# APLIKASI KRIPTOGRAFI UNTUK PENGAMANAN E-DOKUMEN DENGAN METODE *HYBRID* : BIOMETRIK TANDATANGAN DAN DSA (*DIGITAL SIGNATURE ALGORITHM*)

**Tesis**

**Untuk Memenuhi Sebagian Persyaratan**

**Mencapai Derajat Sarjana S-2**

**Program Studi Magister Sistem Informasi**


**Oleh :**

**ANA WAHYUNI**

**J4F009005**


**PROGRAM PASCA SARJANA**

**UNIVERSITAS DIPONEGORO**

**SEMARANG**

**2011**

**TESIS**

**APLIKASI KRIPTOGRAFI UNTUK PENGAMANAN E-DOKUMEN DENGAN METODE HYBRID : BIOMETRIK TANDATANGAN DAN DSA (*DIGITAL SIGNATURE ALGORITHM*)**

Telah dipertahankan di depan Dewan Penguji pada tanggal 25 Juli 2011

| Penguji I | Penguji II |
|---|---|
| Prof. Dr. Ir. Eko Sediyono, M.Kom | Drs. Eko Adi Sarwoko, M. Kom |
| NIDN. 0628096101 | NIP.196511071992031003 |

| Pembimbing I | Pembimbing II |
|---|---|
| Drs. Bayu Surarso, M.Sc,Ph.D | Aris Sugiharto, S.Si, M.Kom |
| NIP. 196311051988031001 | NIP. 197111081997021004 |

Mengetahui
Ketua Program Studi
Magister Sistem Informasi

Prof. Drs. Mustafid, M, Eng, Ph. D
NIP. 195505281980031002

**DAFTAR ISI**

## DAFTAR TABEL

## DAFTAR GAMBAR

## DAFTAR LAMPIRAN

## ABSTRACT

Exchange of computer-based documents such as e-mail message or document in an e-mail on the Internet has been widely used as a commercial transaction. To ensure e-document is still intact / authentic to the party verifier in transit on the network insecure one way to provide digital signatures on e-documents. One method to create a digital signature is a method of biometric signatures in combination with the DSA (Digital Signature Algorithm). The purpose of this research is to create cryptographic applications with a hybrid method: Biometric signatures and DSA (Digital Signature Algorithm) as one solution to the problem of key management and meet the needs non singular of signer. Biometric signature is used offline. In this study as an input (key generator) is offline signature of one or more users generate one or more digital signatures to a single e-document. Furthermore, e-documents, digital signatures and public key is transmitted over the Internet via e-mail on hand verifier. Then the verifier verifies whether the result is a valid means of e-documents are still authentic / intact and the sender is the actual signer of the e-document. In contrast, if the results are not valid means of e-document is not authentic / intact and signer or not the actual

sender of the e-document.



Key words: signer, verifier, off-line signature, digital signatures, Biometric signatures, DSA (Digital Signature Algorithm).


## ABSTRAK

Pertukaran dokumen berbasis komputer seperti pesan e-mail atau dokumen dalam pesan e-mail di internet sudah luas digunakan sebagai transaksi komersiil. Untuk memastikan e-dokumen masih utuh/ otentik sampai di pihak *verifier* dalam perjalanan di jaringan *insecure* salah satunya dengan cara memberi tandatangan *digital* pada e-dokumen. Salah satu metode untuk membuat tandatangan *digital* adalah metode biometrik tandatangan yang dikombinasikan dengan DSA (*Digital Signature Algorithm*). Tujuan dari penelitian ini adalah membuat aplikasi kriptografi dengan metode *hybrid :* Biometrik tandatangan dan DSA (*Digital Signature Algorithm*) sebagai salah satu solusi pada masalah manajemen kunci dan memenuhi kebutuhan ketidaktunggalan *signer*. Biometrik yang digunakan adalah tandatangan *offline.* Pada penelitian ini sebagai masukan (generator kunci) adalah tandatangan *offline* satu atau lebih pengguna menghasilkan satu atau lebih tandatangan *digital* untuk satu e-dokumen. Selanjutnya e-dokumen, tandatangan *digital* dan kunci publik ditransmisikan lewat internet via *e-mail* pada fihak *verifier*. Kemudian pihak *verifier* memverifikasi apakah hasilnya valid artinya e-dokumen tersebut masih otentik/ utuh dan pengirim adalah *signer* sebenarnya dari e-dokumen tersebut. Sebaliknya jika hasilnya tidak valid artinya e-dokumen tersebut sudah tidak otentik/ utuh dan atau pengirim bukanlah *signer* sebenarnya dari e-dokumen tersebut.

Kata kunci : *signer, verifier,* tandatangan *off-line,* tandatangan *digital,* Biometrik tandatangan, DSA (*Digital Signature Algorithm*).

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Pertukaran dokumen berbasis komputer seperti pesan *e-mail* atau dokumen dalam pesan *e-mail* di internet sudah luas digunakan sebagai transaksi komersiil. Dokumen sering berisi informasi penting seperti kontrak resmi, transaksi keuangan, *record* penjualan dan lain-lain. Seringkali hal terpenting yang disertakan pada dokumen yaitu tandatangan (*handwritten) signer.* Verifikasi suatu dokumen dalam hal otorisasi (pemberi kuasa) dilakukan terhadap tandatangan seseorang atau beberapa *signer* yang menandatangani dokumen tersebut. Dokumen sangat penting dalam transaksi komersil lewat internet misalnya pada *e-mail*. Dokumen berbasis komputer disebut dokumen elektronik (e-dokumen) atau dokumen *digital*. Untuk menjamin bahwa e-dokumen yang diterima masih utuh/ otentik artinya sama dengan e-dokumen yang dikirim dan *signer* adalah penandatangan sebenarnya dari e-dokumen tersebut, salah satunya dengan memberi tandatangan *digital.*

Tandatangan *digital* adalah suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan/ *signer.* (Munir, 2005) Namun algoritma kriptografi untuk membuat tandatangan *digital* misalnya DSA (*Digital Signature Algorithm*), RSA (Rivest, Shamir, Adleman) atau ECDSA (*Elliptic Curve Digital Signature Algorithm*) hanya menghasilkan satu tandatangan *digital* untuk satu e-dokumen. Hal ini tidak sesuai dengan konsep tandatangan *digital* yang bergantung pada pengirim/ *signer* dimana *signer* lebih dari satu. Sehingga

diperlukan konsep baru mengenai tandatangan *digital* yang dapat berfungsi sebagai otorisasi suatu e-dokumen sama halnya otorisasi tandatangan (*handwritten)* beberapa *signer* pada dokumen fisik. Salah satu metode yang dapat memenuhi kebutuhan tandatangan *digital* oleh lebih dari satu *signer* adalah dengan biometrik tandatangan *offline.* Tandatangan *offline* adalah tandatangan pada dokumen fisik yang didigitasi oleh *scanner* (Najmul, 2006).

Biometrik tandatangan menunjukkan identifikasi otomatis dari seseorang berdasarkan karakter behavioral. Metode ini mengidentifikasi lebih baik daripada metode tradisional yang melibatkan *password* dan PIN. Biometrik tandatangan dapat digunakan untuk membangkitkan kunci privat dengan menghitung *hash* dari kode string biometrik tandatangan dengan fungsi *hash* SHA-1. Keluaran dari SHA-1 adalah bilangan *integer* 160 bit sesuai dengan panjang kunci privat x pada DSA (*Digital Signature Algorithm).* Sehingga algoritma DSA dapat digabungkan (*hybrid*) dengan biometrik tandatangan (Feng, H; & Chong Wa,C. 2002).

DSA (*Digital Signature Algorithm)* merupakan salah satu kriptografi kunci publik yang digunakan untuk otentikasi, pengamanan data dan perangkat anti sangkal. Pada DSA dibutuhkan program khusus untuk membangkitkan kunci dan masalah yang timbul adalah kepercayaan pengguna pada program tersebut. *Digital Signature Standart* yang dipublikasikan oleh *Federal Information Processing Standard* pada FIPS PUB 186-3, kunci privat digunakan untuk jangka waktu tertentu dan dapat diperpanjang selama pembangkitan tandatangan *digital* menggunakan kunci privat tersebut. Demikian juga berlaku untuk kunci publik yang dapat digunakan terus-menerus selama pasangannya yaitu kunci

privat digunakan untuk membangkitkan tandatangan *digital.* Demikian juga parameter DSA dapat digunakan bersama pada sekelompok pengguna dan bersifat publik. Parameter DSA bernilai tertentu (tetap) dan dapat tetap dipakai atau diperpanjang untuk beberapa periode waktu. Algoritma DSA dirancang untuk menjaga dari lawan *(attacker)* yang diasumsikan tidak tahu kunci privat *signer* yang digunakan untuk membangkitkan tandatangan *digital*. Menurut peneliti, penggunaan parameter, kunci publik dan privat yang tetap untuk suatu waktu tertentu dan diperpanjang untuk periode waktu tertentu, merupakan celah ketidakamanan penggunaan algoritma DSA, karena pihak *attacker* mempunyai kesempatan dan waktu seiring dengan kecepatan *processor* yang semakin bertambah. Pihak *attacker* dapat memecahkan kunci privat sebagai pasangan kunci publik yang digunakan untuk membuat tandatangan *digital.* Jika hal ini terjadi maka *attacker* dapat menyamar sebagai *signer* sah (pemegang kunci privat) dan mengubah e-dokumen yang sah sekaligus membuat tandatangan *digital*-nya untuk dikirimkan pada pihak *verifier*. Pada proses verifikasi di pihak *verifier,* akan didapat hasil verifikasi "valid" karena hasil dekripsi tandatangan *digital* sama dengan nilai *massage hash* e-dokumen walaupun berasal dari *attacker*. Sehingga fungsi tandatangan *digital* sebagai otentifikasi e-dokumen dan *signer* sah menjadi tidak berguna. Solusi dari masalah ini yaitu dengan menggunakan parameter, kunci publik dan kunci privat yang dinamis yaitu bernilai berbeda untuk tiap proses pembuatan tandatangan *digital*. Jadi sangat perlu bahwa setiap kunci diubah jauh sebelum ia dapat ditemukan dengan cara *exhaustive search* (Munir, 2006). Sehingga perlu dibangun aplikasi kriptografi

dengan metode DSA (*Digital Signature Algorithm*) yang dapat membangkitkan kunci secara dinamis walaupun dengan masukan yang sama. Hal ini menjadi salah satu solusi dalam hal manajemen kunci.

Selain persoalan manajemen kunci, pada DSA hanya menghasilkan satu nilai tandatangan *digital* pada satu e-dokumen. Pada kenyataannya seringkali *signer* tidak hanya satu orang pada satu e-dokumen. Jika *signer* lebih dari satu maka diperlukan tandatangan *digital* lebih dari satu untuk satu e-dokumen tersebut. Hal ini dapat diatasi dengan penggunaan biometrik khususnya tandatangan (*handwritten*) masing-masing *signer*. Pada *paper* Pawan dan Siyal (2001) tandatangan (*handwritten*) seseorang disebut biometrik tandatangan.

Biometrik tandatangan dapat digunakan dalam proses menurunkan kunci privat untuk menandatangani e-dokumen. Kunci privat dibangkitkan secara dinamis dari salah satu sampel biometrik tandatangan. Pembangkitan dinamis kunci privat membuktikan kemudahan penandatanganan e-dokumen sebagaimana dapat menandatangani e-dokumen kapanpun dimanapun tanpa membawa *disk* atau *smart card*.

Penggunaan biometrik tandatangan masing-masing *signer* dapat digunakan untuk membuat tandatangan *digital* yang mengijinkan lebih dari satu *signer* pada satu e-dokumen. Sedangkan tandatangan *digital* pada DSA dengan masukan biometrik tandatangan *signer* dapat menghasilkan pasangan kunci secara dinamis dan mengijinkan lebih dari satu *signer*. Sehingga perlu dibangun suatu aplikasi untuk pengamanan e-dokumen dengan metode *hybrid* atau penggabungan dari biometrik tandatangan dan DSA (*Digital Signature*

*Algorithm*) yang dapat membangkitkan kunci secara dinamis walaupun dengan masukan yang sama dan memenuhi kebutuhan adanya *signer* lebih dari satu pada satu e-dokumen.

## 1.2 Perumusan Masalah

Berdasarkan uraian pada latar belakang, permasalahan yang diteliti yaitu bagaimana membangun aplikasi untuk keamanan e-dokumen dengan metode *hybrid* : Biometrik tandatangan dan DSA (*Digital Signature Algorithm*) sebagai solusi dalam hal manajemen kunci dengan pembangkitan sepasang kunci secara dinamis walaupun dengan masukan yang sama dan memenuhi kebutuhan ketidaktunggalan *signer.*

## 1.3 Batasan Masalah

Batasan masalah pada penelitian ini yaitu :

1. Biometrik yang digunakan adalah tandatangan *offline.*
2. Fungsi Hash yang digunakan adalah SHA-1.
3. E-dokumen adalah data *digital* atau *message* dengan format biner berupa file dengan ekstensi php, mp3, pptx, ppt, html, php, pdf, txt, SYS, doc,docx, xls, xlsx, zip, exe, jpg dan bmp.
4. Pengamanan e-dokumen hanya meliputi : kerahasiaan yaitu dekripsi terhadap tandatangan *digital*, autentikasi *signer* dan integritas e-dokumen sebagai hasil verifikasi pada pihak *verifier.*
5. Tidak membahas aspek keamanan pada jalur komunikasi yaitu pada proses transmisi e-dokumen lewat internet via *email*, keamanan yang bersifat fisik dan keamanan yang berhubungan dengan *personal.*

6. Tidak ada proses validasi dari pihak penjamin, yaitu infrastruktur kunci publik manapun.

## 1.4 Keaslian Penelitian

Menurut Tzong-Chen Wu dan Ru-lan Su (1997) pada publikasi yang berjudul "*ID-based group-oriented cryptosystem and its digital signature scheme"*, tandatangan *digital* diterapkan pada kriptosistem berbasis identitas yang berorientasi pada suatu grup (*ID based group oriented cryptosystem*). S*igning* disebut grup. Grup berisi himpunan individu yang dibagi dalam otoritas tinggi, rendah dan ambang (diantara tinggi dan rendah) sesuai dari strategi otoritas yang didefinisikan sebelumnya. Beberapa *sender* dapat mengenkrip alamat pesan pada grup dengan hanya mengetahui identitas grup dan informasi publiknya tanpa perlu mengetahui strategi otoritas grup. Setiap individu dalam grup menyesuaikan pesan dan membangkitkan tanda tangan grup sesuai dengan strategi otorisasi yang diambil. *Verifier* dapat memvalidasi tandatangan grup dengan hanya mengetahui identitas grup dan informasi publiknya.

Menurut Vesna Hassler dan Helmut Biely (1999) pada publikasi yang berjudul "*Digital signature management"*, dalam manajemen tandatangan *digital* perlu dibangun infrastruktur untuk menerapkan tanda tangan digital berbasis *smart card* di bidang perbankan dan aplikasi perdagangan elektronik, salah satu persyaratan penting adalah untuk memenuhi semua standar internasional yang relevan untuk memastikan interoperabilitas dalam kasus pada jaringan juga untuk infrastruktur sertifikasi internasional. Salah satu bagian dari infrastruktur adalah direktori umum dalam bentuk server-LDAP di mana sertifikat X.509v3 dari

kunci publik tandatangan dapat diambil. Hal ini untuk menyediakan integritas, keaslian data dan *non-repudiation* dari semua informasi direktori penting juga untuk menerapkan fitur keamanan. Dengan cara ini pengguna dapat yakin bahwa pengguna sedang berbicara ke direktori terpercaya saat mengambil sertifikat dan informasi yang terkait dengan sertifikat. Dalam penelitian ini diberikan gambaran proyek dan beberapa wawasan dari spesifikasi sistem dan desain untuk infrastruktur kunci publik pada tandatangan *digital*.

Menurut Pawan, K.J.;& Siyal, M. Y (2001) pada publikasi berjudul "*Novel biometric digital signature for internet based applications*.", nomor identifikasi pribadi, *password*, kartu cerdas dan sertifikat *digital* adalah beberapa cara untuk otentifikasi pengguna di berbagai aplikasi perdagangan elektronik. Namun ini tidak berarti benar-benar mengenali seseorang, tetapi hanya pengetahuan dari data milik pengguna. Makalah ini memperkenalkan gagasan biometrik tandatangan, sebuah pendekatan baru untuk mengintegrasikan biometrik dengan infrastruktur kunci publik, menggunakan generasi tandatangan *digital* yang berbasis biometrik yang aman, berguna, cepat, nyaman, *non-invasif* dan benar mengidentifikasi pembuat dari transaksi. Juga ditunjukkan dua skema untuk penandatanganan biometrik menggunakan algoritma tandatangan *digital* RSA dan DSA, serta membahas masalah-masalah yang berhubungan dengan hal tersebut.

Menurut Munir R (2005) pada publikasi berjudul "Penggunaan Tandatangan Digital untuk Menjaga Integritas Berkas Perangkat Lunak", pendistribusian perangkat lunak melalui web di internet memiliki sejumlah

masalah. Salah satunya adalah masalah integritas berkas yang telah di-*download*. Integritas berkas perangkat lunak berkaitan dengan keaslian berkas program, keutuhan, dan keabsahan pengembang perangkat lunak. Berkas program dapat dimodifikasi oleh pihak ketiga (menjadi tidak asli) atau mengalami kerusakan (*corrupt*) oleh virus atau gangguan selama transmisi dari komputer *server* ke komputer *client* (menjadi tidak utuh). Selain itu, pengguna perangkat lunak perlu memastikan bahwa program yang ia *download* dibuat oleh pengembang program yang sah, dan bukan pengembang lain yang menyamar sebagai pengembang program yang asli. Masalah integritas berkas perangkat lunak ini dapat diselesaikan dengan menggunakan tandatangan *digital*. Tandatangan *digital* dibangkitkan dengan algoritma kriptografi kunci publik. Tandatangan *digital* bergantung pada isi berkas program dan kunci pengembang perangkat lunak. Melalui proses verifikasi, pengguna dapat membuktikan integritas berkas perangkat lunak yang ia *downlaod* dari situs web pengembang.

Menurut Hao Feng dan Chan Choong Wah (2002) pada publikasi yang berjudul "*Private key generation from on-line handwritten signature"*, sampel tandatangan manual secara *online* diperoleh dengan meng-*capture* proses penandatanganan pada tablet. Fitur dinamis diperoleh meliputi kecepatan, tekanan, sudut pena dan lain-lain yang sulit dipalsukan. Hal ini digunakan sebagai sampel tes biometrik. Kriptosistem BioPKI adalah sebagai solusi yang menunjukkan cara baru untuk menggabungkan dua teknologi, biometrik dan PKI. Kriptografi kunci publik yang dipilih adalah DSA daripada RSA, karena lebih

unggul dalam kecepatan pembuatan kunci dan penandatangan. Dari sampel tes kemudian memfilter secara random dan pemalsuan sederhana dapat ditentukan koding fitur untuk menentukan kode fitur untuk masing-masing fitur terdefinisi dan menghubungkan masing-masing kode ke dalam kode string. Pada tahap ini dilakukan pencatatan dan verifikasi tanda tangan. Kemudian pembangkit kunci privat didapat dari kode string tandatangan manual sebagai *input* dan digunakan untuk tandatangan *digital* pada e-dokumen. Pada paper ini hanya membahas pembangkitan kunci privat dari tandatangan *online* tidak sampai penerapan pada pembuatan tandatangan *digital*.

Kriptosistem biometrik tersebut unggul dalam keaslian tandatangan seseorang tapi ada kesulitan dalam verifikasinya, jika tanda tangan seseorang itu asli dan sangat mirip, namun sistem dapat menolak tandatangan tersebut karena tidak semua bit yang dihasilkan tepat benar atau lebih besar dari batas toleransi penerimaan. Juga ada kemungkinan tandatangan seseorang tidak unik. Hal ini akan menyulitkan dalam penandatanganan *digital* e-dokumen. Oleh karena itu pada penelitian ini digunakan tandatangan *offline* yaitu tandatangan manual (*handwritten*) pada dokumen cetak/ fisik yang didigitasi dengan alat pemindai (*scanner*) untuk membangkitkan kunci privat. Keaslian tandatangan manual seseorang merupakan tanggung jawab seorang staf atau suatu unit pada suatu lembaga yang diberi wewenang dalam pakta integritas. Sedangkan pada implementasinya tanpa melibatkan infrastruktur kunci publik (PKI/ *Public Key Infrastructure*) Indonesia ataupun asing, karena aplikasi kriptografi yang diteliti tidak sampai tahap penerapan pada sistem di PKI.

Pada penelitian ini dibangun aplikasi kriptografi untuk pengamanan e-dokumen dengan metode *hybrid* yaitu biometrik tandatangan dan DSA. Biometrik tandatangan yang digunakan yaitu tandatangan (*handwritten) offline* untuk membangkitkan parameter dan sepasang kunci secara dinamis. Tandatangan *offline* digunakan untuk mempermudah penandatanganan e-dokumen, karena tidak harus semua bit tepat untuk setiap tandatangan dan tandatangan seseorang tidak harus unik tetapi verifikasi hanya pada keabsahan tandatangan *digital*. Biometrik tandatangan *offline* yang digabung dengan DSA memenuhi kebutuhan satu e-dokumen dengan lebih dari satu *signer* sehingga menghasilkan tidak hanya satu tandatangan *digital*, tapi sebanyak *signer*-nya. Keaslian penelitian ini diberikan pada tabel 1.1, yaitu pada perbedaan mengenai tandatangan *digital* menurut *paper* dari Hao dan Chan (2002), publikasi FIPS pada FIPS PUB 186-3 (2009) dan pada penelitian ini.

## 1.5 Tujuan Penelitian

Tujuan yang dicapai dalam penelitian ini adalah membangun aplikasi kriptografi untuk pengamanan e-dokumen dengan metode *hybrid* : biometrik tandatangan dan DSA (*Digital Signature Algorithm)* sehingga menjadi solusi dalam hal manajemen kunci dengan pembangkitan parameter dan sepasang kunci secara dinamis walaupun dengan masukan yang sama dan memenuhi kebutuhan ketidaktunggalan *signer*. Pengamanan e-dokumen dijamin dengan hasil proses verifikasi.

**Tabel 1.1** *Komparasi dengan Penelitian Sebelumnya*

| Hao dan Chan (2002) | FIPS PUB 186-3 (2009) | Penelitian ini |
|---|---|---|
| 1.Dengan masukan | 1.Dengan masukan | 1.Dengan masukan |

| | | |
|---|---|---|
| tandatangan *online* dari satu *signer* menghasilkan satu pasang kunci untuk membangkitkan lebih dari satu tandatangan *digital*. Demikian pula parameter bernilai tetap dan digunakan pada jangka waktu tertentu dan dapat diperpanjang waktunya. Jadi sifat penggunaan sepasang kunci adalah beberapa kali pakai untuk jangka waktu tertentu dan dapat diperpanjang untuk tiap kali membuat tandatangan *digital*.<br>2. Dari satu e-dokumen, dapat diberi tandatangan *digital* hanya satu, karena hanya mengijinkan satu *signer*. | bilangan acak random atau *pseudorandom* menghasilkan parameter, kunci privat dan publik bernilai tetap dan digunakan pada jangka waktu tertentu dan dapat diperpanjang waktunya. Jadi sifat penggunaan sepasang kunci adalah beberapa kali pakai untuk jangka waktu tertentu dan dapat diperpanjang untuk tiap kali membuat tandatangan *digital*.<br>2. Dari satu e-dokumen, dapat diberi tandatangan *digital* hanya satu, karena hanya mengijinkan satu *signer*. | biometrik tandatangan *offline* untuk membangkitkan parameter dan sepasang kunci secara acak (dinamis) pada setiap saat pembuatan sepasang kunci dilakukan. Jadi sifat penggunaan sepasang kunci adalah sekali pakai atau setiap saat diperlukan dapat menggunakan parameter dan sepasang kunci yang berbeda walaupun dengan masukan yang sama atau kunci dengan beberapa kali pakai tergantung keputusan *signer*.<br>2. Dari satu e-dokumen, dapat diberi tandatangan *digital* lebih dari satu, karena mengijinkan lebih dari satu *signer*. |

### 1.6 Manfaat Penelitian

Sistem yang dibangun menggunakan biometrik tandatangan yang dikombinasikan dengan keuntungan dari penggunaan kriptografi kunci publik DSA yaitu integritas dan kepercayaan yang bermanfaat untuk menjaga keotentikan isi e-dokumen dan kepercayaan pada pihak penandatangan/ *signer* yang sebenarnya.

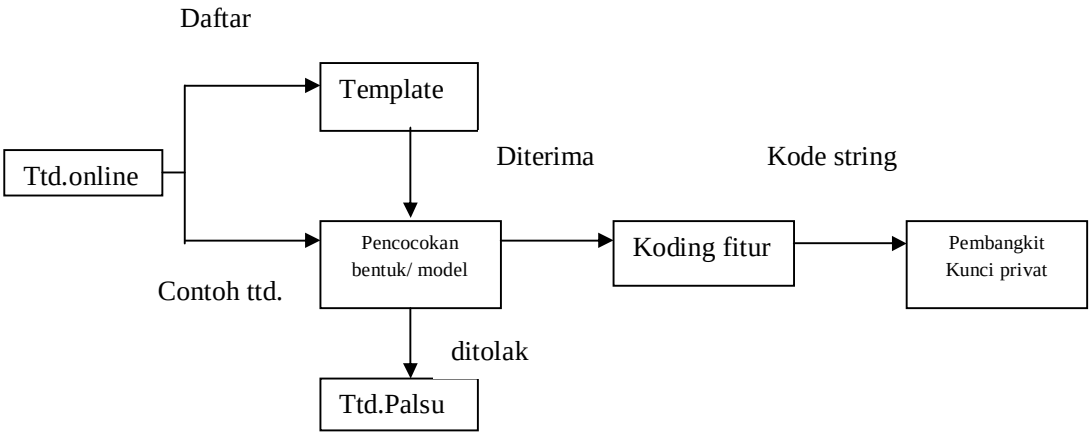
# BAB II

# TINJAUAN PUSTAKA

## 2.1 Tinjauan Pustaka

Menurut Tzong-Chen Wu dan Ru-lan Su (1997) pada publikasi yang berjudul *ID-based group-oriented cryptosystem and its digital signature scheme*, teknik penyandian dan tandatangan *digital* sangat penting untuk melindungi informasi sensitif atau konfidensiil dari tindakan ilegal yaitu pengungkapan/ pembongkaran, perusakan atau pemodifikasian. Pada aplikasinya pesan sering kali dialamatkan atau ditandatangani oleh grup/ sekelompok orang misalnya dewan direktur atau komite. Pada kasus ini *sender* akan mentransmisikan pesan rahasia pada grup. Kriptosistem berorientasi grup terdiri dari tahap inisialisai sistem, pembangkit informasi publik grup, pembangkit kunci personal individu, enkripsi dan dekripsi. Beberapa *sender* dapat mengenkrip alamat pesan pada grup dengan hanya mengetahui identitas grup dan informasi publiknya tanpa perlu mengetahui strategi otoritas grup. Setiap individu dalam grup menyesuaikan pesan dan membangkitkan tandatangan grup sesuai dengan strategi otorisasi yang diambil. *Verifier* dapat memvalidasi tandatangan grup dengan hanya mengetahui identitas grup dan informasi publiknya. Kriptosistem ini sangat sesuai digunakan pada penyiaran (*broadcasting*) atau komunikasi *point* ke *multipoint* yang menghemat biaya komunikasi.

