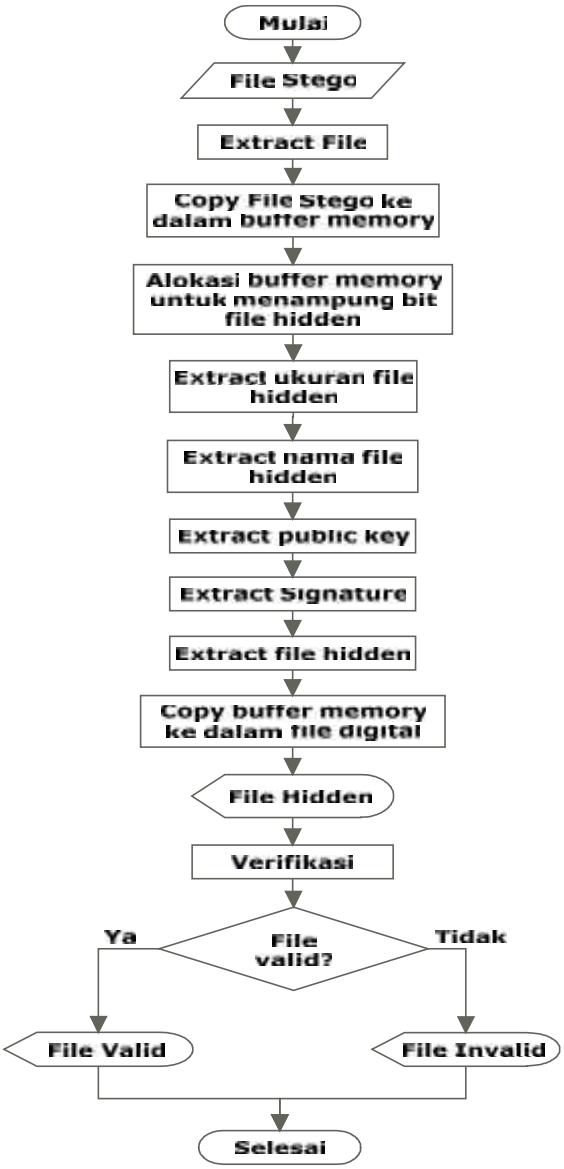
Method embed file menjelaskan langkah-langkah untuk proses *embed file* pada perangkat lunak. Dimulai dengan membuat sepasang kunci, memilih *file child* dan *file parent,* membuat *signature,* menyisipkan *file*, hingga menghasilkan *file result* yang sudah disisipi *file child.* Berikut rancangan *method Embed File* dapat dilihat pada gambar 3.45.



Gambar 3. 45 ***Method Embed File***

3. 2. 6. 2 *Method Extract File*

Method extract file menjelaskan langkah-langkah untuk proses ekstraksi file dan verifikasi file. Dimulai dengan memilih *file* yang akan diekstrak, melakukan proses *extract file* terhadap *file* tersebut, sehingga didapatkan *file* yang disembunyikan, kemudian dilakukan proses verifikasi terhadap *file* tersebut. Berikut rancangan *method extract file* dapat dilihat pada gambar 3.46.



Gambar 3. 46 ***Method Extract File***