Menurut Feng dan Choong Wah (2002), pada publikasi yang berjudul *Private key generation from on-line handwritten signature*, kriptosistem BioPKI merupakan solusi yang menunjukkan metode untuk menggabungkan dua teknologi, biometrik dan PKI. Gambar 2.1 menunjukkan diagram blok dari kriptosistem BioPKI.




**Gambar 2.1** *Diagram Blok Kriptosistem BioPKI*

Kriptosistem ini berisi tiga *stages*/ tingkat yaitu:

1. *shape matching*/ pencocokan bentuk (model)
2. *feature coding*/ koding fitur
3. *private key generation*/ pembangkit kunci privat

Pada *shape matching* meng-*input*-kan *shape*/ model dari sampel tandatangan sebagai fitur dan memfilter secara random juga pemalsuan sederhana. Hanya tanda tangan yang sesuai dengan toleransi kemiripan tertentu yang akan diproses pada *stage feature coding.* Pada *stage feature coding/* koding fitur untuk menentukan kode fitur masing-masing fitur terdefinisi dan menghubungkan masing-masing kode ke dalam kode string. Pada *stage privat key generation/*

tingkat pembangkit kunci privat memproses kode string tersebut sebagai input untuk pembangkit kunci privat. Pada pembangkit kunci privat dari kode string yang telah ditentukan, algoritma tandatangan *digital* yang digunakan yaitu DSA *(Digital Signature Algorithm).* Dengan DSA kunci publik dan kunci privat dapat dihitung sebagai berikut :

1. Perhitungan p, q dan g dimana :

   p = 512 sampai 1.024 bit bilangan prima

   q = 160 bit faktor prima dari p-1

   g = $h^{(p-1)/q}$ mod p, dimana h<(p-1) dan $h^{(p-1)/q}$ mod p > 1

2. Pembangkit kunci privat

   Hitung SHA1-hash dari kode string yang ditentukan. Nilai hash adalah 160 bit kunci privat yang dinotasikan x.

3. Pembangkit kunci publik

   Hitung $y = g^x$ mod p, dimana y adalah p-bit kunci publik

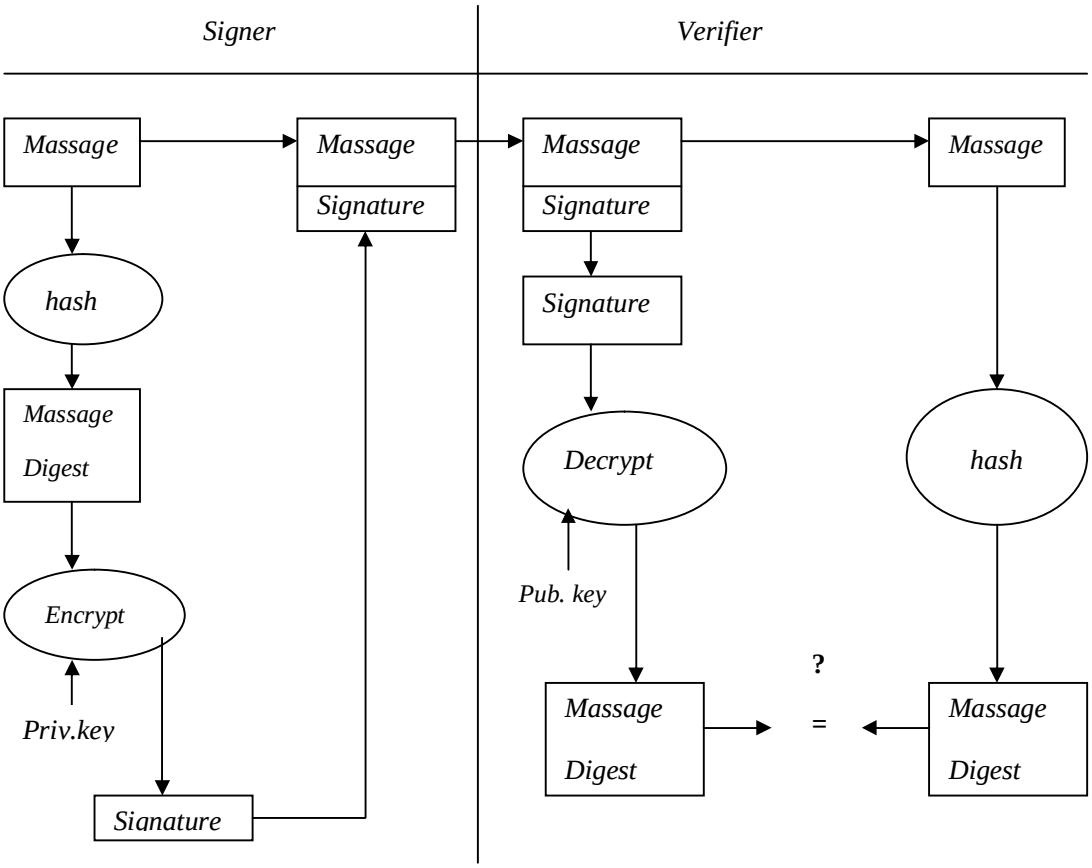
Menurut Munir R (2005), tandatangan *digital* bukanlah tandatangan yang di-dijitasi dengan alat *scanner*, tetapi suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan. Teknik yang umum digunakan untuk membentuk tandatangan *digital* adalah dengan fungsi hash dan melibatkan algoritma kriptografi kunci publik. Mula-mula pesan M ditransformasi oleh fungsi hash H menjadi pesan ringkas h. Pesan ringkas tersebut dienkripsi (E) dengan kunci privat (SK) pengirim pesan: $S = E_{SK}(h)$. Hasil enkripsi (S) inilah yang disebut tandatangan *digital*. Tandatangan *digital* dapat ditambahkan (*append*) pada pesan atau terpisah dari pesan dan

dikirim secara bersamaan. Di tempat penerima, tandatangan diverifikasi untuk dibuktikan keotentikannya dengan cara berikut:

a. Tandatangan *digital* S didekripsi dengan menggunakan kunci publik (PK) pengirim pesan, menghasilkan pesan ringkas semula h, sebagai berikut: $h = D_{PK}(S)$
b. Pengirim kemudian mengubah pesan M menjadi pesan ringkas h' dengan menggunakan fungsi hash satu arah yang sama dengan fungsi hash yang digunakan oleh pengirim.
c. Jika h' = h, berarti tandatangan yang diterima otentik dan berasal dari pengirim yang benar.

Gambar 2.2 memperlihatkan proses pembangkitan tandatangan *digital* (*signing*) pada pihak *signer* dan verifikasi tandatangan *digital* pada pihak *verfier*. Ada beberapa kemungkinan yang terjadi pada pihak *verifier,* yaitu :

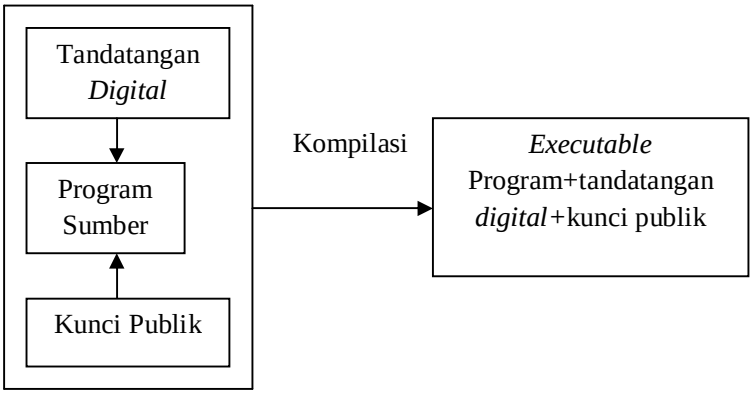
a. Apabila pesan M yang diterima sudah berubah, maka h' yang dihasilkan dari fungsi hash berbeda dengan h semula. Ini berarti pesan tidak asli lagi.
b. Apabila pesan M tidak berasal dari orang yang sebenarnya, maka h yang dihasilkan berbeda dengan h' yang dihasilkan pada proses verifikasi (hal ini karena kunci publik yang digunakan oleh penerima pesan tidak berkoresponden dengan kunci privat pengirim).
c. Bila h = h', ini berarti pesan yang diterima adalah pesan yang asli dan orang yang mengirim adalah orang yang sebenarnya.




**Gambar 2.2** *Proses Penandatanganan dan Verifikasi*

Menurut Munir (2005), tandatangan *digital* dapat diberikan ke sembarang data *digital*, tidak hanya pesan, termasuk di dalamnya berkas program. Ada dua cara yang dapat dilakukan dalam penggunaan tandatangan *digital* untuk berkas program. Cara pertama, tandatangan diletakkan pada berkas terpisah dari berkas *executable*. Namun, pemberian tandatangan dengan cara pertama ini memiliki kelemahan. Pihak ketiga yang memodifikasi program dapat membangkitkan kunci privat dan kunci publiknya sendiri. Kemudian menghitung tandatangan *digital* dengan menggunakan

kunci privatnya, selanjutnya mengganti kunci publik pengembang yang sah dengan kunci publiknya. Pengguna yang men-*download* berkas ini tidak mengetahui bahwa ia telah men-*download* program yang sudah berubah. Cara kedua yang lebih aman adalah menambahkan tandatangan *digital* (termasuk kunci publiknya) pada saat pengembangan program yang ditunjukkan pada gambar 2.3.




**Gambar 2.3** *Penyisipan Tandatangan Digital pada Saat Pengembangan Program*

Dari gambar 2.3 dapat dilihat bahwa tandatangan dibangkitkan dengan mengenkripsi nilai hash dari berkas program dengan menggunakan kunci publik pengembang program. Selanjutnya, tandatangan *digital* dapat disimpan di dalam berkas terpisah atau dikompilasi dengan berkas program sumber sehingga menyatu di dalam program. Integritas perangkat lunak dilakukan dengan memverifikasi tandatangan *digital*. Proses verifikasi membutuhkan kunci publik pengembang program.

## 2.2 Landasan Teori

### 2.2.1 Kriptografi

Kriptografi berasal dari akar kata Yunani kryptos dan gráphō, yang mempunyai arti "tulisan tersembunyi". Kriptografi adalah ilmu yang mempelajari bagaimana membuat suatu pesan yang dikirim pengirim dapat disampaikan kepada penerima dengan aman. Kriptografi dapat memenuhi kebutuhan umum suatu transaksi, yaitu :

1. Kerahasiaan (*confidentiality*) dijamin dengan melakukan enkripsi (penyandian).
2. Keutuhan (*integrity*) atas data dilakukan dengan fungsi *hash* satu arah.
3. Jaminan atas identitas dan keabsahan (*authenticity*) pihak-pihak yang melakukan transaksi dilakukan dengan menggunakan *password* atau sertifikat *digital*. Sedangkan keotentikan data transaksi dapat dilakukan dengan tandatangan digital.
4. Transaksi dapat dijadikan barang bukti yang tidak bisa disangkal (*non-repudiation*) dengan memanfaatkan tandatangan *digital* dan sertifikat *digital*.

Pembakuan penulisan pada kriptografi dapat ditulis dalam bahasa matematika. Fungsi-fungsi yang mendasar dalam kriptografi adalah enkripsi dan dekripsi. Enkripsi adalah proses mengubah suatu pesan asli (*plaintext*) menjadi suatu pesan dalam bahasa sandi (*ciphertext*), C = E (M) dimana :

M = pesan asli

E = proses enkripsi

C = pesan dalam bahasa sandi (untuk ringkasnya disebut sandi)

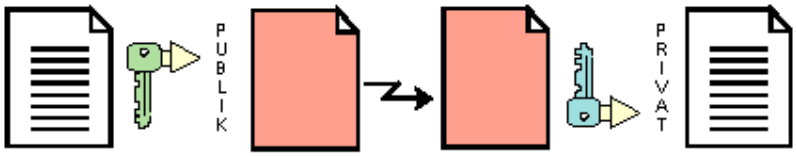
Sedangkan dekripsi adalah proses mengubah pesan dalam suatu bahasa sandi menjadi pesan asli kembali, yaitu M = D (C) dimana D = proses dekripsi. Umumnya, selain menggunakan fungsi tertentu dalam melakukan enkripsi dan dekripsi, seringkali fungsi itu diberi parameter tambahan yang disebut dengan istilah kunci yang dibagi menjadi kunci simetris dan asimetris. (Munir, 2006)

Pada pembahasan selanjutnya yang digunakan adalah kunci asimetris atau disebut kriptografi asimetris/ kriptografi kunci publik.

### 2.2.2 Kriptografi Asimetris/ Kunci Publik

Pada pertengahan tahun 70-an Whitfield Diffie dan Martin Hellman menemukan teknik enkripsi asimetris sering disebut kriptografi asimetris atau kriptografi kunci publik yang merevolusi dunia kriptografi. Kunci asimetris adalah pasangan kunci-kunci kriptografi yang salah satunya dipergunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi. Semua orang yang mendapatkan kunci publik dapat menggunakannya untuk mengenkripsikan suatu pesan, sedangkan hanya satu orang saja yang memiliki rahasia tertentu, dalam hal ini kunci privat untuk melakukan pembongkaran terhadap sandi yang dikirim untuknya.

Dengan cara seperti ini, jika Anto mengirim pesan untuk Badu, Anto dapat merasa yakin bahwa pesan tersebut hanya dapat dibaca oleh Badu, karena hanya Badu yang bisa melakukan dekripsi dengan kunci privatnya. Tentunya Anto harus memiliki kunci publik Badu untuk melakukan enkripsi. Anto bisa mendapatkannya dari Badu, ataupun dari pihak ketiga seperti Tari.




**Gambar 2.4** *Penggunaan Kunci Asimetris*

Teknik enkripsi asimetris ini jauh lebih lambat daripada enkripsi dengan kunci simetris. Oleh karena itu, biasanya bukanlah pesan itu sendiri yang disandikan dengan kunci asimetris, namun hanya kunci simetrislah yang disandikan dengan kunci asimetris. Sedangkan pesannya dikirim setelah disandikan dengan kunci simetris tadi. Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA (merupakan singkatan penemunya yakni Rivest, Shamir dan Adleman) dan DSA (*Digital Signature Algorithm*). (Munir, 2006)

Kriptografi kunci publik yang digunakan pada tandatangan *digital* dapat dinyatakan sebagai :

$E_{kd}$ (M) = C

$D_{ke}$ (C) = M

dimana :

E = enkripsi, D = dekripsi, M = pesan, C = *cipher*, kd = kunci privat, ke = kunci publik. (Kurniawan, 2004)

### 2.2.3 Biometrik

Biometrik berasal dari bahasa Yunani yaitu, bios yang berarti hidup dan metron berarti ukuran. Biometrik adalah suatu metoda untuk mengenali manusia berdasar pada satu atau lebih ciri-ciri fisik atau tingkah laku yang unik. Alasan

menggunakan Biometrik yaitu karena keterbatasan manusia memverifikasi segala hal hanya dari sisi :

a) Verifikasi berdasarkan benda : semua data-data yang dibutuhkan berada pada suatu benda (seperti dokumen atau kartu kredit). Apabila hilang maka orang lain dapat memalsukannya atau menyalahgunakannya.
b) Verifikasi berdasarkan pengetahuan : biasanya menggunakan password, bahkan jika menggunakan algoritma enkripsi terbaikpun, tetap terdapat kunci yang bisa membukanya.

Keunggulan penggunaan biometrik yaitu :

a) Tidak dapat hilang atau lupa
b) Sulit di duplikasi, di-*share* ataupun dipindah tangankan
c) Keaslian lebih terjamin karena harus menghadirkan *person* sebagai alat validasi

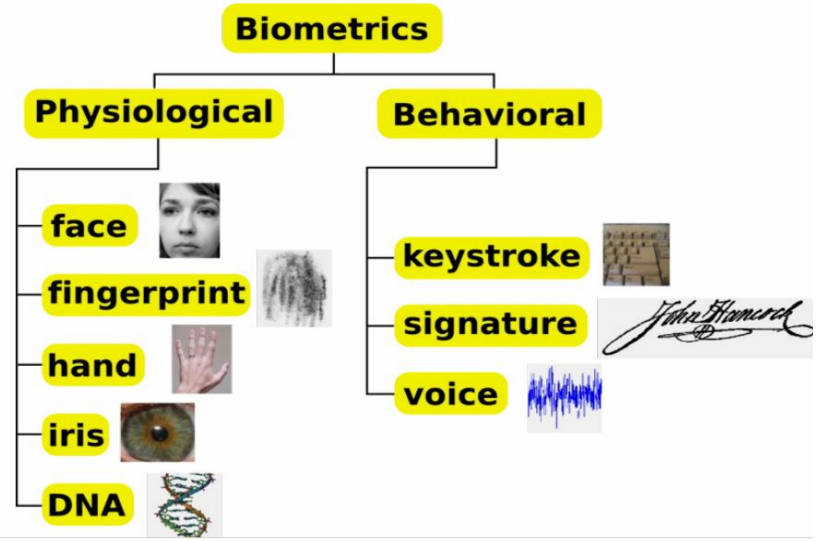
Karakteristik dari Biometrik ditunjukkan pada gambar 2.5 yaitu :

a) Physiological : dihubungkan dengan bentuk tubuh/badan.

Misalnya *fingerprints, face recognition, hand geometry* dan *iris recognition*

b) Behavioral : di hubungkan dengan tingkah laku seseorang.

Misalnya *keystroke, signature, voice.*




**Gambar 2.5** *Bagan Karakteristik Biometrik*

Biometrik menunjukkan identifikasi otomatis dari seseorang berdasarkan phisiologikal atau karakter behavioral. Metode ini mengidentifikasi lebih daripada metode tradisional yang melibatkan password dan PIN. Biometrik mendeskripsikan fisik unik seseorang atau karakteristik behavioralnya. Karena fitur seseorang unik, biometrik merupakan cara untuk mengidentifikasi seseorang dengan latar belakang legal yang cukup (Feng, H; & Chong Wa,C. 2002).

### 2.2.4 Biometrik Tandatangan

Pengertian biometrik tandatangan pertama disebutkan dalam paper Pawan dan Siyal's (2001). Biometrik tandatangan didefinisikan sebagai proses menurunkan kunci privat dari sampel biometrik dan menggunakan kunci privat tersebut untuk menandatangani e-dokumen. Jadi kunci privat unik dapat dibangkitkan dinamis dari salah satu sampel biometrik, tanpa memerlukan penyimpanan. Ini menghilangkan masalah tempat penyimpanan kunci privat yang memecahkan isu manajemen kunci. Pembangkitan dinamis kunci privat

membuktikan kemudahan penandatanganan dokumen sebagaimana dapat menandatangani dokumen kapanpun dimanapun tanpa membawa *disk* atau *smart card* (Feng, H; & Chong Wa,C. 2002).

### 2.2.5 Tandatangan Digital Menggunakan Kriptografi Kunci Publik DSA dan Fungsi *Hash*

Pada beberapa kasus seringkali otentikasi yang diperlukan tetapi kerahasiaan pesan tidak. Maksudnya, pesan tidak perlu dienkripsikan, sebab yang dibutuhkan hanya keotentikan pesan saja. Kebutuhan tersebut dapat dipenuhi dengan pemberian tandatangan *digital*. Algoritma kunci-publik dan fungsi hash dapat digunakan untuk kasus seperti ini. Tandatangan *digital* adalah suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan.

Pada Agustus 1991, NIST (*The National of Standart and Technology*) mengumumkan standard untuk tandatangan *digital* yang dinamakan *Digital Signature Standard* (DSS) yang terdiri dari dua komponen :

*a.* Algoritma tandatangan *digital* yang disebut DSA (*Digital Signature Algorithm)*

b. Fungsi Hash yang disebut SHA ( *Secure Hash Algorithm*)

Jadi DSA untuk penandatanganan pesan dan SHA untuk membangkitkan *massage digest* dari pesan.

Langkah-langkah pada proses tandatangan *digital* sebagai berikut:

a. Menentukan parameter DSA yaitu:

1. p adalah bilangan prima dengan panjang L bit, dimana $2^{L-1} < p < 2^{L}$ dengan $512 \leq L \leq 1024$ dan L adalah kelipatan 64. q, bilangan prima 160 bit, faktor dari p-1 dimana $2^{159} < q < 2^{160}$. Parameter p bersifat publik. Pembangkit bilangan prima lebih rinci, dijelaskan pada subbab 2.2.5.2.
2. $g = h^{(p-1)/q} \bmod p$, dimana $1 < h < p-1$ sehingga $g > 1$. Parameter g bersifat publik.
3. x bilangan bulat yang dibangkitkan random atau pseudorandom dimana $0 < x < q$ dengan panjang160 bit. Parameter x bersifat privat.
4. $y = g^{x} \bmod p$ adalah kunci publik
5. M adalah pesan yang akan diberi tandatangan
6. k = bilangan bulat yang dibangkitkan random atau pseudorandom dimana $0 < k < q$

Parameter p, q dan g bersifat publik dan dapat digunakan bersama dalam sekelompok orang. Parameter p, q dan g juga bernilai tetap untuk periode/ waktu tertentu. Parameter x dan k hanya digunakan untuk pembangkitan tandatangan dan harus dijaga kerahasiaannya. Parameter k harus berbeda untuk setiap tandatangan.

b. Pembangkitan sepasang kunci :

1. Pilih bilangan prima p dan q, dimana $(p-1) \bmod q = 0$
2. Hitung $g = h^{(p-1)/q} \bmod p$, dimana $1 < h < p-1$ dan $g > 1$
3. Tentukan kunci privat $x < q$
4. Hitung kunci publik $y = g^{x} \bmod p$

Jadi didapatkan kunci publik (p,q,g,y) dan kunci privat (p,q,g,x)

c. Pembangkitan tandatangan (*signing*) :

4. Ubah pesan m menjadi massage digest dengan fungsi Hash SHA menghasilkan SHA(M)
5. Tentukan bilangan acak k < q
6. Tanda tangan dari pesan m adalah bilangan r dan s yang didapat dari :

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

   $s = (k^{-1} (SHA(M) + xr)) \bmod q$, $k^{-1}$ adalah invers dari k modulo q.

   Pada perhitungan nilai s, 160-bit string SHA(M) dikonversi terlebih dahulu ke dalam integer yang diberikan di subbab 2.2.6. Jika tandatangan yang dihasilkan benar maka nilai r dan atau s tidak mungkin 0.
7. Kirim pesan beserta tandatangan r dan s

d. Verifikasi keabsahan tandatangan (*verifying*) :

Sebelum diverifikasi, harus dipastikan tersedianya kunci publik pengirim (y), nilai p, q dan g beserta pesan yang bertandatangan r dan s. *Verifier* memeriksa terlebih dahulu apakah $0 < r < q$ and $0 < s < q$ kemudian menghitung :

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

$u_1 = (SHA(M)*w) \bmod q$

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

$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q)$

Jika v = r maka tandatangan sah berarti tandatangan diverifikasi dan *verifier* dapat memiliki keyakinan yang tinggi bahwa pesan yang diterima dikirim

oleh pihak memegang kunci rahasia x sesuai dengan y kunci publiknya, dengan kata lain pesan masih asli dan dokumen dikirim oleh pengirim yang benar. Pembuktian jika v = r' pada M' = M, r' = r dan s'=s diberikan pada subbab 2.2.5.3. Jika v tidak sama r, maka pesan tersebut mungkin telah dimodifikasi, pesan tersebut mungkin telah salah ditandatangani oleh penandatangan, atau pesan mungkin telah ditandatangani oleh pihak lain (bukan penandatangan sebenarnya) berarti pesan tidak valid. Berikut ini diberikan contoh perhitungan DSA :

a. Pembangkitan sepasang kunci

1. Pilih bilangan prima p dan q dengan (p-1) mod q = 0, yaitu

   p = 59419 dan q = 3301 (memenuhi 3301.18 = 59419-1)

2. Hitung $g = h^{(p-1)/q} \bmod p$, dimana $1 < h < p-1$ dan $g > 1$, yaitu (ambil h = 100) $g = 100^{(59419-1)/3301} \bmod 59419 = 18870$

3. Tentukan kunci rahasia x bilangan bulat < q, ambil x = 3223

4. Hitung kunci publik $y = g^x \bmod p = 18870^{3223} \bmod 59419 = 29245$

b. Pembangkitan tandatangan (*signing*)

1. Hitung nilai *hash* dari pesan, misal H(m) = 4321

2. Tentukan bilangan acak k < q, misal diambil k = 997, $k.k^{-1} = 1 \bmod q$, didapat $k^{-1} = 2907$

3. Hitung r dan s sebagai berikut :

   $r = (g^k \bmod p) \bmod q = (18870^{997} \bmod 59419) \bmod 3301 = 848$

   $s = (k^{-1} (H(m) + x\ r)) \bmod q = (2907( 4321+3223. 848)) \bmod 3301 = 183$

4. Kirim pesan m dan tandatangan r dan s

c. Verifikasi keabsahan tandatangan

1. Hitung

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

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

   $w = 469 \bmod 3301 = 469$

   $u_1 = (H(m)*w) \bmod q = (4321.\ 469) \bmod 3301 = 3036$

   $u_2 = (r*w) \bmod q = (848.\ 469) \bmod 3301 = 1592$

   $v = ((g^{u1} * y^{u2}) \bmod p) \bmod q)$

   $= ((18870^{3036} . 29245^{1592}) \bmod 59419) \bmod 3301 = 848$

2. Karena v = r maka tanda tangan sah.

(Munir, 2006)

**2.2.5.1 Tes Primalitas**

Pada pembangkitan bilangan prima p dan q, DSA membutuhkan tes primalitas untuk menentukan suatu bilangan bulat adalah prima, salah satunya dengan Algoritma M.O Rabin pada publikasi FIPS PUB 186-3 sebagai berikut:

Langkah 1. Ambil i = 1 dan n $\geqslant$ 50.

Langkah 2. Ambil w = bilangan integer yang di-test, $w = 1 + 2^a m$, dimana m bilangan ganjil dan $2^a$ adalah pangkat tertinggi dari 2 dibagi w - 1.

Langkah 3. Bangkitkan bilangan random integer b pada range 1 < b < w.

Langkah 4. Ambil j = 0 dan $z = b^m \bmod w$.

Langkah 5. Jika j = 0 dan z = 1, atau jika z = w - 1, ke langkah 9.

Langkah 6. Jika j > 0 dan z = 1, ke langkah 8.

Langkah 7. j = j + 1. Jika j < a, maka $z = z^2 \bmod w$ , kembali ke langkah 5.

Langkah 8. w bukan prima. *Stop*.

Langkah 9. Jika i < n, ambil i = i + 1 dan kembali ke langkah 3. Sebaliknya w mungkin prima.

Pada iterasi ke n akan didapat probabilitas bukan prima tidak lebih besar dari $1/4^n$ . Untuk $n \geq 50$ akan memberikan probabilitas *error* yang dapat diterima. (FIPS PUB 186-3, 2009)

**2.2.5.2 Pembangkit Bilangan Prima Pada DSA**

Pada DSA dua bilangan prima p dan q memenuhi tiga kondisi yaitu :

a. $2^{159} < q < 2^{160}$

b. $2^{L-1} < p < 2^L$ , dimana $L = 512 + 64j$ untuk beberapa $0 \leq j \leq 8$

c. q pembagi p - 1.

Pembangkit prima dimulai dengan SHA dan *user* disediakan SEED untuk membangun prima q pada range $2^{159} < q < 2^{160}$ . Kemudian nilai SEED digunakan untuk membangun x pada range $2^{L-1} < x < 2^L$ . Prima p ditentukan dari pembulatan x ke bilangan yang kongruen dengan 1 mod 2q sebagai berikut :

Integer x pada range $0 \leq x < 2^g$ dikonversi pada barisan sepanjang g bit dengan ekspansi biner yaitu :

$$x = x_1*2^{g-11} + x_2*2^{g-2} + ... + x_{g-1}*2 + x_g \rightarrow \{ x_1,...,x_g \}.$$

Sebaliknya jika akan diinvers barisan sepanjang g bit $\{ x_1,...,x_g \}$dikonversi menjadi integer dengan aturan :

$$\{ x_1,...,x_g \} \rightarrow x_1*2^{g-1} + x_2*2^{g-2} + ... + x_{g-1}*2 + x_g.$$

Bit pertama dari barisan tersebut adalah MSB (*the most significant bit*) dari integer dan bit terakhir adalah LSB (*the least significant bit*).

Ambil L - 1 = n*160 + b, dimana b dan n adalah integer dan $0 \leq b < 160$.

Langkah-langkah untuk membangkitkan bilangan prima pada DSA sebagai berikut :

Langkah 1. Ambil barisan minimal 160 bit dan sebut sebagai SEED. Panjang SEED dalam bit adalah g.

Langkah 2. Hitung

$$U = SHA[SEED] \text{ XOR } SHA[(SEED+1) \bmod 2^g ].$$

Langkah 3. Nilai q adalah U dengan mengambil *the most significant bit* (the $2^{159}$ bit) dan *the least significant bit* 1. Dalam operasi boolean, q = U OR $2^{159}$ OR 1 dengan $2^{159} < q < 2^{160}$.

Langkah 4. Gunakan algoritma tes primalitas untuk menentukan apakah q prima.

Langkah 5. Jika q tidak prima, kembali ke langkah 1.

Langkah 6. Ambil *counter* = 0 dan *offset* = 2.

Langkah 7. Untuk k = 0,...,n , hitung $V_k = SHA[(SEED + offset + k) \bmod 2^g ]$.

Langkah 8. Hitung W dalam integer yaitu :

$$W = V_0 + V_1*2^{160} + ... + V_{n-1}*2^{(n-1)*160} + (V_n \bmod 2^b) * 2^{n*160}$$ dan

hitung $X = W + 2^{L-1}$. Dengan *range* $0 \leq W < 2^{L-1}$ dan $2^{L-1} \leq X < 2^L$.

Langkah 9. Ambil c = X mod 2q dan hitung p = X - (c - 1). Dengan catatan p kongruen terhadap 1 mod 2q.

Langkah 10. Jika $p < 2^{L-1}$, ke langkah 13.

Langkah 11. Lakukan uji primalitas untuk p.

Langkah 12. Jika p lolos tes pada langkah 11, ke langkah 15.

Langkah 13. Ambil counter = counter + 1 dan offset = offset + n + 1.

Langkah 14. Jika counter $\geqslant 2^{12} = 4096$ ke langkah 1, sebaliknya (jika counter < 4096) ke langkah 7.

Langkah 15. Simpan nilai SEED dan nilai *counter* untuk digunakan pada penentuan nilai p dan q.

Kekuatan tes pembangkit bilangan prima tersebut adalah kemungkinan suatu bilangan bukan prima lolos tes adalah kurang dari $2^{-80}$.

(FIPS PUB 186-3, 2009)

**2.2.5.3 Pembuktian v = r'**

Jika pesan yang dikirim M dengan tandatangan r dan s , pesan yang diterima M' dengan tandatangan r' dan s' , y adalah kunci publik penandatangan akan dibuktikan untuk M' = M, r' = r and s' = s tandatangan diverifikasi jika v = r'.

**Lemma 2.1** :

Jika p dan q prima dan q pembagi (p-1), h integer positif lebih kecil dari p dan $g = h^{(p-1)/q} \bmod p$ maka $g^q \bmod p = 1$ dan jika m mod q = n mod q maka $g^m \bmod p = g^n \bmod p$.

Bukti :

$g^q \bmod p = (h^{(p-1)/q} \bmod p)^q \bmod p$

$= h^{(p-1)} \bmod p$

$= 1$

dengan teorema little Fermat (*Fermat's Little Theorem*).

Jika m mod q = n mod q, maka , m = n + kq untuk beberapa integer k sehingga :

$g^m \bmod p = g^{n+kq} \bmod p$

$= (g^n g^{kq}) \bmod p$

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

$= g^n \bmod p$

Jadi terbukti $g^q \bmod p = 1$.

**Teorema 2.1 :** (Pembuktian v = r')

Jika M' = M, r' = r, dan s' = s pada verifikasi tandatangan, maka v = r'.

Bukti :

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

$u1 = ((SHA(M'))w) \bmod q = ((SHA(M))w) \bmod q$

$u2 = ((r')w) \bmod q = (rw) \bmod q.$

Diketahui $y = g^x \bmod p$, dengan lemma 2.1 didapat :

$v = ((g^{u1} y^{u2}) \bmod p) \bmod q$

$= ((g^{SHA(M)w} y^{rw}) \bmod p) \bmod q$ , $y = g^x \bmod p$

$= ((g^{SHA(M)w} g^{xrw}) \bmod p) \bmod q$

$= ((g^{(SHA(M)+xr)w}) \bmod p) \bmod q.$

Juga

$s = (k^{-1}(SHA(M) + xr)) \bmod q.$

Sehingga

$w = (k(SHA(M) + xr)^{-1}) \bmod q$

$(SHA(M) + xr)w \bmod q = k \bmod q.$

Dengan lemma 2.1 didapat :

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

$= r$

$= r'$.

(FIPS PUB 186-3, 2009)

#### 2.2.5.4 Pembangkit Bilangan Random Pada DSA

Pada komputasi algoritma DSA diperlukan bilangan acak atau pseudorandom x dan k. Bilangan k adalah unik untuk setiap pesan. Bilangan acak yang dihasilkan dalam range 0 sampai 160 bit. Pembangkit bilangan pseudorandom integer yang direkomendasikan FIPS menggunakan fungsi satu arah G (t, c), dimana t adalah 160 bit, c adalah b bit (160 sampai 512) dan G (t, c) adalah 160 bit. Salah satu cara untuk membangun G adalah melalui *Secure Hash Algorithm* (SHA), sebagaimana didefinisikan dalam *Secure Hash Standard* (SHS). Pesan 160-bit *digest output* dari algoritma SHA dari *input* pesan M dinotasikan dengan SHA (M) (FIPS PUB 186-3, 2009).

#### 2.2.5.5 Algoritma Komputasi m Nilai x

Kunci privat penandatangan adalah x, berikut ini adalah algoritma untuk membangkitkan m nilai x.

Langkah 1. Pilih bilangan (bersifat rahasia) *seed-key* sebagai XKEY.

Langkah 2. Ambil dalam notasi hexa :

t = 67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0.

sebagai nilai awal untuk H0 || H1 || H2 || H3 || H4 pada SHA-1.

Langkah 3. Untuk j = 0 sampai m - 1 hitung :

a. $XSEED_j$ = *optional user input.*

b. XVAL = (XKEY + $XSEED_j$) mod $2^b$.

c. $x_j$ = G(t,XVAL) mod q.

d. XKEY = (1 + XKEY + $x_j$) mod $2^b$.

(FIPS PUB 186-3, 2009)

**2.2.5.6 Algoritma Prekomputasi Nilai k dan r**

Algoritma berikut digunakan untuk prekomputasi k, k-1 dan r untuk m pesan pada suatu waktu tertentu.

Langkah 1. Pilih bilangan (bersifat rahasia) *seed-key* sebagai KKEY.

Langkah 2. Ambil dalam notasi hexa :

t = EFCDAB89 98BADCFE 10325476 C3D2E1F0 67452301.

sebagai nilai awal untuk pergantian iterasi yaitu H0 || H1 || H2 || H3 || H4 pada SHA-1.

Langkah 3. Untuk j = 0 sampai m - 1 hitung

a. k = G(t,KKEY) mod q.

b. Hitung $k_j^{-1} = k^{-1}$ mod q.

c. Hitung $r_j = (g^k \text{ mod } p) \text{ mod } q$.

d. KKEY = (1 + KKEY + k) mod $2^b$.

Langkah 4. Ambil $M_0$ , ... , $M_{m-1}$ sebagai m pesan berikutnya.

Untuk j = 0 sampai m - 1 hitung :

a. Ambil h = SHA($M_j$).

b. Ambil $s_j = (k_j^{-1}(h + xr_j)) \text{ mod } q$.

c. Tandatangan digital untuk $M_j$ adalah ($r_j$,$s_j$).

Langkah 5. Ambil t = h.

Langkah 6. Ke langkah 3.

Pada langkah 3 dilakukan prekomputasi dari kuantitas yang diperlukan untuk menandatangani pesan m berikutnya. Eksekusi langkah 4 dimulai ketika pesan m pertama telah siap diproses dan dapat ditunda ketika pesan m berikutnya belum siap. Segera setelah langkah 4 dan 5 lengkap, langkah 3 dapat dieksekusi dan hasilnya disimpan sampai pesan m berikutnya siap.

Ruang untuk KKEY adalah dua array dengan panjang m untuk menyimpan $r_0$ , ... $r_{m-1}$ dan $k_0^{-1}$, ... , $k_{m-1}^{-1}$ yang dikomputasi pada langkah 3. Penyimpanan $s_0$ , ... , $s_{m-1}$ hanya diperlukan jika tandatangan untuk sekelompok pesan disimpan, sebaliknya $s_j$ pada langkah 4 dapat diganti dengan ruang tunggal. (FIPS PUB 186-3, 2009)

**2.2.5.7 Membangun Fungsi G dari SHA**

Fungsi G (t, c) dapat dibangun dengan menggunakan langkah-langkah (a) - (e) menurut *Spesifikasi Secure Hash Standard*. Sebelum menjalankan langkah ini, $\{H_j\}$ dan $M_1$ harus diinisialisasi sebagai berikut:

i. Inisialisasi $\{H_j\}$ dengan membagi 160 bit dari t ke dalam lima bagian 32-bit sebagai berikut :

$t = t_0 \| t_1 \| t_2 \| t_3 \| t_4$

Kemudian $H_j = t_j$ untuk j = 0 sampai 4.

ii. Hanya ada satu blok pesan $M_1$ , yang berinisial :

$M_1 = c \| 0^{512-b}$ (b bit pertama dari $M_1$ berisi c dan sisanya (512-b) bit adalah 0)

Kemudian langkah a sampai e berikut dieksekusi.

Sebelum diproses, diinisialisasi $\{H_i\}$ dalam notasi hexa sebagai berikut :

$H_0$ = 67452301

$H_1$ = EFCDAB89

$H_2$ = 98BADCFE

$H_3$ = 10325476

$H_4$ = C3D2E1F0

Kemudian $M_1$, $M_2$, ... , $M_n$ diproses sebagai berikut :

a. Bagi $M_i$ dalam 16 *words* $W_0$, $W_1$, ... , $W_{15}$, dimana $W_0$ adalah *word* paling kiri.

b. Untuk t = 16 sampai 79 ambil $W_t = S^1(W_{t-3}$ XOR $W_{t-8}$ XOR $W_{t-14}$ XOR $W_{t-16})$.

c. Ambil A = $H_0$, B = $H_1$, C = $H_2$, D = $H_3$, E = $H_4$.

d. Untuk t = 0 smpai 79 hitung

TEMP = $S^5(A) + f_t(B,C,D) + E + W_t + K_t$;

E = D; D = C; C = $S^{30}(B)$; B = A; A = TEMP;

e. Ambil $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

Hasil akhir dari langkah e didapat G(t,c) 160 bit string yang direpresentasikan pada lima *word* $H_0 \| H_1 \| H_2 \| H_3 \| H_4$ . (FIPS PUB 183-3, 2008)

**2.2.5.8 Pembangkitan Kuantitas g, $k^{-1}$ dan $s^{-1}$**

Algoritma berikut digunakan untuk membangkitkan kuantitas g, $k^{-1}$, and $s^{-1}$ pada *digital signature standart.*

Untuk membangkitkan g :

langkah 1. Bangkitkan p dan q analog pada subbab 2.2.5.2

langkah 2. Ambil e = (p - 1)/q.

langkah 3. Ambil h = bilangan integer, dimana 1 < h < p – 1 dan h berbeda dari nilai yang sudah ada.

langkah 4. Ambil $g = h^e \bmod p$.

langkah 5. Jika g = 1, ke langkah 3.

Algoritma untuk perhitungan multiplikasi invers $n^{-1} \bmod q$ untuk n dengan $0 < n < q$, dimana $0 < n^{-1} < q$ yaitu :

langkah 1. Ambil i = q, h = n, v = 0, dan d = 1.

langkah 2. Ambil t = i DIV h, dimana DIV adalah pembagi integer.

langkah 3. Ambil x = h.

langkah 4. Ambil h = i - tx.

langkah 5. Ambil i = x.

langkah 6. Ambil x = d.

langkah 7. Ambil d = v - tx.

langkah 8. Ambil v = x.

langkah 9.Jika h > 0, kembali ke langkah 2.

langkah 10.Hitung $n^{-1} = v \bmod q$.

Pada langkah 10, v dimungkinkan bernilai negatif. Nilai operasi v mod q harus diantara 1 dan q - 1.

(FIPS PUB 186-3, 2009)

**2.2.6 Fungsi Hash SHA-1**

Fungsi Hash (*hash function*) merupakan fungsi yang bersifat satu arah dimana jika dimasukkan data, maka akan menghasilkan sebuah "*checksum*" atau "*fingerprint*" dari data tersebut. Sebuah pesan yang dilewatkan ke fungsi hash akan menghasilkan keluaran yang disebut *Message Authenticated Code* (MAC). Dilihat dari sisi matematik, *hash function* memetakan satu set data ke dalam

sebuah set yang lebih kecil dan terbatas ukurannya. Fungsi hash satu arah mempunyai sifat sebagai berikut :

1. Diberikan M , harus mudah menghitung H(M) = h
2. Diberikan M , sangat sulit atau mustahil mendapatkan M sedemikian sehingga H(M) = h
3. Diberikan M , sangat sulit atau mustahil mendapatkan M' sedemikian sehingga H(M) = H(M'). Bila diperoleh pesan M' semacam ini maka disebut tabrakan (*collision*).
4. Sangat sulit atau mustahil mendapatkan dua pesan M dan M' sedemikian sehingga H(M) = H(M').

Sebuah fungsi hash satu arah H(M) beroperasi pada *pre-image* pesan M dengan panjang sebarang dan mengembalikan nilai hash h yang memiliki panjang tetap. Fungi hash dikembangkan berdasarkan ide sebuah fungsi kompresi. Fungsi satu arah ini menghasilkan nilai hash berukuran n pada input sebesar b. Input tersebut berupa suatu fungsi kompresi blok pesan dan hasil blok pesan sebelumnya. Sehingga hash suatu blok M adalah : $h_i = f(M_i,h_{i-1})$ dimana :

$h_i$ = nilai hash saat ini

$M_i$ = blok pesan saat ini

$h_{i-1}$ = nilai hash blok pesan sebelumnya

SHA-1 adalah algoritma hash yang paling banyak digunakan publik (selanjutnya ditulis SHA). SHA merupakan keluarga fungsi hash satu arah. SHA menerima masukan berupa pesan dengan ukuran maksimum $2^{64}$ bit (2.147.483.648 gigabyte) dan menghasilkan *massage digest* (MD) dengan

panjang 160 bit. MD kemudian digunakan dalam DSA untuk menghitung tandatangan digital pesan tersebut. MD pesan yang sama dapat diperoleh oleh *verifier* ketika menerima pesan dari pengirim dengan cara memasukkan pesan tersebut pada fungsi SHA. SHA dikatakan aman karena secara matematis tidak mungkin menemukan dua pesan yang berbeda yang menghasilkan MD yang sama atau tidak mungkin menemukan pesan aslinya jika diberikan suatu nilai hash-nya.

Pesan M dengan panjang l bit dimana $0 \leq l \leq 2^{64}$ . Algoritma pada SHA-1 menggunakan :

1. Pesan yang tersusun (*message schedule*) pada 80 dari 32 bit word.
2. Lima variabel kerja pada masing-masing 32 bit.
3. Nilai hash pada lima dari 32 bit word, hasil akhir adalah 160 bit MD.

   Word pada *message schedule* diberi label $W_0$, $W_1$ , …, $W_{79}$ . Lima variabel kerja diberi label a, b, c, d dan e. Word dari nilai hash diberi label $H_0^{(i)}$, $H_1^{(i)}$ , …, $H_4^{(i)}$ yang akan menampung nilai hash awal $H^{(0)}$ untuk diganti dengan nilai hash yang berurutan setelah blok pesan diproses $H^{(i)}$ dan berakhir dengan nilai hash final $H^{(N)}$ . SHA-1 juga menggunakan *temporary* tunggal *word* T.

(FIPS PUB 183-3, 2008)

#### 2.2.6.1 *Preprocessing*/ Proses Awal SHA-1

Proses awal SHA-1 yaitu :

1. *Padding* pesan M sehingga berkelipatan 512 bit. Pada pesan M dengan panjang ℓ bit, tambahkan (pad) bit 1 pada akhir pesan diikuti k bit 0,

dimana k adalah solusi nonnegatif terkecil dari persamaan $l + \ell + k \equiv 448 \mod 512$. Kemudian tambahkan blok 64 sehingga sama dengan panjang $\ell$ pada notasi biner. Misal pesan adalah ”abc” mempunyai panjang 8x3 = 24, pesan di-*padding* dengan satu bit 1 kemudian 448-(24+1) = 423 bit 0 dan panjang pesan yaitu 24, sehingga panjang pesan menjadi 512 bit atau kelipatannya .

$$\underbrace{01100001}_{a}\ \underbrace{01100010}_{b}\ \underbrace{01100011}_{c}\ 1\ \overbrace{00...00}^{423}\ \overbrace{00...0\underbrace{11000}_{l=24}}^{64}$$

2. *Parsing*/ uraikan pesan M dalam N blok 512 bit, $M^{(1)}$, $M^{(2)}$ , …, $M^{(N)}$ . Hal ini karena 512 bit dari blok input dituliskan sebagai 32-bit word sebanyak 16. Blok pertama 32 bit dari blok pesan i dinotasikan $M_0^{(i)}$ , 32-bit berikutnya $M_1^{(i)}$ dan seterusnya sampai $M_{15}^{(i)}$ .
3. *Setting* nilai hash awal $H_0^{(0)}$ yaitu 32-bit *word* sebanyak lima, dalam hexa sebagai berikut :

   $H_0^{(0)}$ = 67452301

   $H_1^{(0)}$ = efcdab89

   $H_2^{(0)}$ = 98badcfe

   $H_3^{(0)}$ = 10325476

   $H_4^{(0)}$ = c3d2e1f0

(FIPS PUB 183-3, 2008).

**2.2.6.2 Komputasi Hash SHA-1**

Komputasi hash SHA-1 menggunakan fungsi logika $f_0$, $f_1$…, $f_{79}$ . Masing-masing fungsi dimana $0 \leq t \leq 79$ beroperasi pada tiga 32-bit word, yaitu x,y,z

dan menghasilkan keluaran 32-bit. Fungsi $f_t$ didefinisikan pada persamaan 2.1 yaitu :

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z), & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z, & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z, & 60 \le t \le 79 \end{cases} \quad \text{……..}$$

(2.1)

SHA-1 menggunakan delapan konstanta 32-bit word, Kt pada persamaan 2.2 yaitu :

$$K_t = \begin{cases} 5a827999, & 0 \le t \le 19 \\ 6ed9eba1, & 20 \le t \le 39 \\ 8f1bbcdc, & 40 \le t \le 59 \\ ca62c1d6, & 60 \le t \le 79 \end{cases} \quad \text{…….}$$

(2.2)

Setelah *preprocessing* lengkap, blok pesan $M^{(1)}$, $M^{(2)}$ , …, $M^{(N)}$ kemudian diproses dengan langkah sebagai berikut :

Untuk i = 1 sampai N lakukan :

1. Persiapkan urutan pesan $\{W_t\}$ yaitu :

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \text{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79 \end{cases}$$

2. Inisialisasi lima variabel kerja a, b, c, d dan e dengan nilai hash ke (i-1) yaitu:

   $a = H_0^{(i-1)}$

   $b = H_1^{(i-1)}$

$c = H_2^{(i-1)}$

$d = H_3^{(i-1)}$

$e = H_4^{(i-1)}$

3. Untuk t = 0 sampai 79 :

   $T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$

   $e = d$

   $d = c$

   $c = ROTL^{30}(b)$

   $b = a$

   $a = T$

4. Hitung nilai tengah hash ke-i, $H^{(i)}$ yaitu :

   $H_0^{(i)} = a + H_0^{(i-1)}$

   $H_1^{(i)} = b + H_1^{(i-1)}$

   $H_2^{(i)} = c + H_2^{(i-1)}$

   $H_3^{(i)} = d + H_3^{(i-1)}$

   $H_4^{(i)} = e + H_4^{(i-1)}$

Setelah empat langkah diatas diulang sebanyak N kali akan menghasilkan 160 bit *message digest* dari pesan M yaitu :

$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$ .

(FIPS PUB183-3, 2008)

### 2.2.7 Tandatangan Digital dengan Metode *Hybrid* : Biometrik Tandatangan dan DSA (*Digital Signature Algorithm*)

Pada metode *hybrid* : biometrik tandatangan dan DSA, biometrik yang digunakan yaitu tandatangan *offline.* Kunci privat dan publik dapat dihitung sebagai berikut :

1. Perhitungan p, q dan g
   a) p = 512 sampai 1.024 bit bilangan prima
   b) q = 160 bit faktor prima dari p-1
   c) g = $h^{(p-1)/q}$ mod p, dimana h<(p-1) dan $h^{(p-1)/q}$ mod p > 1
2. Pembangkit kunci privat

   Hitung kode string tandatangan *offline* yang ditentukan. Ambil nilai kode sting tandatangan *offline* sebagai nilai SEED untuk membangkitkan kunci privat x.
3. Pembangkit kunci publik

   Hitung y = $g^x$ mod p. Nilai y adalah p-bit kunci publik

Langkah-langkah DSA analog seperti diberikan pada subbab 2.2.5.

### 2.2.8 OpenSSL

OpenSSL merupakan *open source implementasi* dari protokol SSL (*Secure Socket Layer*) dan TLS (*Transport Layer Security*). SSL dan TLS, merupakan kelanjutan dari protokol kriptografi yang menyediakan komunikasi yang aman di internet. OpenSSL termasuk salah satu program *open source* yang direkomendasikan pada standar keamanan FIPS 140-2 yang dikeluarkan oleh NIST pada CMVP (*Cryptographic Module Validation Program*). Jadi sudah

menjadi *tool* standard dan bebas pakai. OpenSSL dapat diunduh di http://www.openssl.org/.

*Library* OpenSSL digunakan untuk implementasi semua versi protokol SSL, terutama untuk penggunaan algoritma kriptografi simetris dan kriptografi kunci publik, algoritma hash dan *message digest*, juga pembangkit bilangan pseudorandom dan mendukung tandatangan digital untuk manipulasi format sertifikat umum/ bersama serta me-*manage* material kunci/ key. OpenSSL mendukung semua penggunaan kriptografi dan akselerasi hardware. Kriptografi pada openSSL merupakan *link* sederhana pada *socket library* ketika membangun program. OpenSSL tidak membutuhkan *developer* memahami algoritma dan *protocol* kriptografi, tetapi hanya membutuhkan pemahaman bagaimana mengaplikasikan algoritma kriptografi.

Beberapa contoh fungsi pada penerapan openSSL untuk DSA dan fungsi hash SHA-1 yaitu :

- $ openssl dsaparam -out dsaparam.pem 1024

  Me-*generate* parameter DSA dan menuliskannya pada file dsaparam.pem. Panjang bilangan prima dan parameternya maksimum 1024 bit. Ekstensi pem berarti nilai yang diberikan pada format *mode base 64.*

- $ openssl gendsa -out dsaprivatekey.pem -des3 dsaparam.pem

  Me-generate kunci privat DSA menggunakan parameter dari file dsaparam.pem dan mengenkripnya dengan chipper 3DES kemudian menyimpan hasilnya pada file dsaprivatekey.pem.

- $ openssl dsa -in dsaprivatekey.pem -pubout -out dsapublickey.pem

Menghitung kunci public yang berpasangan dengan kunci privat pada file dsaprivatkey.pem dan menyimpannya pada file dsapublickey.pem

- $ openssl dsa -in dsaprivatekey.pem -out dsaprivatekey.pem -des3 -passin pass:oldpword -passout pass:newpword

  Membaca kunci privat dari file dsaprivatekey.pem, mendekripnya dengan password “oldpword”, di-enkrip kembali menggunakan password “newpword” dan menuliskan kunci privat yang di-enkrip pada file dsaprivatekey.pem

- $ openssl dgst -sha1 file.txt

  Menghitung hash SHA-1 dari nama file : file.txt dan menuliskannya pada format heksadesimal.

- $ openssl sha1 -out digest.txt file.txt

  Menghitung SHA-1 dari file.txt dan menuliskan pada digest.txt pada format heksadesimal.

- $ openssl dgst -dss1 -sign dsakey.pem -out dsasign.bin file.txt

  Menandatangani hash dss1 dari file.txt menggunakan kunci privat pada dsakey.pem dan menuliskan keluaran tandatangan pada dsasign.bin.

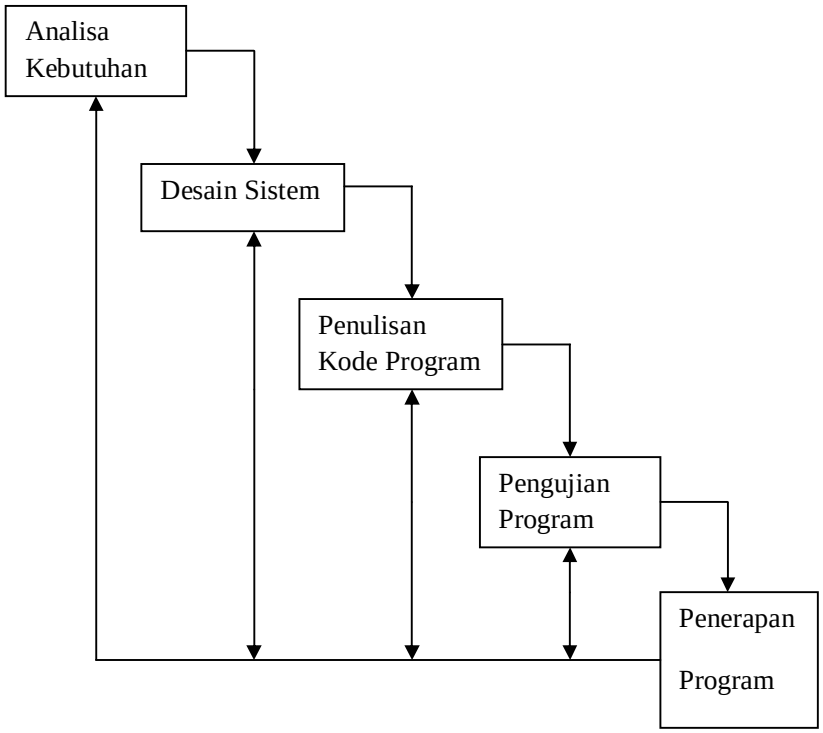
- $ openssl dgst -dss1 -prverify dsakey.pem -signature dsasign.bin file.txt

  Memverifikasi tandatangan dari file.txt dengan dsasign.bin menggunakan SHA-1 dan kunci privat DSA dsakey.pem

(Chandra Pravir, Messier Matt, Viega John, 2002)

### 2.2.9 Model Pengembangan Sistem dengan Metode *Waterfall*

Metode *Waterfall* adalah sebuah metode pengembangan software yang bersifat sekuensial dan terdiri dari 5 tahap yang saling terkait dan mempengaruhi seperti terlihat pada gambar 2.6.




**Gambar 2.6** *Tahapan Pengembangan Sistem dengan Metode Waterfall*

(Pressman, 1997)

Keterkaitan dan pengaruh antar tahap ini ada karena *output* sebuah tahap dalam *Waterfall Model* merupakan input bagi tahap berikutnya, dengan demikian ketidaksempurnaan hasil pelaksanaan tahap sebelumnya adalah awal ketidaksempurnaan tahap berikutnya. Memperhatikan karakteristik ini, sangat penting bagi tim pengembang dan perusahaan untuk secara bersama-sama melakukan analisa kebutuhan dan desain sistem sesempurna mungkin sebelum

masuk ke dalam tahap penulisan kode program. Berikut adalah penjelasan detail dari masing-masing tahap dalam *Waterfall model*, yaitu :

1. Analisa kebutuhan.

   Analisa kebutuhan merupakan tahap pertama yang menjadi dasar proses pembuatan *software*. Kelancaran proses pembuatan software secara keseluruhan dan kelengkapan fitur *software* yang dihasilkan sangat tergantung pada hasil analisa kebutuhan ini. Untuk memperoleh informasi tentang proses bisnis dan kebutuhan perusahaan, umumnya tim pengembang melakukan wawancara, diskusi dan survey. Beberapa perusahaan membantu memperlancar penyelesaian tahap ini dengan terlebih dahulu menyusun *scope of work software* yang akan dibuat sebagai acuan kerja tim pengembang. Hasil analisa kebutuhan yang tidak lengkap berpotensi menyebabkan beberapa permasalahan yang tidak diharapkan, antara lain : waktu pembuatan *software* menjadi lebih lama, proses dalam *software* tidak sesuai dengan proses bisnis dan *software* tidak dapat memenuhi semua kebutuhan perusahaan. Untuk meminimalkan risiko ini, disarankan perusahaan melakukan konfirmasi pemahaman tim pengembang tentang proses bisnis dan kebutuhan perusahaan dengan cara meminta resume hasil analisa kebutuhan dan menyempurnakannya bersama tim pengembang jika diperlukan.

2. Desain sistem.

Desain sistem merupakan tahap penyusunan proses, data, aliran proses dan hubungan antar data yang paling optimal untuk menjalankan proses bisnis dan memenuhi kebutuhan perusahaan sesuai dengan hasil analisa kebutuhan. Dokumentasi yang dihasilkan dari tahap desain sistem ini antara lain : *System Flow, Data Flow Diagram (DFD) dan Entity Relationship Diagram (ERD). System Flow* merupakan bagan aliran dokumen dari satu bagian perusahaan ke bagian lain baik secara manual maupun melalui Sistem Informasi. *Data Flow Diagram* adalah diagram yang menunjukan aliran data di antara pengguna, proses dan database yang terkait dengan software. *Entity Relationship Diagram* merupakan diagram yang menunjukan bagaimana data dan informasi software akan di simpan di dalam database beserta dengan hubungan antar data.

3. Penulisan kode program.

   Penulisan kode program merupakan tahap penerjemahan desain sistem yang telah dibuat ke dalam bentuk perintah-perintah yang dimengerti komputer dengan mempergunakan bahasa pemrograman, *middleware* dan *database* tertentu di atas *platform* yang menjadi standar perusahaan.

4. Pengujian program.

   Pengujian *software* dilakukan untuk memastikan bahwa *software* yang dibuat telah sesuai dengan desainnya dan semua fungsi dapat dipergunakan dengan baik tanpa ada kesalahan. Pengujian *software* biasanya dilakukan dalam 2 atau 3 tahap yang saling independen, yaitu : pengujian oleh internal tim pengembang, pengujian oleh divisi *Quality*

*Assurance* dan pengujian oleh pengguna di perusahaan. Dalam tahap ini, perusahaan harus memastikan bahwa kerangka / skenario pengujian *software* dibuat dengan lengkap meliputi semua proses, kebutuhan dan pengendalian yang ada di dalam dokumen analisa kebutuhan dan desain sistem.

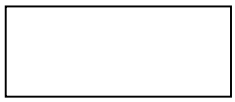
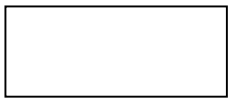
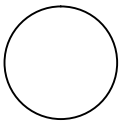
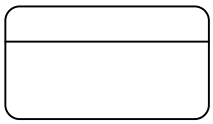
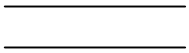
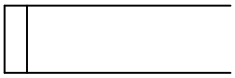
5. Penerapan program.

   Penerapan program merupakan tahap dimana tim pengembang menerapkan / meng-*install software* yang telah selesai dibuat dan diuji ke dalam lingkungan Teknologi Informasi perusahaan dan memberikan pelatihan kepada pengguna di perusahaan. Pada saat melaksanakan pelatihan perusahaan harus yakin bahwa semua karyawan benar-benar menguasai Sistem Informasi yang dibuat sesuai dengan tugas, kewenangan dan tanggung-jawabnya. Untuk mendukung penguasaan ini pada waktu operasional harian setelah pelatihan, perusahaan sebaiknya memastikan pengembang telah memberikan buku *User Manual* dari Sistem Informasi yang dibuat.

   (Pressman, 1997)

### 2.2.10 *Data Flow Diagram* (DFD)

*Data Flow Diagram* (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. Simbol yang digunakan pada DFD diberikan pada gambar 2.7.

| Notasi Yourdon/ DeMarco | Notasi Gane & Searson | Keterangan |
|---|---|---|
|  |  | Simbol entitas eksternal/ terminator, menggambarkan asal atau tujuan data di luar sistem |
|  |  | Simbol proses, menggambarkan proses dimana aliran data masuk ditransformasikan ke aliran data keluar |
|  |  | Simbol aliran data, menggambarkan arah aliran data |
|  |  | Simbol file, menggambarkan tempat data disimpan |

**Gambar 2.7** *Simbol pada DFD* (Whitten, 2004)

# BAB III

# CARA PENELITIAN

## 3.1 Bahan Penelitian

Bahan penelitian yang digunakan dalam proses penelitian ini sebagai berikut :

### 3.1.1 Obyek Penelitian

Obyek yang diteliti dalam penelitian tentang pengamanan e-dokumen dengan menggunakan metode *hybrid :* biometrik tandatangan dan DSA yaitu biometrik tandatangan *offline* dan e-dokumen. Tandatangan *offline* adalah *scan* tandatangan manual atau tandatangan beserta keterangan, misalnya nomer dokumen, tempat, tanggal, nama atau keterangan lainnya. Jenis file e-dokumen yang digunakan dalam penelitian ini berupa file dengan ekstensi : php, mp3, pptx, ppt, html, pdf, txt, bmp, sys, docx, doc, xlsx, xls, zip, exe, jpg. Sedangkan tandatangan *offline* yang digunakan dalam penelitian ini berupa *scan* tandatangan manual dalam file dengan ekstensi jpg.

### 3.1.2 Metode Pengumpulan Data

Metode pengumpulan data dalam penelitian ini adalah sebagai berikut :

1. Observasi

   Merupakan metode pengumpulan data dengan cara melakukan pengamatan secara langsung pada obyek yang diteliti yaitu biometrik tandatangan *offline* dan e-dokumen yang perlu diberi tandatangan *digital.* .

2. Studi Pustaka

   Merupakan metode pengumpulan data dengan cara mengumpulkan data-data dari berbagai sumber yang mendukung penelitian baik itu dari buku, jurnal ilmiah, makalah prosiding maupun artikel lainnya yang mendukung penelitian. Hasil dari studi pustaka berupa teori dan perkembangan terkini mengenai kriptosistem tandatangan *digital* dan teori pendukung lainnya.

**3.2 Alat Penelitian**

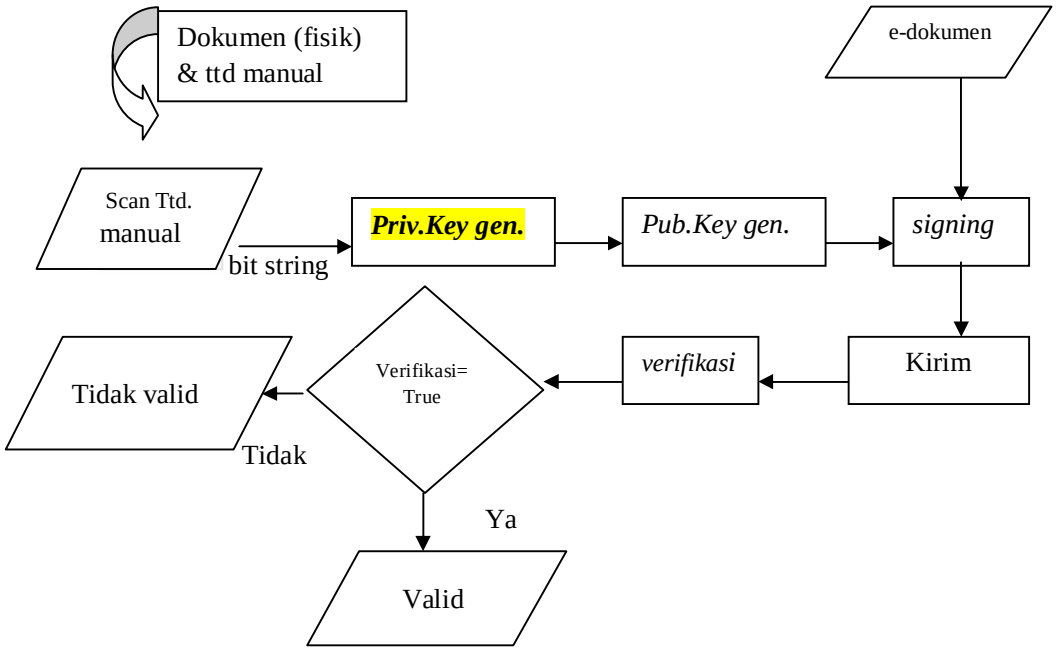
Alat penelitian yang digunakan dalam proses penelitian ini sebagai berikut :

1. Perangkat Keras berupa seperangkat komputer dengan spesifikasi Pentium (R) Dual Core CPU T4200 @ 2.00 GHz 2.00 GHz, 1.87 GB of RAM.
2. Perangkat Lunak berupa Microsoft Windows XP dan Ms. Visual Studio 2008 khususnya VB.NET, dan OpenSSL.

**3.3 Jalan Penelitian**

Adapun jalan penelitian aplikasi keamanan e-dokumen dengan metode *hybrid :* biometrik tandatangan dan DSA dapat dilihat pada gambar 3.1. Pada gambar 3.1 dapat dijelaskan bahwa pada hasil *print out* dokumen (dokumen fisik) yang telah diberi tandatangan oleh s*igner,* dengan proses *scanning*, diambil tanda tangan manual tersebut untuk diproses menjadi kode string. Dari kode string tersebut digunakan sebagai nilai SEED untuk membangkitkan (*Key generation*) parameter dan sepasang kunci yaitu kunci privat dan kunci publik. Kunci publik diturunkan dari kunci privat yang didapat. Sepasang kunci tersebut digunakan pada pembangkitan tandatangan *digital* (*signing*). E-dokumen, kunci publik dan

tandatangan *digital* selanjutnya dikirim via internet yaitu sebagai file lampiran dalam *e-mail*. Setelah *e-mail* diterima pihak *verifer*, kemudian dilakukan proses verifikasi terhadap file lampiran tersebut. Pada hasil proses verifikasi akan menampilkan hasil verifikasi *valid* berarti pesan masih asli dan e-dokumen dikirim oleh pengirim/ *signer* sebenarnya atau tidak *valid*. Sedangkan perbedaan proses pembangkitan kunci yang digunakan pada *paper* Hao dan Chan (2002), publikasi FIPS PUB 186-3 (2009) dan pada penelitian ini diberikan pada tabel 3.1.




**Gambar 3.1**. *Alur Proses Tandatangan Digital dengan Metode Hybrid : Biometrik Tandatangan dan DSA*

**Tabel 3.1** *Perbedaan Proses Pembangkitan Kunci pada Penelitian ini dengan Sebelumnya*

| Hao dan Chan (2002) | FIPS PUB 186-3 (2009) | Pada Penelitian ini |
|---|---|---|
| Pembangkitan kunci privat x, dilakukan dengan menghitung SHA-01 dari kode string tandatangan *online.* | Pembangkitan kunci privat x, dilakukan dengan nilai *seedkey* dengan panjang $\geqslant$ N = panjang bilangan prima yang dibangkitkan melalui bilangan *random* (satuan bit). | Pembangkitan kunci privat x, dilakukan dengan mengambil kode string tandatangan *offline* sebagai nilai *seed* untuk membangkitkan parameter, kunci privat dan publik yang dinamis. |

### 3.3.1 Pengembangan Sistem dengan Metode Waterfall

Pada penelitian ini dibangun suatu sistem dalam bentuk perangkat lunak tandatangan *digital.* Sedangkan metode yang digunakan untuk membangun perangkat lunak tersebut adalah metode *waterfall* yaitu suatu metode pengembangan sistem yang bersifat sekuensial dan terdiri dari 5 tahap yang saling terkait dan mempengaruhi.

Tahapan tersebut meliputi :

1. Analisa kebutuhan

   Pada tahap analisa kebutuhan dilakukan pengumpulan informasi proses tandatangan digital dan daftar kebutuhan *user* secara lengkap. Hal ini dapat diperoleh dengan menyusun *scope of work software* yang akan dibuat beserta

perangkat lunak yang dibutuhkan. Pada penelitian ini *scope of work software* yang dibuat meliputi :

a) Proses penyimpanan tandatangan manual *offline*.
b) Proses hash pada bit string dari *scanning* tanda tangan manual.
c) Proses pembangkitan kunci privat dan kunci publik dengan panjang kunci 512 sampai 1024 bit, sesuai standar FIPS (*Federal Information Processing Standart*).
d) Proses penandatanganan digital/ *signature*.
e) Proses penyimpanan *signature* e-dokumen (tandatangan digital).
f) Proses penyimpanan *public* dan *privat key*.
g) Proses verifikasi tandatangan digital.

Sedangkan kebutuhan perangkat lunak bahasa pemrograman yang dibutuhkan pada penelitian ini adalah yang memiliki kemampuan sebagai berikut :

a) Dapat melakukan proses komputasi dengan cepat, termasuk penanganan *input* dan *output*.
b) Memiliki ketelitian yang tinggi
c) Bahasa pemrograman yang didukung oleh *vendor* dan komunitas sehingga memudahkan pengembangan sistem di kemudian hari.

Dari kebutuhan kemampuan tersebut bahasa pemrograman VB.NET pada Ms. Visual Studio 2008 dapat memenuhi semua kriteria, sehingga digunakan untuk membangun sistem pada penelitian ini.

2. Perancangan sistem

Desain/ perancangan sistem merupakan tahap penyusunan proses, data, aliran proses dan hubungan antar data yang paling optimal untuk menjalankan program yang dibuat sesuai dengan hasil analisa kebutuhan. Dokumentasi yang dihasilkan dari tahap desain sistem ini yaitu *System Flow/ Flow Chart* dan *Data Flow Diagram (DFD). System Flow* merupakan bagan aliran dokumen dari satu bagian ke bagian lain baik secara manual maupun melalui sistem. *Data Flow Diagram* adalah diagram yang menunjukan aliran data di antara pengguna, proses dan database yang terkait dengan *software*. *Entity Relationship Diagram* menunjukan bagaimana data dan informasi software di simpan di dalam database beserta dengan hubungan antar data. Sedangkan pada desain *software* mempertimbangkan aspek *user* dengan membuat desain *interface* berbasis GUI (*Graphical User Interface*) yaitu menggunakan sistem menu *pulldown* agar user mudah memanggil, memproses dan menyimpan hasil dari semua fasilitas yang ada. Tahap ini dijelaskan lebih lanjut pada subbab 3.3.2.

3. Penulisan kode (*coding*) program.

   Penulisan kode program merupakan tahap penerjemahan desain sistem yang telah dibuat ke dalam bentuk perintah-perintah yang dimengerti komputer. Pada penelitian ini penulisan kode program dilakukan secara modular sesuai dengan pembagian blok-blok program pada tahap perancangan sistem, dengan mempergunakan VB.NET 2008 dan OpenSSL. Sedangkan kode program dapat dilihat pada lampiran 1 sampai 4.

4. Pengujian program.

   Pengujian *software* dilakukan untuk memastikan bahwa *software* yang dibuat telah sesuai dengan desainnya dan semua fungsi dapat dipergunakan dengan baik tanpa ada kesalahan. Pengujian *software* dilakukan untuk memastikan bahwa kerangka/ skenario pengujian *software* dibuat dengan lengkap meliputi semua proses, kebutuhan dan pengendalian yang ada di dalam dokumen analisa kebutuhan dan desain sistem. Pada penelitian ini pengujian program dilakukan pada beberapa jenis file e-dokumen sebagai masukan, variasi kunci, pembuatan dan verifikasi tandatangan *digital*.

5. Penerapan program.

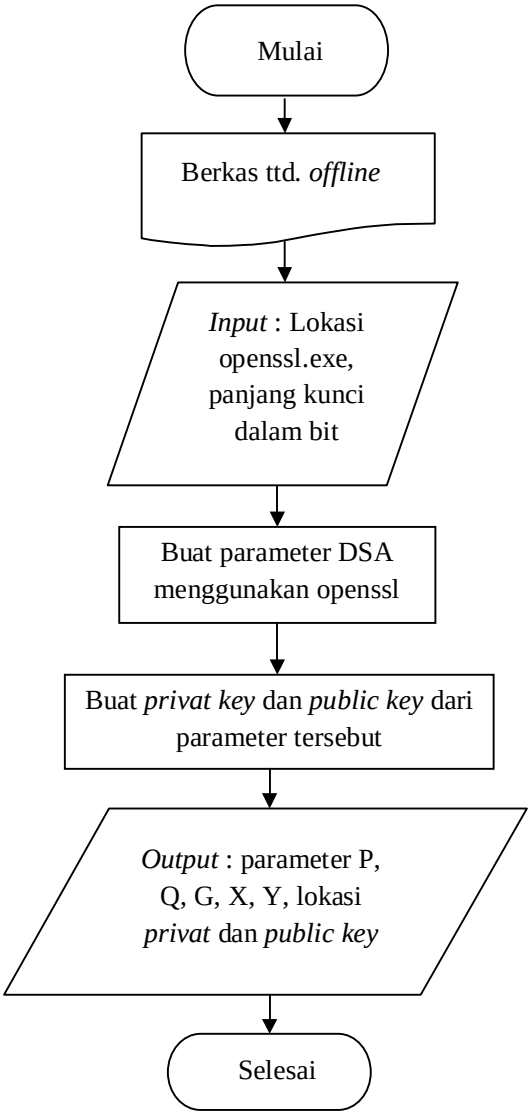
   Penerapan program merupakan tahap dimana peneliti menerapkan/ meng-*install software* yang telah selesai dibuat dan diuji ke dalam lingkungan Teknologi Informasi perusahaan dan memberikan pelatihan kepada pengguna di perusahaan juga memberikan buku *User Manual* –nya. Namun tahap ini tidak dilakukan karena keterbatasan waktu penyelesaian penelitian ini.

### 3.3.2 Perancangan Aplikasi Kriptografi dengan Metode *Hybrid* : Biometrik Tandatangan dan DSA (*Digital Signature Algorithm*)

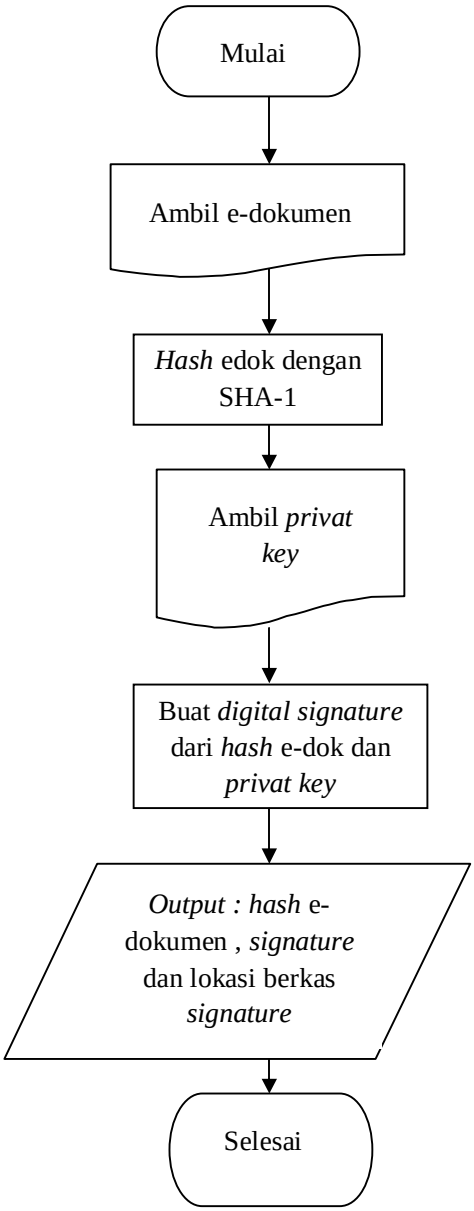
Pada tahap perancangan aplikasi dengan metode *hybrid :* Biometrik Tandatangan dan DSA meliputi penyusunan *Flow Chart, Data Flow diagram (DFD),* perancangan *interface* sebagai deskripsi dari tahap penyusunan proses, data, aliran proses dan hubungan antar data *input* dan *output, System Flow* dan *Data Flow Diagram (DFD).*

### 3.3.2.1 *Flow Chart*

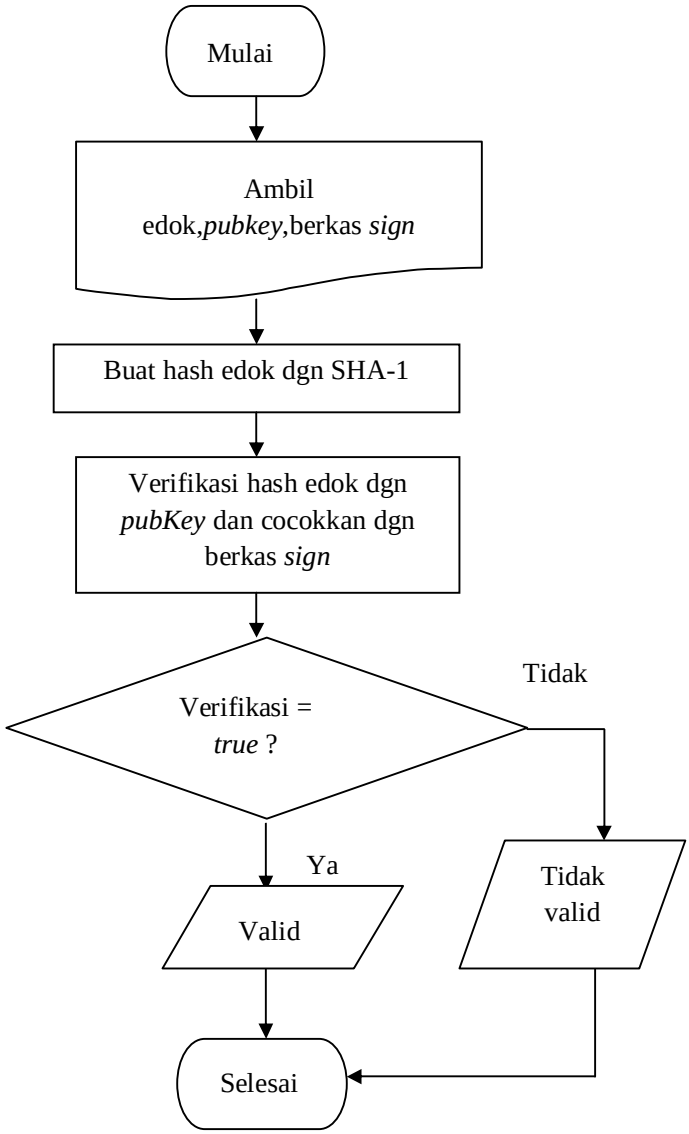
*Flow Chart* pada metode metode *hybrid :* Biometrik Tandatangan dan DSA disusun sesuai tahapan proses yaitu terdiri dari *Create key Flow Chart, Signing Flow Chart,* dan *Verify Flow Chart* masing-masing diberikan pada gambar 3.2, 3.3 dan 3.4.




**Gambar 3.2** *Create Key Flowchart*



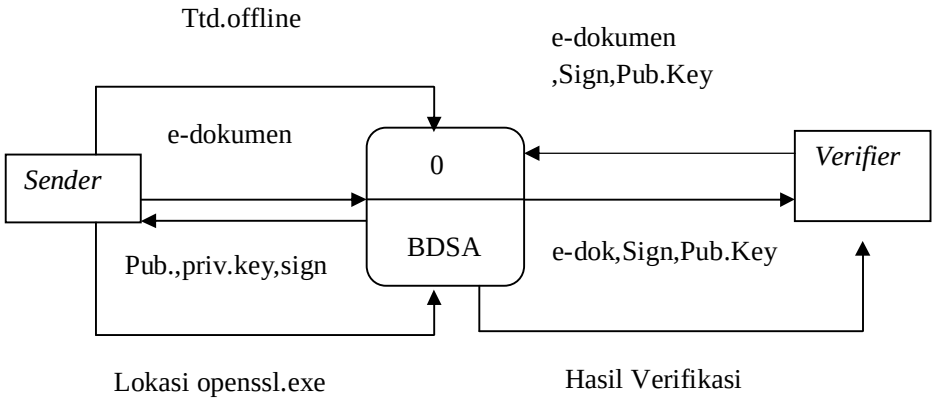

**Gambar 3.3** S*igning Flowchart*




**Gambar 3.4** *Verify Flow Chart*

### 3.3.2.2 DFD (*Data Flow Digram*)

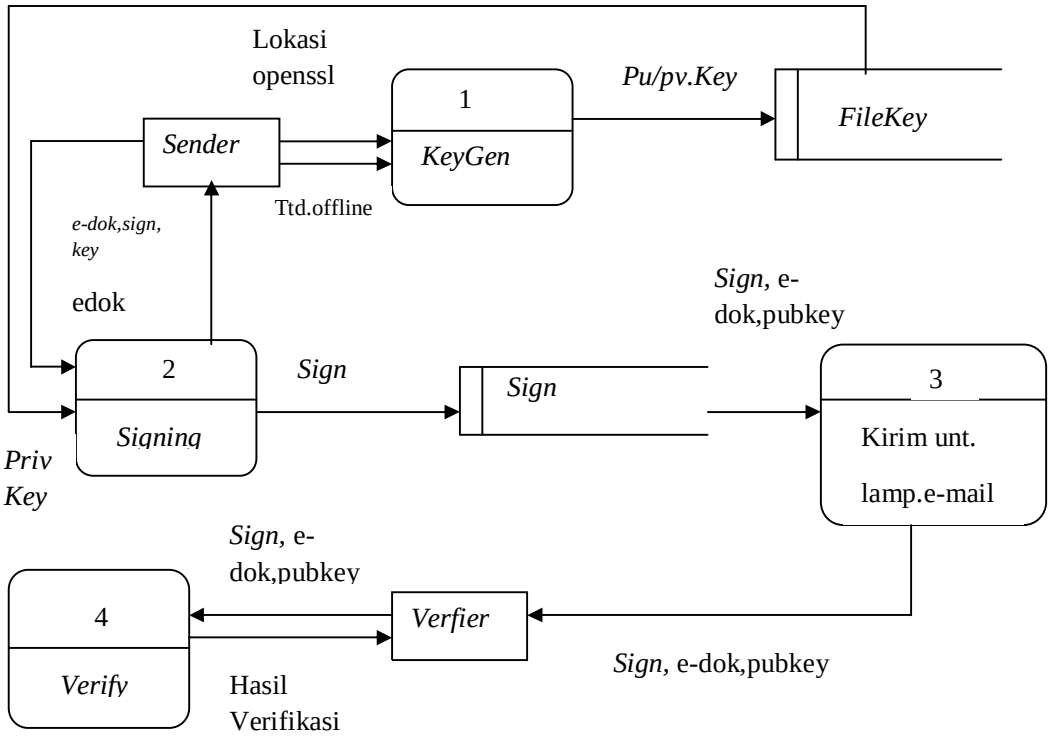
DFD pada metode *hybrid :* Biometrik tandatangan dan DSA terdiri dari diagram konteks (DFD level 0) dan DFD level 1 yang masing-masing diberikan pada gambar 3.5 dan 3.6.




**Gambar 3.5** *Diagram Konteks (DFD level 0)*

Pada diagram konteks terdapat dua entitas luar yaitu *sender* atau *signer* dan *verifier* yang berinteraksi dengan satu proses yaitu BDSA. *Sender* meng-*input*-kan tandatangan *offline*, file e-dokumen yang akan diberi tandatangan *digital* dan lokasi OpenSSL. Pada proses BDSA dilakukan proses pembangkitan kunci dan *signing* yang menghasilkan tandatangan *digital* (.sign) beserta kunci publik (.pub) dan privat (.priv). Kemudian e-dokumen, tandatangan *digital* (.sign) beserta kunci publik (.pub) dikirim kepada *verifier* melalui internet berupa lampiran pada *email.* Oleh *verifier* akan di-inputkan ke proses BDSA untuk verifikasi dan memberikan hasil verifikasi tersebut kepada *verifier*.

Proses BDSA selanjutnya dijabarkan menjadi proses yang lebih terinci yaitu pada DFD level 1 yang diberikan pada gambar 3.6.




**Gambar 3.6** *DFD level 1*

Pada DFD level 1 terdiri dari 4 proses yaitu proses pembangkitan kunci (*key generator*), proses pembangkitan tandatangan digital (*signing*), proses pengiriman e-mail dan proses verifikasi (*verify*). Penjelasan dari masing-masing proses sebagai berikut:

a. Proses 1 yaitu proses pembangkitan kunci (*key generator*) : *signer*/ penandatangan sebagai *sender*/ pengirim pesan memasukkan tandatangan *offline* dan lokasi openSSL ke proses 1 yaitu proses pembangkitan kunci (*key generator)*. Hasil dari proses 1 yaitu *public key* dan *privat key* yang disimpan pada file key.

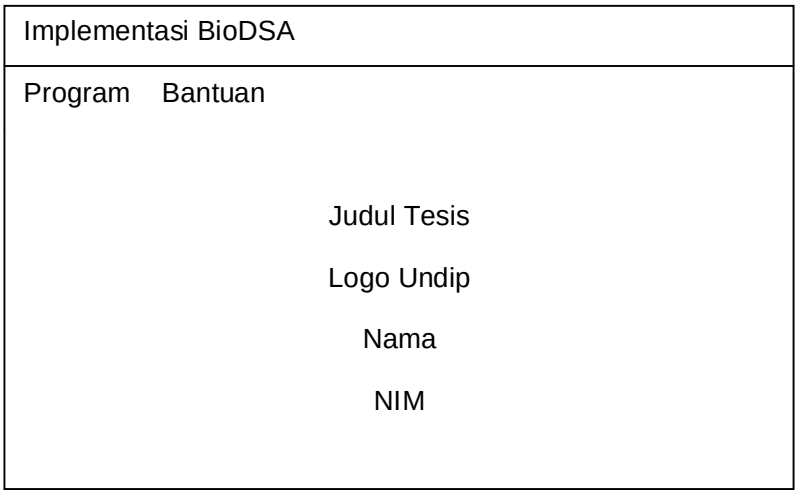
b. Proses 2 yaitu proses pembangkitan tandatangan *digital* (*signing*) : *sender* memasukkan e-dokumen dan file *privat key* ke proses 2 yaitu *signing*. Hasil dari proses 2 yaitu tandatangan *digital* (*signature*) untuk disimpan dalam file sign. Selanjutnya e-dokumen, kunci publik dan *signature* siap ditransmisikan lewat *e-mail* kepada *verifier*.

c. Proses pengiriman e-mail (proses 3) : *sender* mengirim *e-mail* beserta file lampiran (e-dok, file sign dan public key) kepada *verifier*.

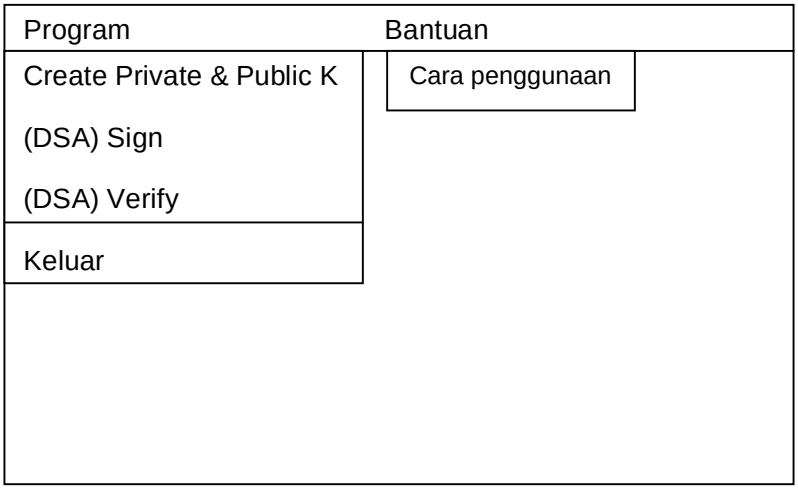
d. Proses verifikasi (*verify*) (proses 4) : setelah *verifier* menerima *e-mail* beserta file lampiran dari *sender*, file lampiran selanjutnya diverifikasi pada proses 4 yaitu diperiksa ketersediaan e-dokumen, kunci publik dan *signature* untuk proses verifikasi. Hasil proses verifikasi adalah valid jika e-dokumen yang diterima masih asli (otentik) dan dikirim oleh pihak sebenarnya atau sebaliknya yaitu tidak valid jika e-dokumen yang diterima sudah berubah (tidak otentik) dan atau dikirim bukan oleh *signer* sebenarnya.

### 3.4 Perancangan Antar Muka

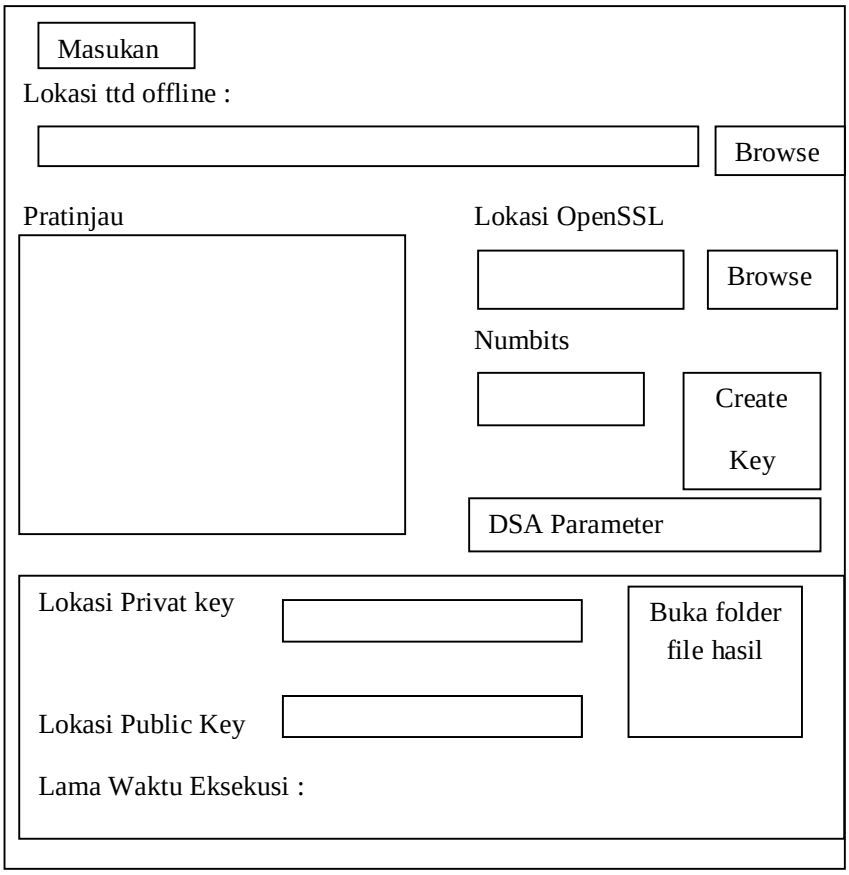
Perancangan antar muka program utama, menu dan submenu program, jendela "*Create DSA Public and Privat Key*", jendela "Tandatangani e-dokumen" dan jendela "verifikasi tandatangan *digital*" masing-masing diberikan pada gambar 3.7 sampai 3.11.



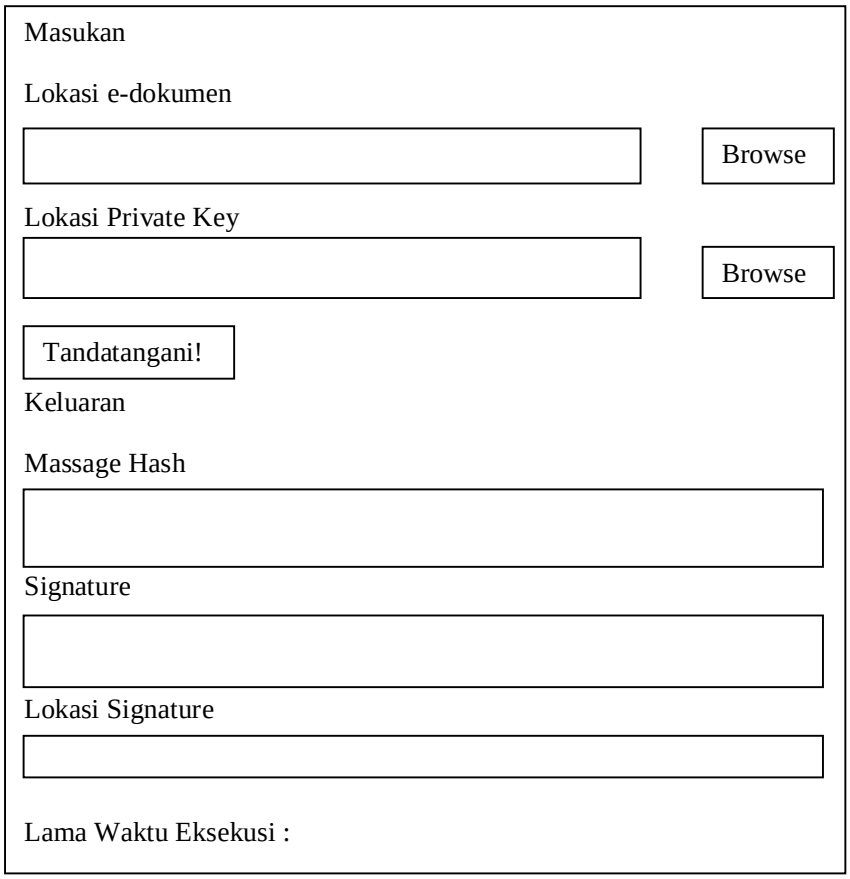

**Gambar 3.7** *Perancangan Antar Muka Program Utama*.



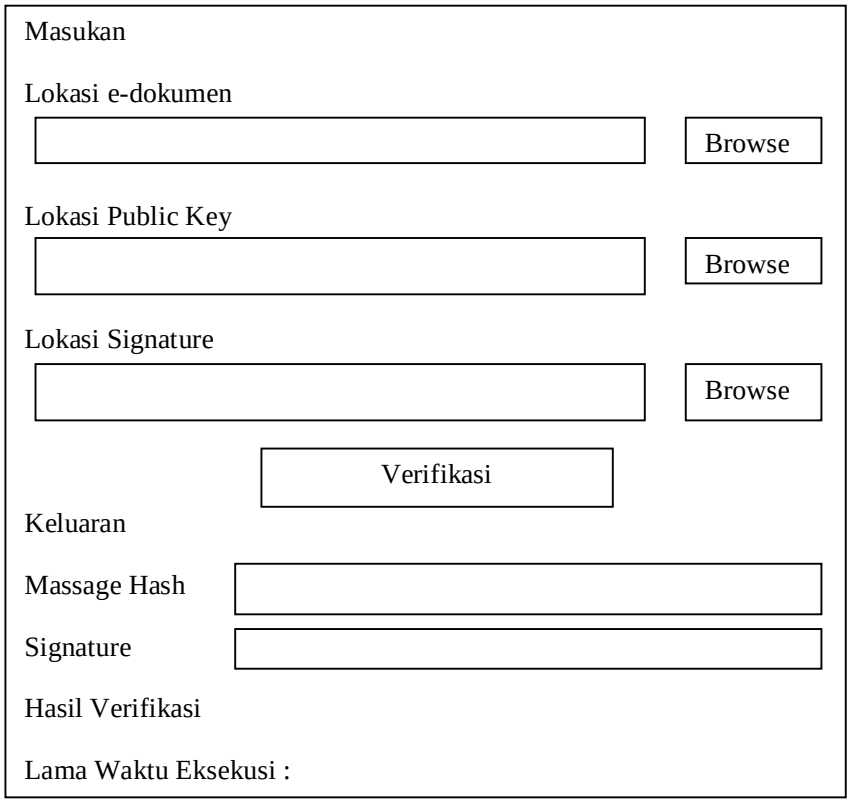

**Gambar 3.8** *Perancangan Antar Muka Menu dan Submenu Program*




**Gambar 3.9** *Perancangan Antar Muka Jendela "Create DSA Public and Privat Key"*




**Gambar 3.10** *Perancangan Antar Muka Jendela "Tandatangani e-dokumen"*




**Gambar 3.11** *Perancangan Antar Muka Jendela Verifikasi Tandatangan Digital*

# BAB IV

# HASIL PENELITIAN DAN PEMBAHASAN

Bab ini menunjukkan hasil penelitian dan pembahasan tentang pengamanan e-dokumen dengan metode *hybrid :* Biometrik tandatangan dan DSA (*Digital Signature Algorithm*).

## 4.1 Hasil Penelitian

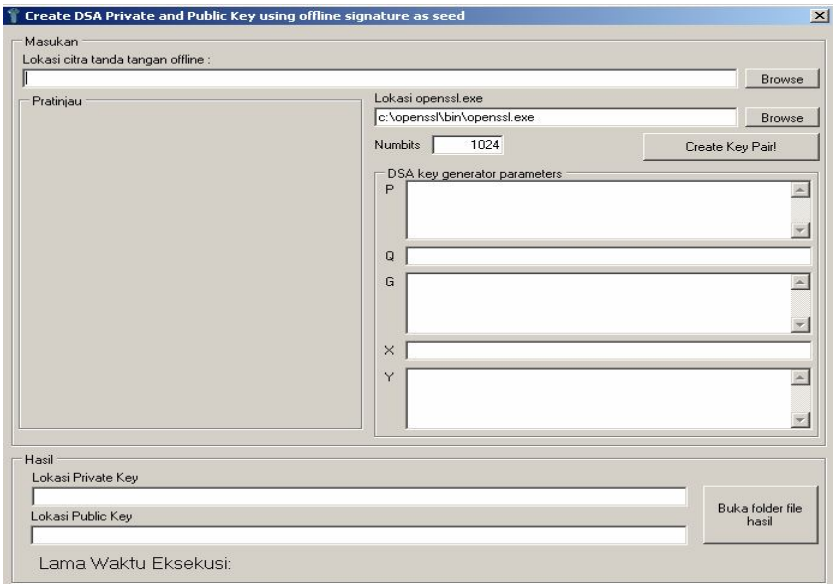
Hasil dari penelitian ini adalah perangkat lunak tandatangan *digital* sebagai implementasi dari metode *hybrid :* Biometrik tandatangan dan DSA. Perangkat lunak yang dihasilkan terdiri dari tiga proses inti yaitu proses pembuatan kunci, pembuatan tandatangan *digital* dan proses verifikasi yang masing-masing tampilan jendela untuk ketiga proses tersebut diberikan pada gambar 4.1, gambar 4.2 dan gambar 4.3. Sebagai masukan tandatangan *offline* yang digunakan pada penelitian ini, diberikan pada tabel 4.1.

**Tabel 4.1** *Tandatangan Offline yang Digunakan dalam Penelitian*

| No | Nama file | Keterangan |
|---|---|---|
| 1 | Ttd1.jpg | *Signer* sah |
| 2 | Ttd2.jpg | *Signer* tidak sah |
| 3 | Ttdars.jpg | *Signer* sah |
| 4 | Ttdby.jpg | *Signer* sah |

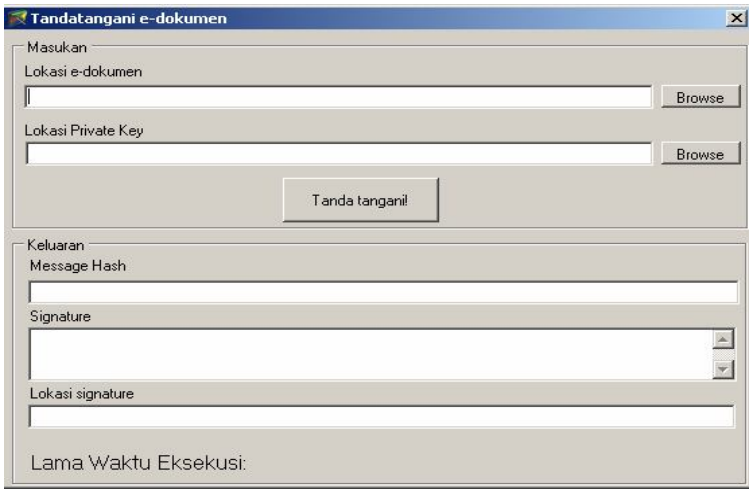
Gambar 4.1 menunjukkan jendela "*Create DSA Private and Public Key using offline signature as seed*" untuk membuat kunci privat dan kunci publik

dengan masukan adalah tandatangan *offline*, panjang bit kunci yang dipakai antara 512 sampai 1024 bit dan lokasi openSSL, jika bukan berada di lokasi *default*-nya. Sedangkan sebagai keluaran adalah parameter DSA, lokasi kunci privat dan publik.



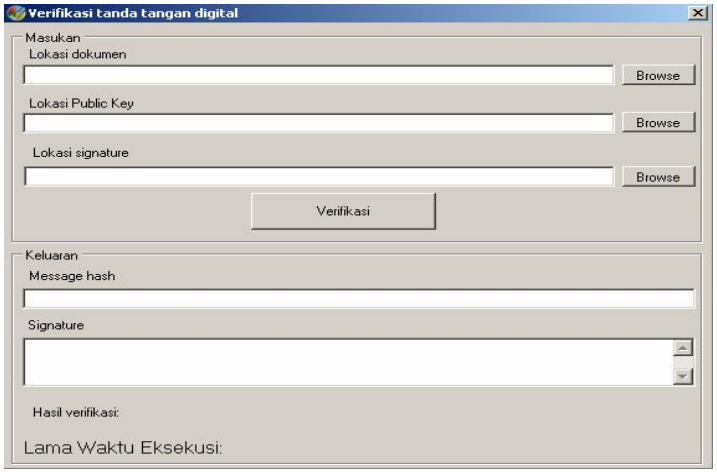

**Gambar 4.1** *Tampilan Jendela "Create DSA Private and Public Key using offline signature as seed"*

Gambar 4.2 menunjukkan jendela 'Tandatangani e-dokumen" untuk membuat tandatangan *digital* dari e-dokumen. Sebagai masukan pada proses pembuatan tandatangan *digital* adalah lokasi e-dokumen dan lokasi kunci privat. Tandatangan *digital* yang dihasilkan, ditulis pada file *signature* dengan ekstensi sig. Sebagai keluarannya adalah *massage hash* e-dokumen, *signature* dan lokasi *signature*.




**Gambar 4.2** *Tampilan Jendela "Tandatangani e-dokumen"*

Gambar 4.3 menunjukkan jendela "verifikasi tandatangan *digital*" untuk melakukan verifikasi atas keutuhan/ keaslian e-dokumen dan keabsahan *signer*. Sebagai masukan adalah lokasi e-dokumen, lokasi kunci publik dan lokasi *signature.* Sedangkan keluarannya adalah massage hash e-dokumen, *signature* dan hasil verifikasi.




**Gambar 4.3** *Tampilan Jendela "Verifikasi Tandatangan Digital"*

Keunggulan dari perangkat lunak yang dihasilkan pada penelitian ini adalah sebagai konsep baru pada penerapan tandatangan *digital* yaitu :

1. Sebagai salah satu solusi dalam aspek manajemen kunci dengan membangkitkan kunci secara dinamis. Hal ini karena dengan masukan tandatangan *offline* yang sama dan panjang kunci juga sama, dapat membangkitkan kunci yang berbeda. Sehingga sifat pemakaian kunci adalah sekali pakai untuk satu tandatangan *digital* e-dokumen.
2. Dapat memenuhi kebutuhan otorisasi beberapa *signer* pada satu e-dokumen. Hal ini karena pada satu e-dokumen dapat diberi tandatangan *digital* oleh lebih dari satu *signer*.

**4.2 Pembahasan**

Pada implementasinya perangkat lunak atau program ini harus sudah di-*instal* (semua file yang ada di folder "bin\Release\" di-*copy*) di komputer pengguna *signer* dan *verifier*. Selain itu komputer pengguna juga harus sudah ter-*install* .NET framework minimal versi 3.5, jika sudah terinstal visual studio 2008, otomatis .NET framework tersebut ter-*instal*.

Pembahasan penelitian ini dilakukan dengan hasil simulasi terhadap satu dan dua *signer*. *Signer* adalah pihak pembuat dan atau pengirim via *e-mail* : e-dokumen, tandatangan *digital*, dan kunci publik, sedangkan *verifier* adalah pihak penerima *e-mail* yang melakukan proses verifikasi terhadap file lampiran yang dikirim via *e-mail* dari *signer*.

Simulasi dilakukan dengan e-dokumen yang utuh/ otentik dan yang telah berubah, pasangan kunci yang dipakai dari *signer* yang sah/ sebenarnya dan

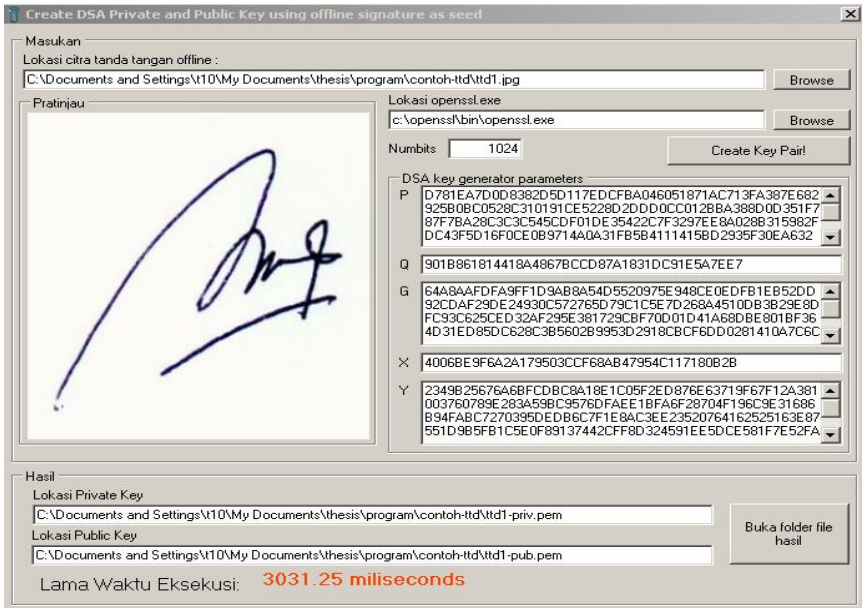
*signer* tidak sah/ bukan sebenarnya juga dengan kunci dan atau *signature* yang dirusak pihak *attacker*.

### 4.2.1 Simulasi Tandatangan Digital dengan Satu *Signer.*

Berikut ini disimulasikan tandatangan *digital* dengan mengambil satu tandatangan *offline* yaitu ttd1.jpg sebagai penandatangan (*signer*) sah e-dokumen.

#### 4.2.1.1 Proses Pembuatan Kunci Privat dan Publik serta Proses *Signing*

Gambar 4.4 menampilan hasil proses membuat kunci dengan masukan ttd1.jpg dan panjang kunci 1024 bit.



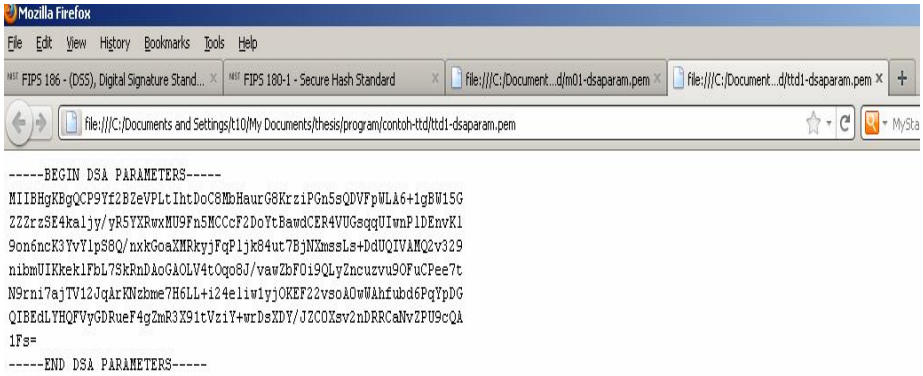

**Gambar 4.4** *Contoh Tampilan Proses Membuat Sepasang Kunci Privat dan Publik dari Ttd1.jpg..*

Sebagai kunci publik adalah Y dan kunci privat adalah X, sedangkan P, Q dan G adalah nilai parameter DSA. Tampilan nilai X dan Y pada "*DSA key generator parameter*" adalah pada format hexa yang dijamin dengan potongan kode program pada lampiran 2 yaitu :

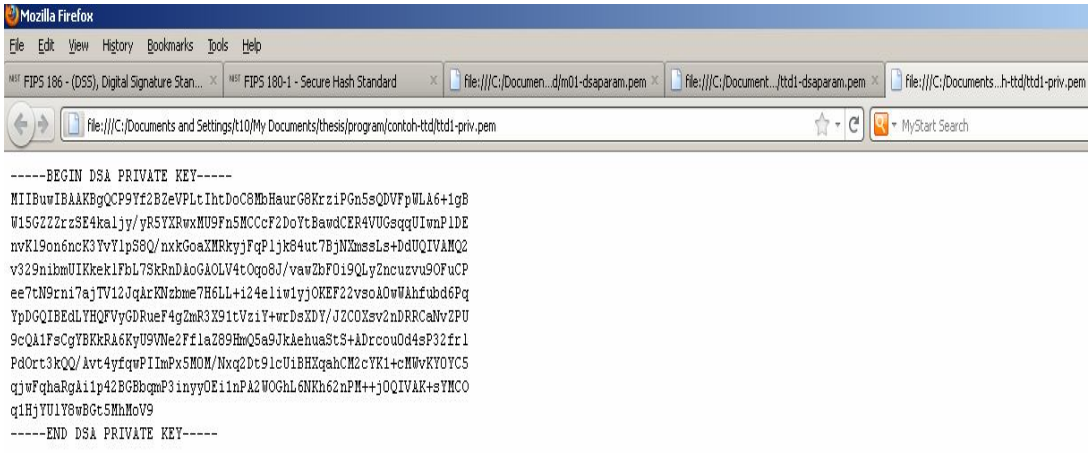
```
txtP.Text = dsa.HexP
txtQ.Text = dsa.HexQ
txtG.Text = dsa.HexG
txtX.Text = dsa.HexX
txtY.Text = dsa.HexY
```

Sedangkan file yang dihasilkan yaitu kunci publik (*-pub.pem) dan kunci privat (*-priv.pem) adalah pada format pem yaitu *mode base* 64 ( Chandra; Messier; dan Viega, 2001). Hal ini untuk menunjukkan adanya perbedaan hasil pada tampilan keluaran nilai X, Y dan isi file kunci privat dan kunci publik hanya pada formatnya saja sedangkan nilainya sama. Selanjutnya pasangan kunci tersebut siap digunakan untuk menandatangani/ membuat tandatangan *digital* e-dokumen.

Hasil keluaran pada proses "*create key pair*" adalah file parameter DSA (*-dsaparam.pem), kunci privat (*-priv.pem) dan kunci publik (*-pub.pem) yang tersimpan di lokasi yang sama dimana tandatangan manual yang digunakan untuk membuat kunci disimpan. Contoh tampilan sebagai hasil proses pembuatan kunci dari ttd1.jpg adalah parameter DSA, kunci privat dan publik yang dihasilkan berdasarkan gambar 4.4, masing-masing ditunjukkan pada gambar 4.5, 4.6 dan 4.7.




**Gambar 4.5** *Contoh Tampilan Parameter DSA yang Dihasilkan dari Ttd1.jpg dalam Mode Base 64*

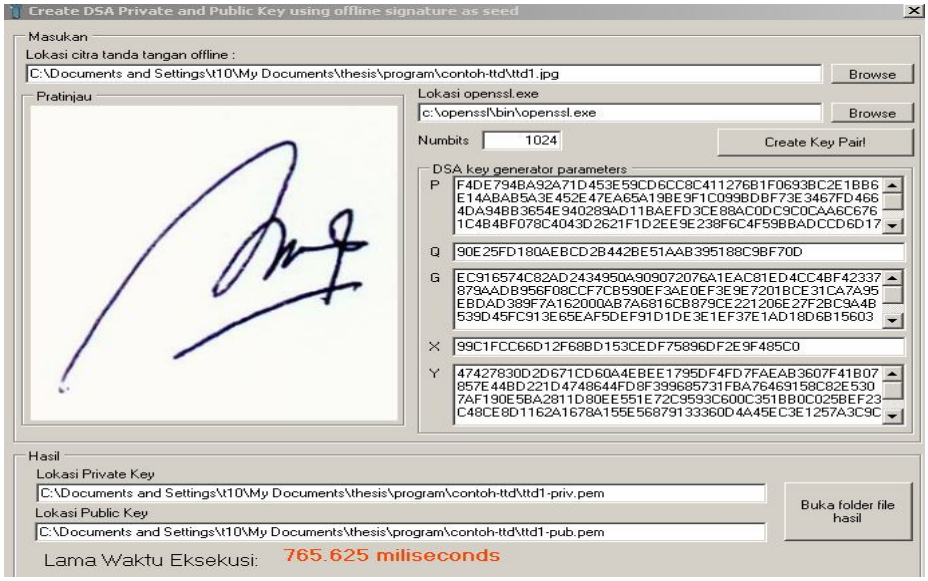



**Gambar 4.6** *Contoh Tampilan Kunci Privat yang Dihasilkan dari Ttd1.jpg dalam Mode Base 64*

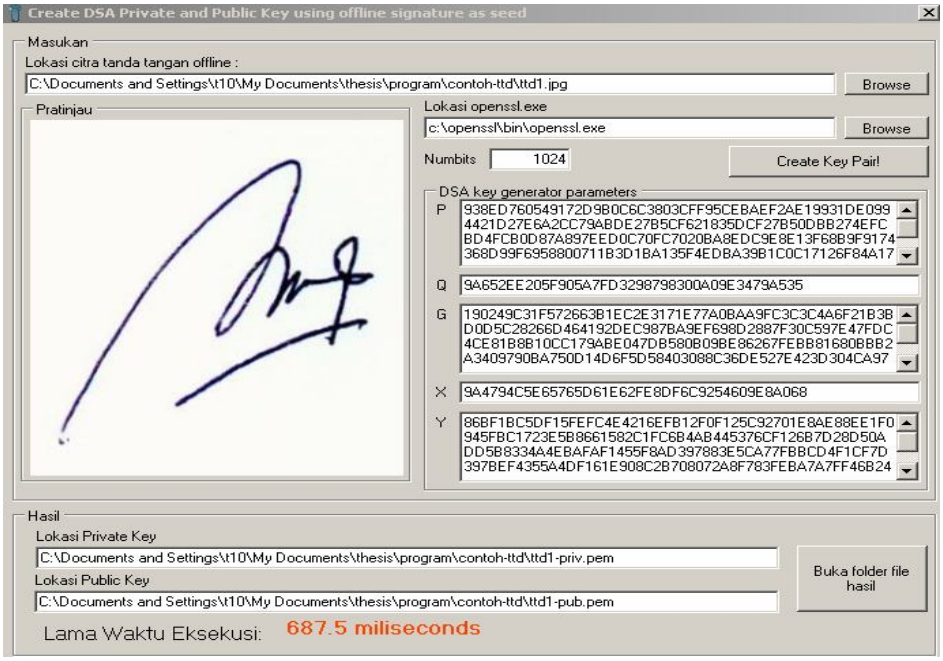

**Gambar 4.7** *Contoh tampilan kunci publik (pada mode base 64) dari ttd1-pub.pem.*

Pada masukan yang sama yaitu ttd1.jpg dan panjang kunci 1024 dapat menghasilkan kunci publik dan privat yang berbeda. Hal ini terjadi karena hasil/ nilai bilangan prima P, Q, G adalah acak. Sebagai contoh dari masukan ttd1.jpg dan panjang kunci 1024 bit, selain hasil kunci pada gambar 4.6 dan 4.7 dapat menghasilkan kunci yang lain yaitu kunci publik 1 dan 2, masing-masing (P1,Q1,G1,Y1) dan (P2,Q2,G2,Y2) dan kunci privat 1 dan 2 masing-masing

(P1,Q1,G1,X1) dan (P2,Q2,G2,X2) yang ditunjukkan masing-masing pada gambar 4.8 dan 4.9.




**Gambar 4.8** *Kunci Privat dan Publik (1) yang Dihasilkan dari Ttd1.jpg*.




**Gambar 4.9** *Kunci Privat dan Publik (2) yang Dihasilkan dari Ttd1.jpg*.

Perbedaan pasangan kunci (1) dan (2) yang dihasilkan gambar 4.8 dan 4.9 diberikan pada tabel 4.2.

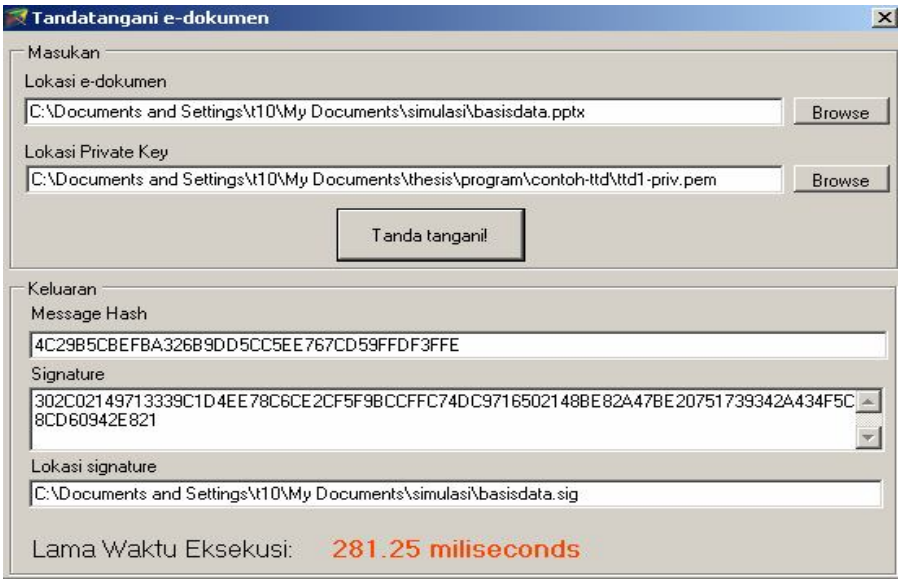
**Tabel 4.2** *Perbedaan kunci yang dihasilkan berdasarkan gambar 4.8 dan 4.9*

| P1 | P2 | Q1 | Q2 | G1 | G2 | X1 | X2 | Y1 | Y2 |
|---|---|---|---|---|---|---|---|---|---|
| F4D… | 939… | 90E… | 9A6… | EC9… | 190… | 99C… | 9A4… | 474… | 86B… |

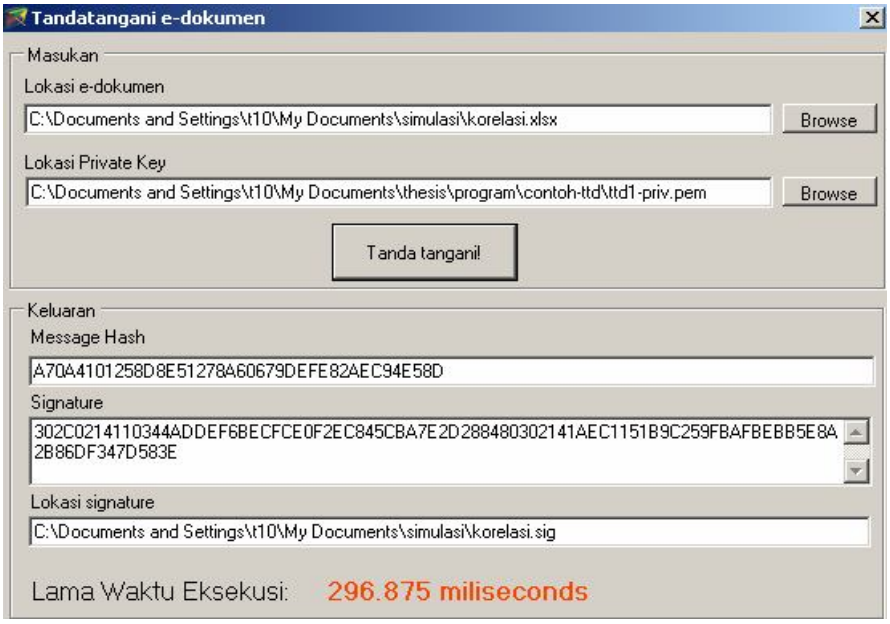
Berdasarkan tabel 4.2 dapat dijelaskan bahwa dari tandatangan yang sama (satu tandatangan *offline*) dengan panjang kunci juga sama, dapat digunakan untuk membuat pasangan kunci yang berbeda sehingga dapat memenuhi kemungkinan tandatangan seseorang yang tidak unik artinya ada beberapa orang yang mempunyai tandatangan yang sama/ hampir sama tetapi kunci yang dihasilkan berbeda. Selain itu hasil pembuatan kunci dari satu masukan yang sama dapat menghasilkan lebih dari satu pasang kunci yang berbeda, artinya kapanpun diperlukan dapat menghasilkan kunci yang berbeda dari satu masukan yang sama. Hal ini menunjukkan konsep baru pada pembangkitan kunci yang bersifat dinamis, sehingga merupakan salah satu solusi pada aspek manajemen kunci. Jadi sifat penggunaan kunci adalah sekali pakai artinya setiap kali diperlukan dapat membuat pasangan kunci yang baru/ berbeda walaupun dari masukan "*Create Pair Key*" yang sama. Selain itu dapat pula menggunakan pasangan kunci yang sama/ yang sudah ada beberapa kali untuk membuat tandatangan *digital* beberapa e-dokumen seperti pada konsep penggunaan kunci di paper Hao dan Chan (2002) dan FIPS PUB 186-3 (2009). Keputusan menggunakan kunci sekali pakai atau beberapa kali pakai ada di pihak pengguna, namun dari sisi

keamanan jauh lebih baik jika menggunakan kunci satu kali pakai karena sangat perlu kunci diubah jauh sebelum ia dapat ditemukan dengan cara *exhaustive search* (Munir, 2006).

Pada proses *signing* atau pembuatan tandatangan *digital*, dalam penelitian ini disebut *signature*, satu pasang kunci dapat digunakan untuk membuat *signature* dari satu e-dokumen (kunci sekali pakai) atau lebih dari satu e-dokumen (kunci beberapa kali pakai). Gambar 4.10 dan 4.11 untuk menunjukkan masukan lokasi *privat key* dan *numbit* yang sama masing-masing untuk menandatangani e-dokumen basisdata.pptx dan korelasi.xlsx. Keduanya menggunakan masukan lokasi *private key* (file kunci privat yang sama) yaitu ttd1-priv.pem dan panjang kunci 1024 bit.




**Gambar 4.10** *Hasil Signature dari Basisdata.pptx dengan Kunci Ttd1-priv.pem*



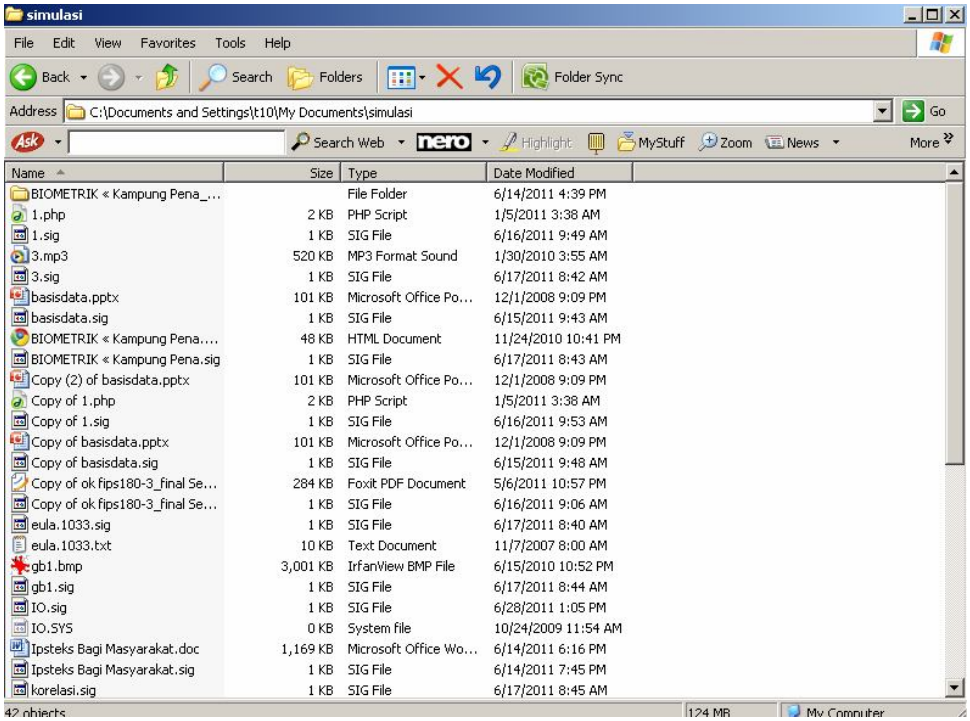

**Gambar 4.11** *Hasil Signature dari Korelasi.xlsx dengan Kunci Ttd1-priv.pem*

Sebagai contoh lain pada gambar 4.12a dan 4.12b, semua file *signature* (*.sig) dibuat dengan satu file kunci privat yaitu ttd1-priv.pem. Sedangkan pada penelitian ini, file e-dokumen yang sudah diberi *signature* berdasarkan gambar 4.12a dan 4.12b diberikan pada tabel 4.3.

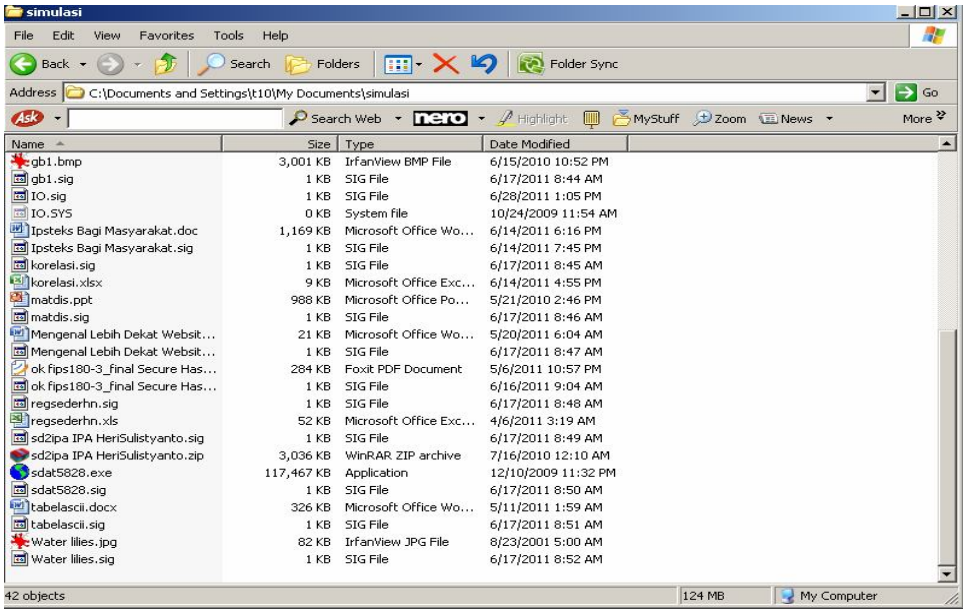
Pada penelitian ini ukuran file e-dokumen yang sudah ditandatangani yang ditunjukkan pada gambar 4.12, minimal 0 KB pada file IO.SYS sampai 117.467 KB pada file sdat5828.exe.

**Tabel 4.3** *File E-dokumen yang Sudah Diberi Signature Berdasarkan Gambar 4.12a dan 4.12b*

| No. | File E-dokumen | File Signature |
|---|---|---|
| 1 | 1.php | 1.sig |
| 2 | 3.mp3 | 3.sig |
| 3 | Basisdata.pptx | Basisdata.sig |
| 4 | Copy(2) of basisdata.pptx | Copy(2) of basisdata.sig |
| 5 | Biometrik<kampungPena.html | Biometrik<kampungPena.sig |
| 6 | Copy of 1.php | Copy of 1.sig |
| 7 | Copy of okfips 180-3 final.pdf | Copy of okfips 180-3 final.sig |
| 8 | Eula 1033.txt | Eula 1033.sig |
| 9 | Gb1.bmp | Gb1.sig |
| 10 | IO.SYS | IO.sig |
| 11 | Ipsteks Bagi Masyarakat.doc | Ipsteks Bagi Masyarakat.sig |
| 12 | Korelasi.xlsx | Korelasi.sig |
| 13 | Matdis.ppt | Matdis.sig |
| 14 | Mengenal lebih dekat Website.docx | Mengenal lebih dekat Website.sig |
| 15 | Ok fips 180-3 final.pdf | Ok fips 180-3 final.sig |
| 16 | Regsedhn.xls | Regsedhn.sig |
| 17 | Sd2ipa IPA HeruSulistyanto.zip | Sd2ipa IPA HeruSulistyanto.sig |
| 18 | Sdat5828.exe | Sdat5828.sig |
| 19 | Tabelascii.docx | Tabelascii.sig |
| 20 | Waterlilies.jpg | Waterlilies.sig |




**Gambar 4.12a**. *Contoh Beberapa Tipe File dan Ukurannya beserta Signature-nya*




**Gambar 4.12b**. *Lanjutan Gambar 4.12a.*

#### 4.2.1.2 Proses Verifikasi

Pada penelitian ini, disimulasikan proses verifikasi berdasarkan file lampiran yang dikirim via *e-mail*. Diasumsikan seluruh file lampiran yang dikirim *signer* diterima oleh *verifier* tanpa ada satu file –pun yang hilang (dicuri) dalam perjalanan transmisi. File yang dilampirkan berupa file e-dokumen, *signature :* (e-dokumen).sig, dan kunci publik : (*-pub).pem. File lampiran dapat dikirim bersama-sama atau terpisah. Contoh tampilan *e-mail* yang diterima *verifier* diberikan pada gambar 4.13. Pada gambar 4.13 file lampiran berupa file e-dokumen adalah Ipstek Bagi Masyarakat.doc, file *signature* adalah Ipstek Bagi Masyarakat.sig dan file kunci publik adalah ttd1-pub.pem.


**Gambar 4.13** *Contoh E-mail yang Diterima Verifier Beserta File Lampiran (yang Sah).*

Pada contoh kasus 2 sampai 8 diasumsikan ada *attacker* yang mengirimkan file lampiran yang tidak sah yaitu dengan isi file e-dokumen yang

sudah diubah/ tidak sah dan atau penggantian kunci publik dan atau penggantian pada *signature*. Diasumsikan pula *attacker* yang tidak merusak kunci publik, privat dan *signature,* tapi hanya mengganti dengan kunci lain dan atau *signature* lain. Kunci atau *signature* rusak didefinisikan sebagai hasil peng-*edit*-an isi file kunci atau *signature*. Pada penelitian ini file **e-dokumen sah** disimpan di C:\Documents and Setting\t10\My Documents**pengabdian**\Ipsteks Bagi Masyarakat.doc dan *signature* asli disimpan di C:\Documents and Setting\t10\My Documents**pengabdian**\Ipsteks Bagi Masyarakat.sig. File **e-dokumen tidak sah** disimpan di C:\Documents and Setting\t10\My Documents**simulasi**\Ipsteks Bagi Masyarakat.doc dan *signature* tidak asli disimpan di C:\Documents and Setting\t10\My Documents**simulasi**\Ipsteks Bagi Masyarakat.sig. **Kunci publik sah** di simpan C:\Documents and Setting\t10\My Documents\program\contoh-ttd**ttd1-pub.pem** dan **kunci publik tidak sah** di simpan C:\Documents and Setting\t10\My Documents\program\contoh-ttd**ttd2-pub.pem.** Tampilan isi kunci privat dan publik pada *mode base* 64 dari ttd1.jpg diberikan masing-masing pada gambar 4.6 dan 4.7, sedangkan untuk ttd2.jpg diberikan pada lampiran 6. Hasil verifikasi berdasarkan beberapa kemungkinan kasus yang dapat terjadi, sebagai berikut :

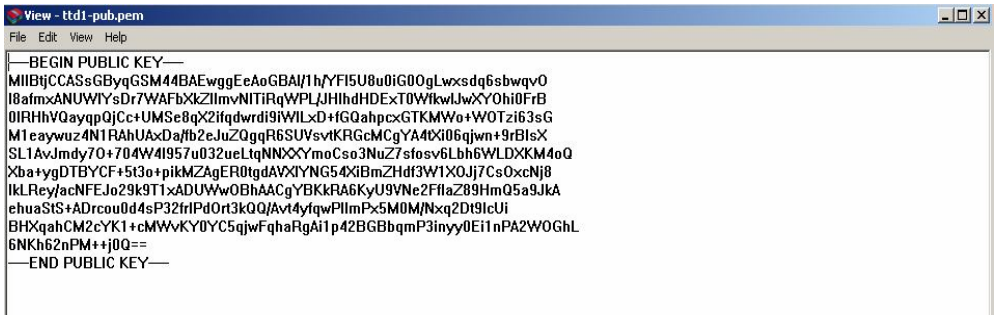
**Kasus 1** : e-dokumen sah/ otentik, *signature* dan kunci publik sah.

Gambar 4.14 menunjukkan isi sebagian halaman pertama file “Ipsteks Bagi Masyarakat.doc” yang sah/ masih utuh yang disimpan di C:\Documents and Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.doc. Gambar 4.15 menunjukkan isi file signature “Ipsteks Bagi Masyarakat.sig”

sah/sebenarnya. Gambar 4.16 menunjukkan isi file kunci publik ttd1-pub.pem sah/ sebenarnya.

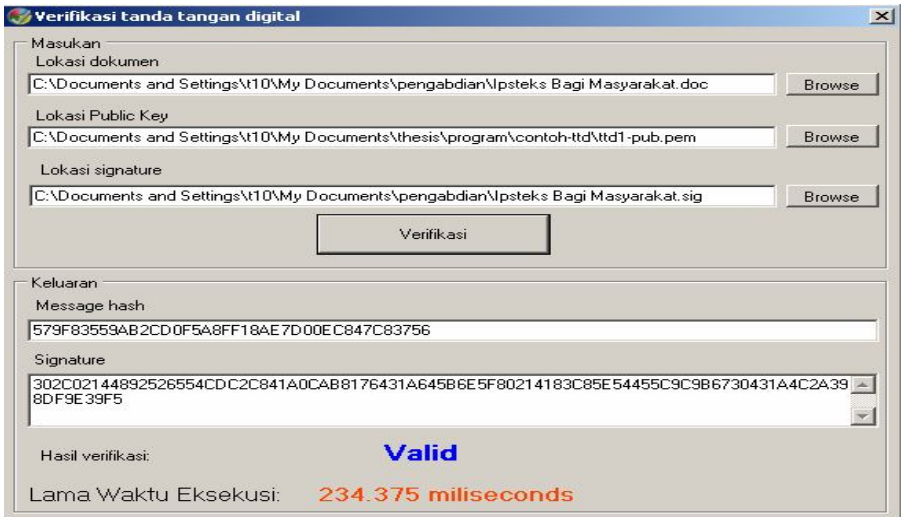

**Gambar 4.14** *Tampilan Potongan Halaman Pertama File Ipteks Bagi Masyarakat.doc yang Diunduh Verifier (File Otentik/ Sah).*


**Gambar 4.15** *Tampilan Signature yang Sah (hexa) pada File Ipteks Bagi Masyarakat.sig, yang Diunduh oleh Verifier*




**Gambar 4.16** *Contoh Tampilan Kunci Publik (mode base 64) ttd1-pub.pem Sebenarnya, yang Diunduh oleh Verifier*

Gambar 4.17 menunjukkan hasil verifikasi oleh *verifier* terhadap file lampiran yang diunduh berdasarkan gambar 4.13.

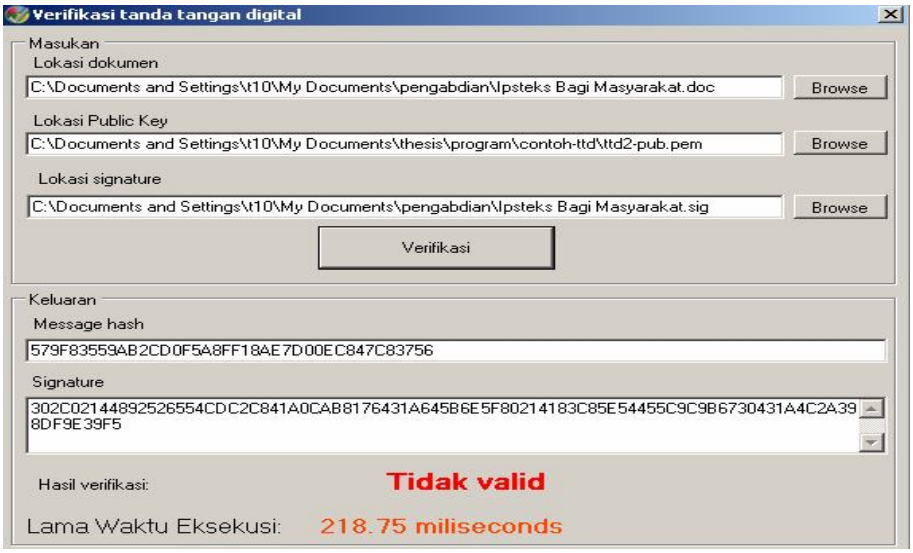



**Gambar 4.17** *Hasil Verifikasi Terhadap File Lampiran yang Diunduh Berdasarkan Gambar 4.13*

Berdasarkan gambar 4.17, jika e-dokumen masih utuh/ asli dan *signer* adalah *signer* sebenarnya (*signature* dan kunci publik yang sah) maka hasil verifikasi adalah “Valid”.

**Kasus 2**: e-dokumen otentik, *signature* yang sah dan kunci publik tidak sah.

Gambar 4.18 menunjukkan hasil verifikasi file “Ipsteks Bagi Masyarakat.doc“ yang sah/ masih utuh yang disimpan di C:\Documents and Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.doc dengan *signature* sah yang disimpan di C:\Documents and Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.sig dan kunci publik tidak sah (ttd2-pub.pem). Isi kunci publik tidak sah tersebut diberikan pada lampiran 6.




**Gambar 4.18** *Hasil Verifikasi Kasus 2*

Jika diperhatikan potongan pada gambar 4.17 dan 4.18, sebagai hasil keluaran yaitu nilai *massage hash* dan *signature* masing-masing adalah sama yang dapat dilihat pada gambar 4.19.


(a)

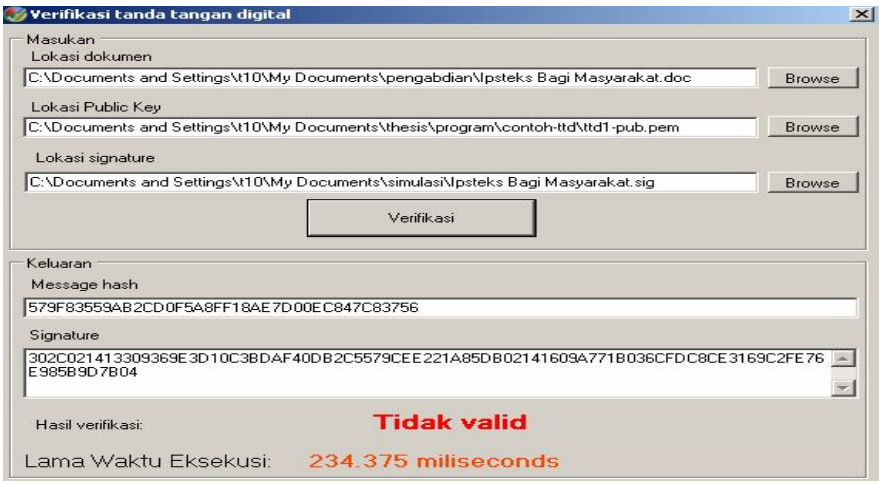

(b)

**Gambar 4.19** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.18*

Berdasarkan gambar 4.19, hasil verifikasi (a) dan (b) tidak sama. Keluaran yang dihasilkan sama karena e-dokumen sah ditunjukkan dengan nilai *message hash* yang sama dan *signature* juga sah ditunjukkan dengan nilai *signature* yang sama. Namun pada gambar 4.19 (b) hasil verifikasi adalah "Tidak valid", hal ini berarti kunci publik yang digunakan untuk mendekrip *signature* bukan pasangan kunci privat yang digunakan untuk membuat *signature* sehingga dapat disimpulkan bahwa *signer* tidak sah.

**Kasus 3**: e-dokumen otentik, *signature* yang tidak sah dan kunci publik sah

Gambar 4.20 menunjukkan hasil verifikasi file "Ipsteks Bagi Masyarakat.doc" yang sah/ masih utuh yang disimpan di C:\Documents and Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.doc dengan *signature* tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.sig dan kunci publik sah (ttd1-pub.pem).




**Gambar 4.20** *Hasil Verifikasi Kasus 3*

Jika diperhatikan potongan pada gambar 4.17 dan 4.20, sebagai hasil keluaran yaitu nilai *massage hash* yang sama dan *signature* berbeda yang dapat dilihat pada gambar 4.21.


(a)


(b)

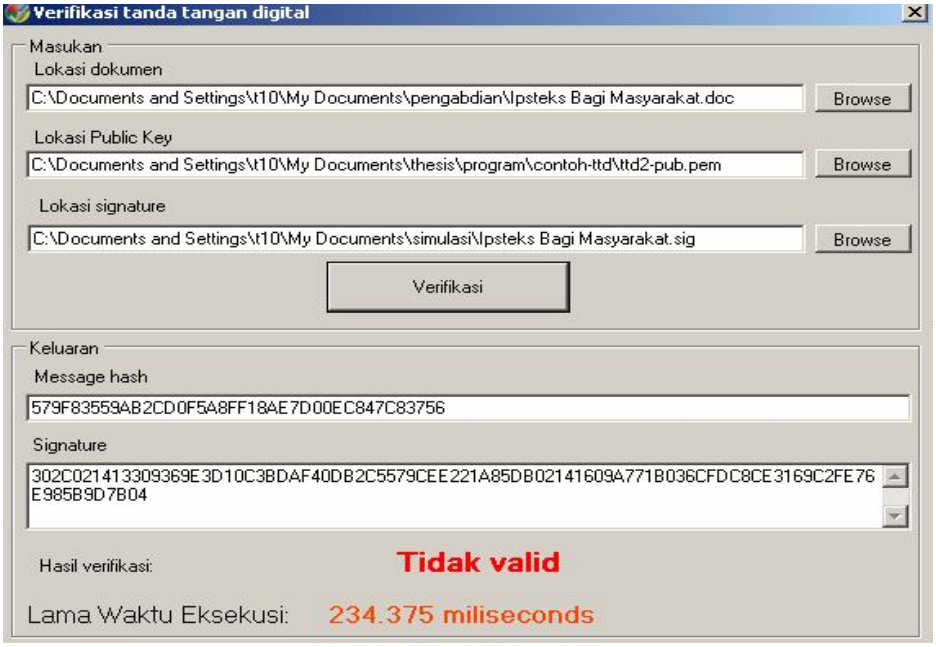
**Gambar 4.21** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.20*

Berdasarkan gambar 4.21(a) dan (b) yang beda pada nilai *signature*-nya. Hal ini berarti *signature* tidak sah atau *signer* tidak sah. Sedangkan nilai *message hash*-nya sama, berarti e-dokumen sah.

**Kasus 4**: e-dokumen otentik, *signature* yang tidak sah dan kunci publik tidak sah.

Gambar 4.22 menunjukkan hasil verifikasi file “Ipsteks Bagi Masyarakat.doc yang sah/ masih utuh yang disimpan di C:\Documents and

Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.doc dengan *signature* tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.sig dan kunci publik tidak sah (ttd2-pub.pem).




**Gambar 4.22** *Hasil Verifikasi Kasus 4*

Jika diperhatikan potongan pada gambar 4.17 dan 4.22, sebagai hasil keluaran yaitu nilai *massage hash* yang sama dan *signature* berbeda yang dapat dilihat pada gambar 4.23.

Berdasarkan gambar 4.23 (a) dan (b) yang beda pada nilai *signature*-nya. Hal ini berarti *signature* tidak sah dan kunci publik juga tidak sah atau *signer* tidak sah. Sedangkan nilai *message hash*-nya sama, berarti e-dokumen sah.


(a)


(b)

**Gambar 4.23** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.22*

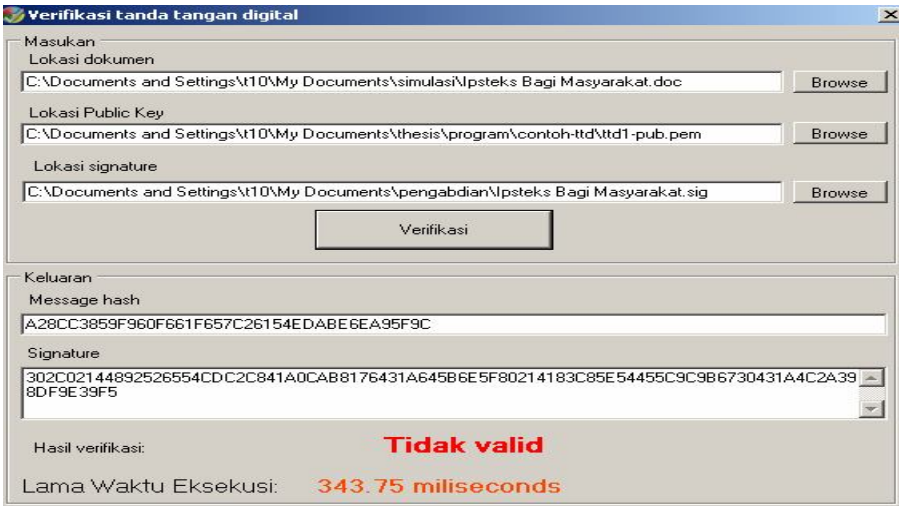
**Kasus 5** : e-dokumen tidak sah/ tidak otentik, *signature* sah dan kunci publik sah.

Gambar 4.24 menunjukkan isi e-dokumen “Ipsteks Bagi Masyarakat.doc” yang tidak otentik dengan diberi tambahan satu spasi pada judul yaitu “Usulan (dua spasi) Program..” file tersebut disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.doc. Beberapa contoh perubahan selain tambahan spasi pada e-dokumen “Ipsteks Bagi Masyarakat.doc” dan hasil verifikasinya diberikan pada lampiran 5. Perubahan pada e-dokumen akan menyebabkan hasil *message digest*-nya berbeda. Hal inilah yang menyebabkan hasil verifikasi “Tidak valid”.

Gambar 4.25 menunjukkan hasil verifikasi file “Ipsteks Bagi Masyarakat.doc yang tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.doc dengan

*signature* sah yang disimpan di C:\Documents and Setting\t10\My Documents\pengabdian\Ipsteks Bagi Masyarakat.sig dan kunci publik sah (ttd1-pub.pem).


**Gambar 4.24** *Tampilan Potongan Halaman Pertama File Ipsteks Bagi Masyarakat.doc yang Tidak Otentik/ Tidak Sah*.




**Gambar 4.25** *Hasil Verifikasi Kasus 5*

Jika diperhatikan potongan pada gambar 4.17 dan 4.25, sebagai hasil keluaran yaitu nilai *massage hash* berbeda dan *signature* sama yang dapat dilihat pada gambar 4.26.


(a)


(b)

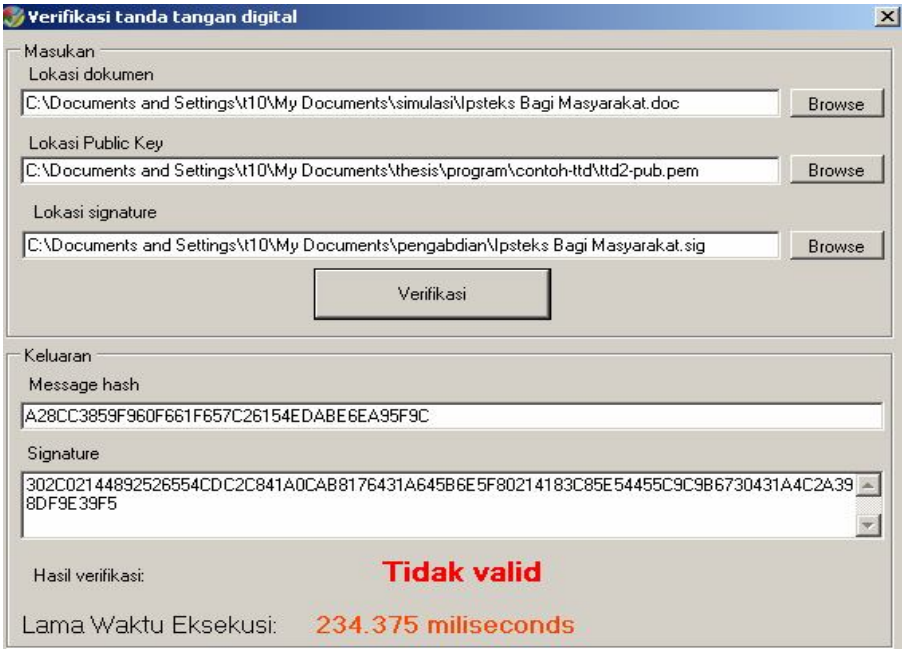
**Gambar 4.26** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.25*

Berdasarkan gambar 4.26 (a) dan (b), yang berbeda pada nilai *message hash*. Hal ini berarti e-dokumen tidak sah/ tidak otentik. Sedangkan nilai *signature*-nya sama, karena *signature* dan kunci publik-nya sah.

**Kasus 6** : e-dokumen tidak sah/ tidak otentik, *signature* sah dan kunci publik tidak sah.

Gambar 4.27 menunjukkan hasil verifikasi file "Ipsteks Bagi Masyarakat.doc yang tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.doc dengan *signature* sah yang disimpan di C:\Documents and Setting\t10\My

Documents\pengabdian\Ipsteks Bagi Masyarakat.sig dan kunci publik tidak sah (ttd2-pub.pem).




**Gambar 4.27** *Hasil Verifikasi Kasus 6*

Jika diperhatikan potongan pada gambar 4.17 dan 4.27, sebagai hasil keluaran yaitu nilai *massage hash* berbeda dan *signature* sama yang dapat dilihat pada gambar 4.28.

Berdasarkan gambar 4.28 (a) dan (b), yang berbeda pada nilai *message hash*. Hal ini berarti e-dokumen tidak sah/ tidak otentik. Sedangkan nilai *signature*-nya sama, karena *signature*-nya sah.

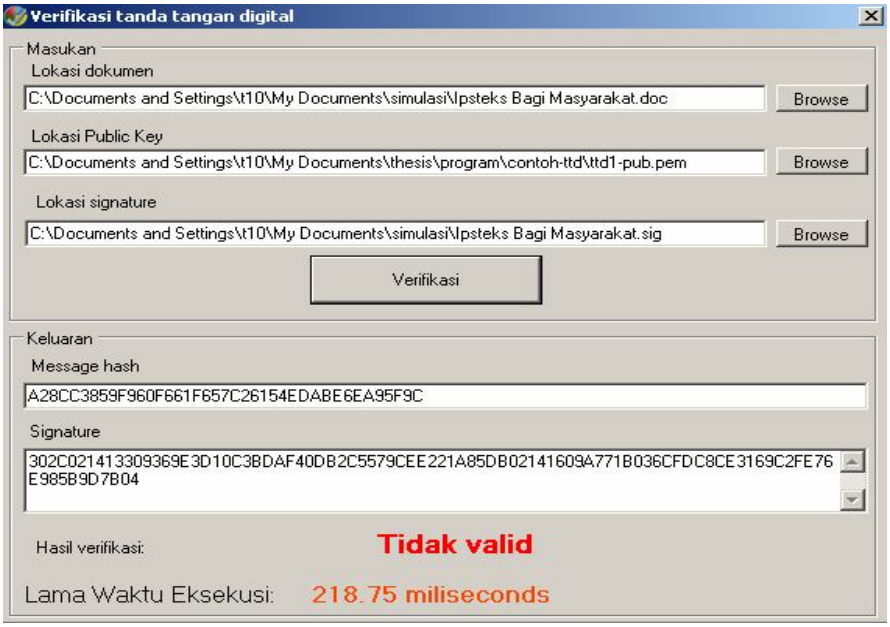

(a)


(b)

**Gambar 4.28** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.27*

**Kasus 7** : e-dokumen tidak sah/ tidak otentik, *signature* tidak sah dan kunci publik sah.

Gambar 4.29 menunjukkan hasil verifikasi file "Ipsteks Bagi Masyarakat.doc yang tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.doc dengan *signature* tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.sig dan kunci publik sah (ttd1-pub.pem).




**Gambar 4.29** *Hasil Verifikasi Kasus 7*

Jika diperhatikan potongan pada gambar 4.17 dan 4.29, sebagai hasil keluaran yaitu nilai *massage hash* berbeda dan *signature* juga berbeda yang dapat dilihat pada gambar 4.30.

Berdasarkan gambar 4.30 (a) dan (b), yang berbeda pada nilai *message hash*. Hal ini berarti e-dokumen tidak sah/ tidak otentik. Sedangkan nilai *signature*-nya juga tidak sama, karena *signature*-nya tidak sah.

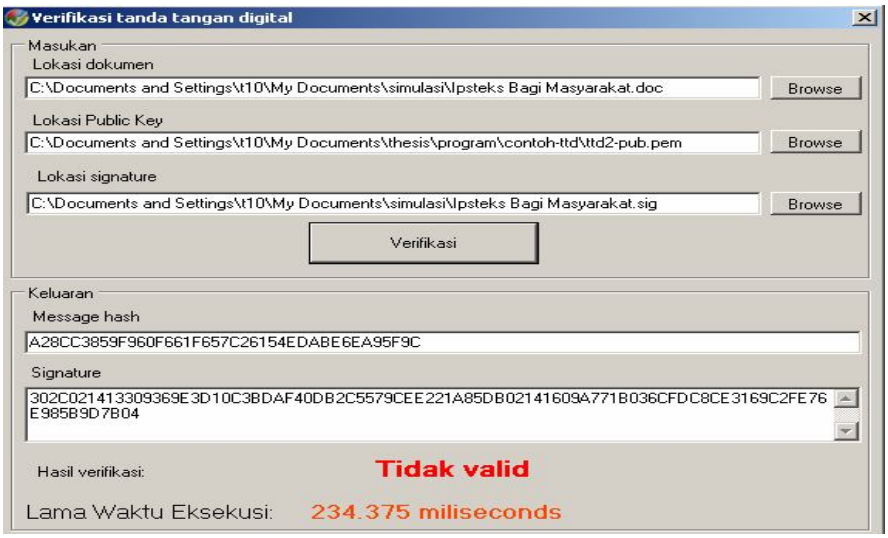

(a)


(b)

**Gambar 4.30** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.29*

**Kasus 8** : e-dokumen tidak sah/ tidak otentik, *signature* tidak sah dan kunci publik tidak sah.

Gambar 4.31 menunjukkan hasil verifikasi file "Ipsteks Bagi Masyarakat.doc yang tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.doc dengan *signature* tidak sah yang disimpan di C:\Documents and Setting\t10\My Documents\simulasi\Ipsteks Bagi Masyarakat.sig dan kunci publik tidak sah (ttd2-pub.pem).




**Gambar 4.31** *Hasil Verifikasi Kasus 8*

Jika diperhatikan potongan pada gambar 4.17 dan 4.31, sebagai hasil keluaran yaitu nilai *massage hash* berbeda dan *signature* juga berbeda yang dapat dilihat pada gambar 4.32.


(a)


(b)

**Gambar 4.32** *Perbandingan Keluaran dari (a) Potongan Gambar 4.17 dan (b) Potongan Gambar 4.31*

Berdasarkan gambar 4.32 (a) dan (b), yang berbeda pada nilai *message hash*. Hal ini berarti e-dokumen tidak sah/ tidak otentik. Sedangkan nilai *signature*-nya juga tidak sama, karena *signature* dan kunci publik-nya tidak sah.

Dari hasil verifikasi kasus 1 sampai 8 dapat diringkas pada tabel 4.4.

**Tabel 4.4** *Rekapitulasi Hasil Verifikasi dari Kasus 1 sampai 8*

| Kasus No | e-dokumen | | Signature | | Kunci publik | | Hasil verifikasi |
|---|---|---|---|---|---|---|---|
| | S | TS | S | TS | S | TS | |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | Valid |
| 2 | 1 | 0 | 1 | 0 | 0 | 1 | Tidak valid |
| 3 | 1 | 0 | 0 | 1 | 1 | 0 | Tidak valid |
| 4 | 1 | 0 | 0 | 1 | 0 | 1 | Tidak valid |
| 5 | 0 | 1 | 1 | 0 | 1 | 0 | Tidak valid |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | Tidak valid |
| 7 | 0 | 1 | 0 | 1 | 1 | 0 | Tidak valid |
| 8 | 0 | 1 | 0 | 1 | 0 | 1 | Tidak valid |

Keterangan : S = sah, TS = tidak sah, 1 = ya, 0 = tidak.

Dari tabel 4.4 dapat disimpulkan bahwa hasil verifikasi valid hanya jika e-dokumen, *signature* dan kunci publik-nya sah. Hal ini analog dengan operasi logika "e-dokumen *and signature and* kunci publik" dimana hanya menghasilkan *true* jika ketiganya *true* atau dalam penelitian ini disebut "sah". Hal ini dijamin dengan potongan kode program pada lampiran 4 yaitu :

```
If File.Exists(txtLokasiDokumen.Text) And
File.Exists(txtLokasiPublicKey.Text) And
File.Exists(txtLokasiSignature.Text) Then
        Dim success As Boolean
```

Berdasarkan simulasi pada kasus 2 sampai 8, jika hasil verifikasinya adalah "Tidak Valid" maka salah satu atau kedua kemungkinan di bawah ini terjadi, yaitu :

a. E-dokumen mengalami perubahan, penambahan atau pengurangan isinya atau sudah tidak utuh/ otentik, dijamin dengan nilai *massage hash* e-dokumen.
b. Penandatangan adalah tidak sah yang berarti bukan orang sebenarnya sebagai *signer,* dijamin dengan nilai *signature* dan atau kunci publiknya.

Kemungkinan lain jika hasil verifikasi "Tidak valid" adalah terjadi kesalahan pengiriman file e-dokumen dan atau *signature* dan atau kunci publiknya. Jadi jika hasil verifikasi "tidak valid" maka *verifier* dapat meyakini bahwa file e-dokumen yang diterima tidak sah/ tidak otentik dan atau *signer* tidak sah.

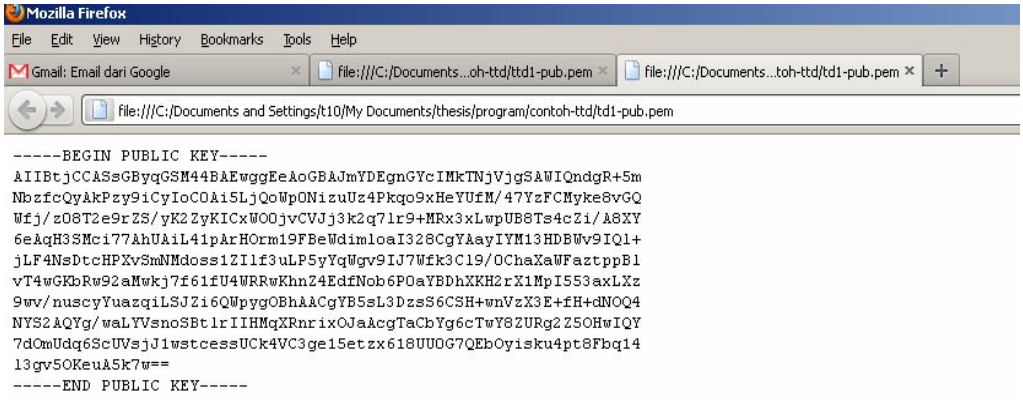
Dimungkinkan juga oleh *attacker,* kunci publik dan atau *signature* yang dikirim oleh *signer* dirusak. Sebagai contoh kunci dan *signature* yang rusak didapat dengan mengganti (me-*edit* manual*)* isi kunci publik dan atau *signature* yang sah lewat text editor/ notepad dan disimpan dalam file dengan format pem untuk kunci dan sig untuk *signature,* atau disimpan dengan format txt dan di-*rename* dengan format pem untuk kunci dan sig untuk *signature.* Oleh *attacker* kunci dan atau *signature* yang dirusak dikirimkan kepada verifier. Hal ini ditunjukkan dengan kunci publik yang digunakan yaitu ttd1-pub.pem seperti pada gambar 4.16 di-*edit* dan di-*rename* sebagai td1-pub.pem yaitu kunci yang rusak,

yang diberikan pada gambar 4.33. Perbedaan isi kunci publik pada ttd1-pub.pem dan td1-pub.pem diberikan pada tabel 4.5.

**Tabel 4.5** *Perbedaan (Dicetak Tebal) Isi ttd1-pub.pem dan td1-pub.pem*

| ttd1-pub.pem | td1-pub.pem |
|---|---|
| **M**IIBtjC…. | **A**IIBtjC…. |

Pada kasus berikut ini, diberikan hasil verifikasi untuk contoh file e-dokumen sah dan tidak sah serta *signature* sah dan tidak sah, dengan kunci yang rusak yaitu td1-pub.pem.

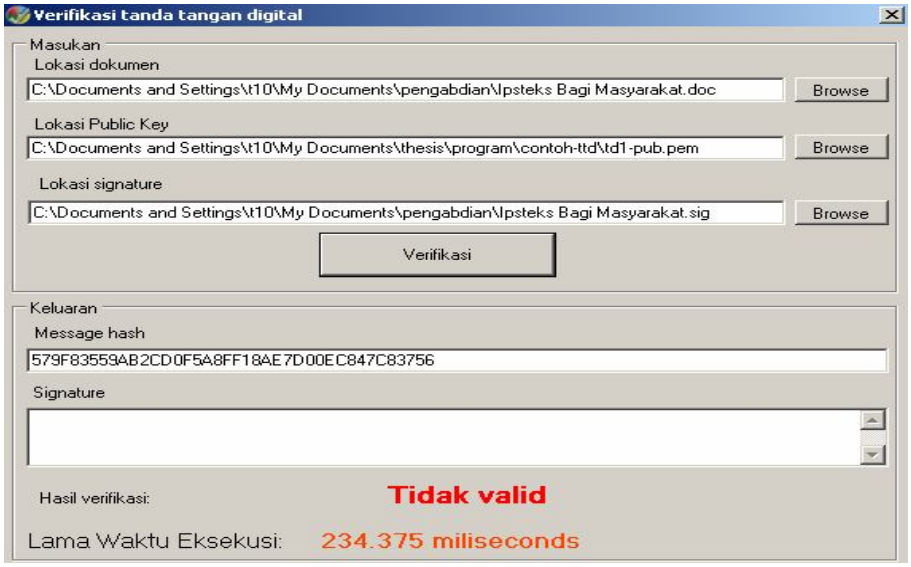



**Gambar 4.33** *Isi td1-pub.pem*

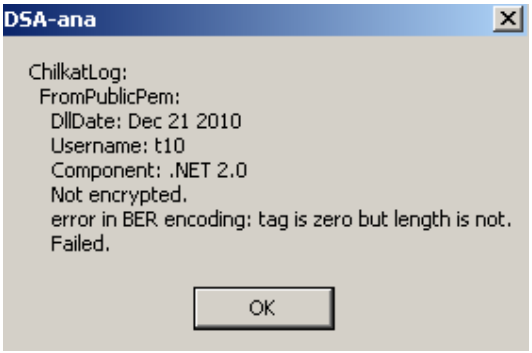
Hasil verifikasi untuk beberapa kasus sebagai berikut :

**Kasus a :**

Jika td1-pub.pem dikirim via e-mail beserta e-dokumen dan *signature* yang sah kepada *verifier* maka hasil verifikasi untuk contoh kasus ini hanya menghasilkan keluaran *massage hash* tanpa *signature* dan hasil verifikasinya "Tidak valid". Hal ini diberikan pada gambar 4.34 beserta jendela peringatannya pada gambar 4.35.




**Gambar 4.34** *Hasil Verifikasi Pada Contoh Kasus a*

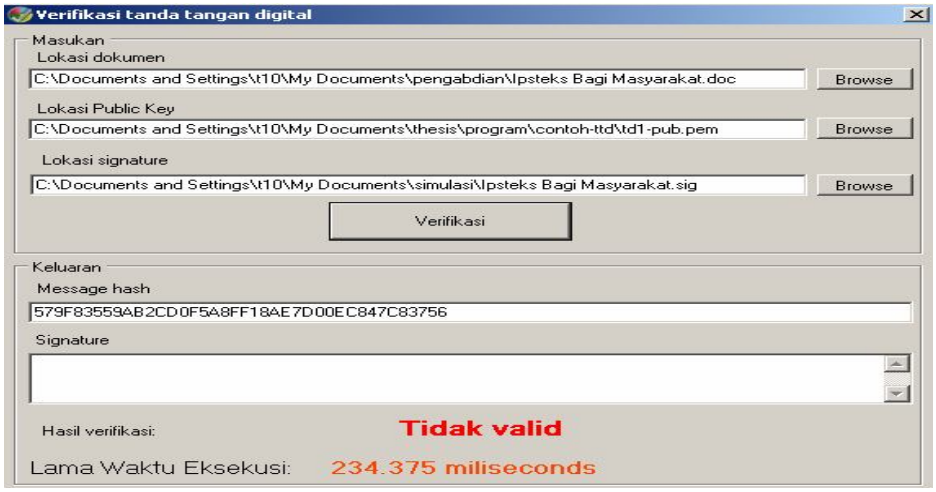



**Gambar 4.35** *Jendela Peringatan yang Muncul pada Contoh Kunci Publik Rusak*

Gambar 4.35 menjelaskan bahwa ada *signature* tidak dapat dibuka karena tidak dapat di-*enkrip* dengan kunci rusak. Kunci rusak karena ada kesalahan pada BER (*Basic Encoding Rules*) pada kunci publik.pem karena *tag* / nilai ujung paling awal diganti manual (pada contoh kasus, M diganti A) tanpa merubah nilai lainnya, jadi panjang kunci tetap/ tidak salah.

**Kasus b :**

Jika td1-pub.pem dikirim via e-mail beserta e-dokumen sah dan *signature* yang tidak sah kepada *verifier* maka hasil verifikasi untuk contoh kasus ini hanya menghasilkan keluaran *massage hash* tanpa *signature* dan hasil verifikasinya "Tidak valid". Hal ini diberikan pada gambar 4.36 beserta jendela peringatan yang sama pada gambar 4.35. Nilai *massage hash* pada gambar 4.34 sama dengan pada gambar 4.36 karena file e-dokumen (sah) yang diverifikasi sama. *Signature* tidak ada/ kosong karena kunci publik td1-pub.pem tidak bisa dipakai untuk membuka *signature*, karena kunci publik tersebut rusak.

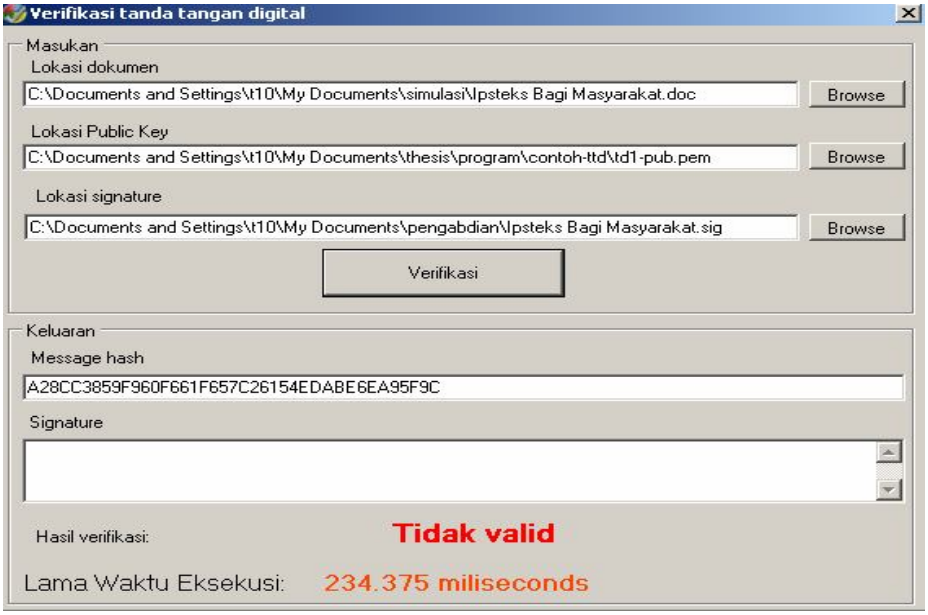



**Gambar 4.36** *Hasil Verifikasi pada Contoh Kasus b*

**Kasus c :**

Jika td1-pub.pem dikirim via e-mail beserta e-dokumen tidak sah dan *signature* yang sah kepada *verifier* maka hasil verifikasi untuk contoh kasus ini hanya menghasilkan keluaran *massage hash* tanpa *signature* dan hasil verifikasinya "Tidak valid". Hal ini diberikan pada gambar 4.37 beserta jendela

peringatan yang sama pada gambar 4.35. Nilai *massage hash* pada gambar 4.34 tidak sama dengan pada gambar 4.35 karena file e-dokumen yang diverifikasi tidak sama, masing-masing e-dokumen yang sah dan tidak sah. *Signature* tidak ada/ kosong karena kunci publik td1-pub.pem tidak bisa dipakai untuk membuka *signature*, karena kunci publik tersebut rusak.

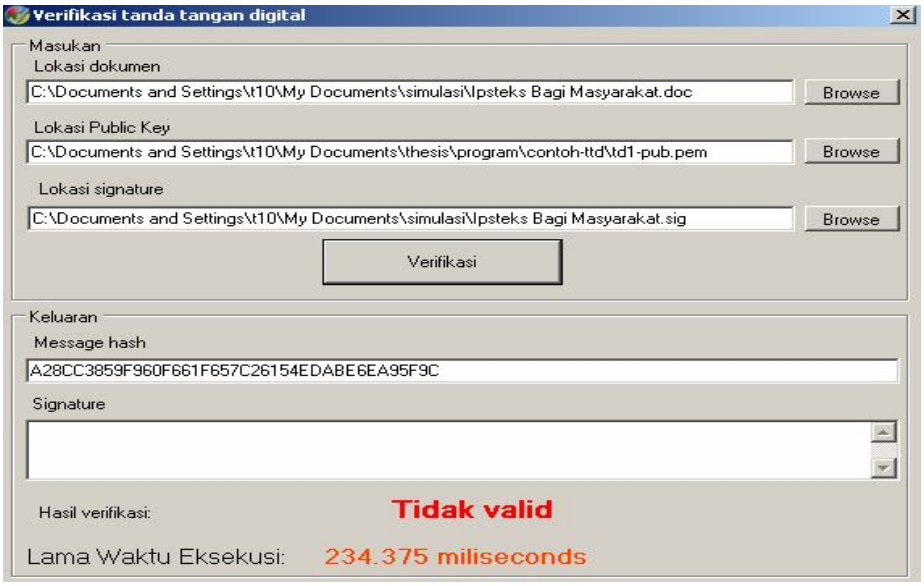



**Gambar 4.37** *Hasil verifikasi pada contoh kasus c*

**Kasus d :**

Jika td1-pub.pem dikirim via e-mail beserta e-dokumen tidak sah dan *signature* yang tidak sah kepada *verifier* maka hasil verifikasi untuk contoh kasus ini hanya menghasilkan keluaran *massage hash* tanpa *signature* dan hasil verifikasinya "Tidak valid". Hal ini diberikan pada gambar 4.38 beserta jendela peringatan yang sama pada gambar 4.35. Nilai *massage hash* pada gambar 4.34 tidak sama dengan pada gambar 4.38 karena file e-dokumen yang diverifikasi

tidak sama, masing-masing e-dokumen yang sah dan tidak sah. *Signature* tidak ada/ kosong karena kunci publik td1-pub.pem tidak bisa dipakai untuk membuka *signature*, karena kunci publik tersebut rusak.




**Gambar 4.38** *Hasil Verifikasi pada Contoh Kasus d*

Pada hasil verifikasi untuk contoh kasus a sampai d dapat diringkas pada tabel 4.6. Dari tabel 4.6 dapat disimpulkan bahwa semua hasil verifikasi kosong/ gagal jika menggunakan kunci publik yang rusak dengan e-dokumen sah atau tidak sah serta signature sah atau tidak sah. Jika tidak ada hasil verifikasi (kosong/ gagal) maka sistem memberikan jendela peringatan seperti pada gambar 4.35 yang menjelaskan bahwa *signature* tidak dapat di-*enkrip* oleh kunci publik tersebut karena kunci publik ada kesalahan pada BER (*Basic Encoding Rules*) atau kesalahan pada aturan penulisan kunci yaitu *tag is zero but length not failed* atau nilai paling ujung awal nol/ tak bernilai tapi panjang kunci tidak salah. Hal

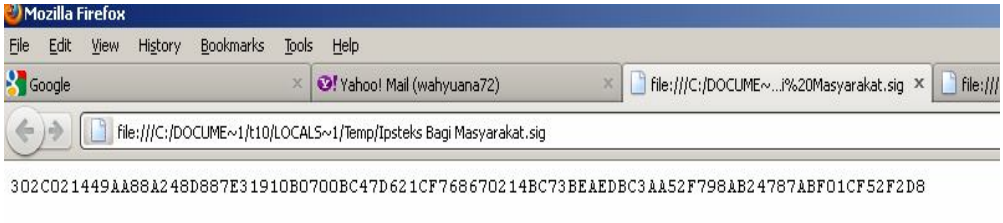
ini karena kunci sah dirusak dengan hanya mengganti nilai paling ujung awal (M diganti A) dan tidak ada nilai kunci yang dihapus.

**Tabel 4.6** *Rekapitulasi Hasil Verifikasi dari Satu Signer dengan Kunci Publik Rusak*

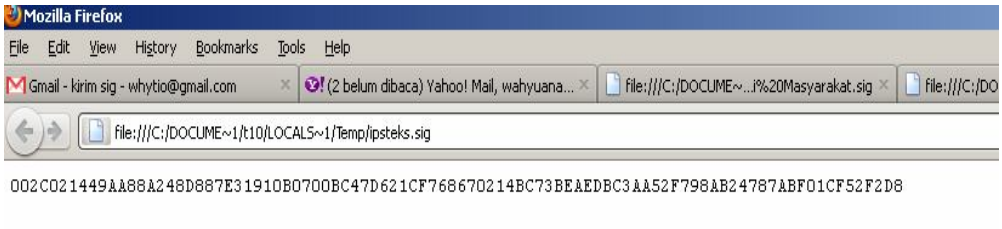
| Kasus No | e-dokumen | | Signature | | Kunci publik | Hasil verifikasi |
|---|---|---|---|---|---|---|
| | S | TS | S | TS | Rusak | |
| a | 1 | 0 | 1 | 0 | 1 | Tidak valid |
| b | 1 | 0 | 0 | 1 | 1 | Tidak valid |
| c | 0 | 1 | 1 | 0 | 1 | Tidak valid |
| d | 0 | 1 | 0 | 1 | 1 | Tidak valid |

Keterangan : S = sah, TS = tidak sah, 1 = ya, 0 = tidak.

Dimungkinkan juga *attacker* merusak *signature.* Gambar 4.39 menunjukkan isi contoh file *signature* yang sah/ asli dan gambar 4.40 menunjukkan contoh isi file *signature* yang rusak dengan merubah pada karakter pertama 3 diganti 0. Perbedaan isi Ipsteks Bagi Masyarakat.sig dan Ipsteks.sig diberikan pada tabel 4.7.




**Gambar 4.39** *Contoh File Signature yang Asli*




**Gambar 4.40** *Contoh File Signature yang Rusak*

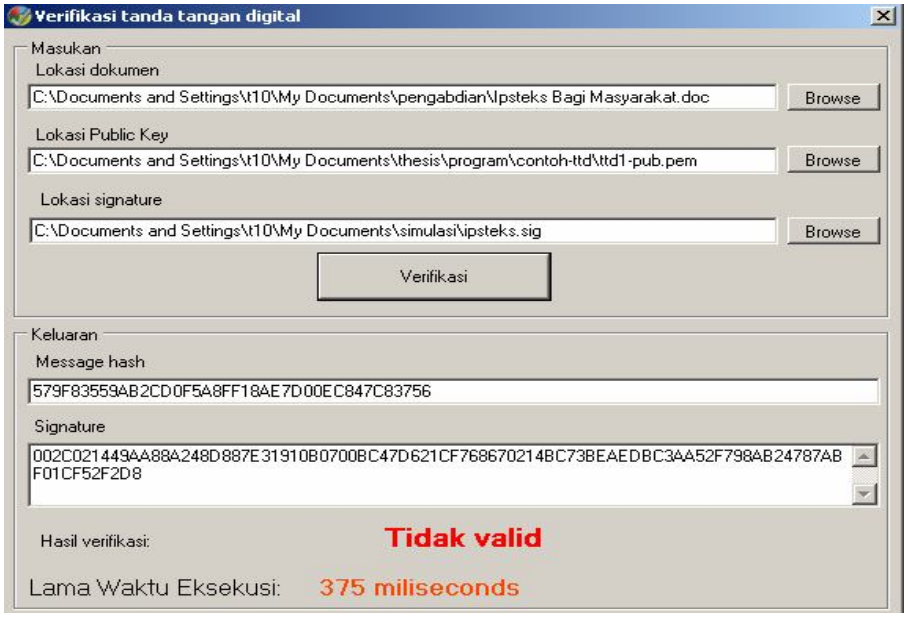
**Tabel 4.7** *Perbedaan (dicetak tebal) isi Ipsteks Bagi Masyarakat.sig dan Ipsteks.sig*

| Ipsteks Bagi Masyarakat.sig | Ipsteks.sig |
|---|---|
| **3**02C02…. | **0**02C02…. |

Contoh file *signature* yang rusak di simpan di C:\Documens and Settings\t10\My Document\simulasi\ipsteks.sig. Contoh file e-dokumen sah, tidak sah serta kunci publik sah, tidak sah analog pada beberapa contoh kasus di atas. Hasil verifikasi untuk beberapa kasus sebagai berikut :

**Kasus e :**

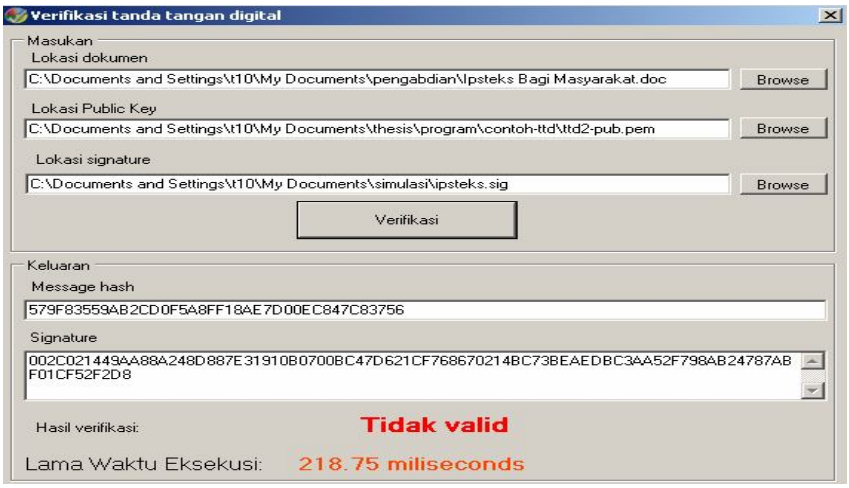
Gambar 4.41 menunjukkan hasil verifikasi untuk contoh e-dokumen yang sah, kunci publik yang sah dan *signature* yang rusak.




**Gambar 4.41** *Contoh Hasil Verifikasi Kasus e*

**Kasus f :**

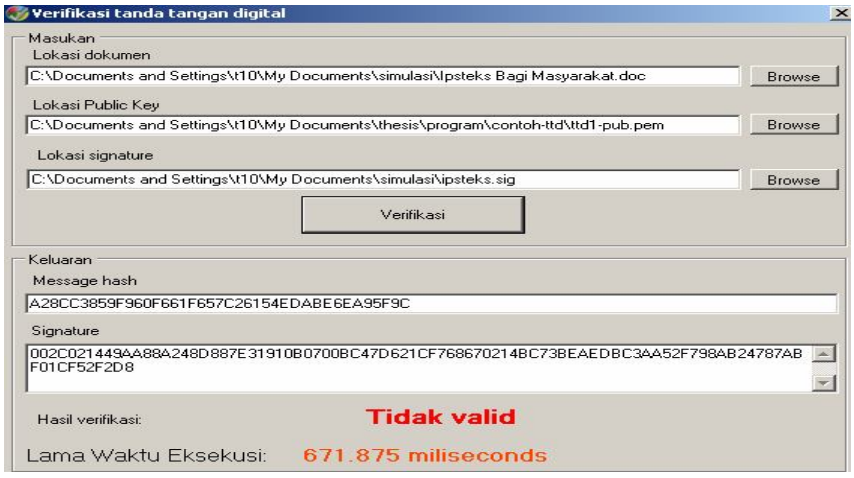
Gambar 4.42 menunjukkan hasil verifikasi contoh e-dokumen yang sah, kunci publik tidak sah dan *signature* yang rusak.




**Gambar 4.42** *Contoh hasil verifikasi kasus f*

**Kasus g :**

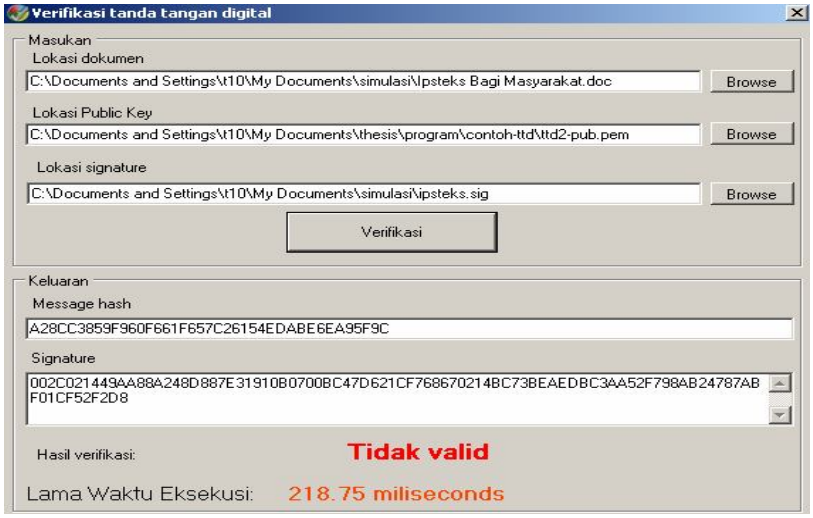
Gambar 4.43 menunjukkan hasil verifikasi contoh e-dokumen tidak sah, kunci publik sah dan *signature* yang rusak.




**Gambar 4.43** *Contoh Hasil Verifikasi Kasus g*

**Kasus h :**

Gambar 4.44 menunjukkan hasil verifikasi contoh e-dokumen tidak sah, kunci publik tidak sah dan *signature* yang rusak.




**Gambar 4.44** *Contoh hasil verifikasi kasus h*

Sedangkan ringkasan hasil verifikasi dari kasus e sampai h diberikan pada tabel 4.8.

**Tabel 4.8** *Rekapitulasi Hasil Verifikasi dari Satu Signer dengan Signature Rusak*

| Kasus No | e-dokumen | | Kunci Publik | | Signature | Hasil verifikasi |
|---|---|---|---|---|---|---|
| | S | TS | S | TS | Rusak | |
| e | 1 | 0 | 1 | 0 | 1 | Tidak valid |
| f | 1 | 0 | 0 | 1 | 1 | Tidak valid |
| g | 0 | 1 | 1 | 0 | 1 | Tidak valid |
| h | 0 | 1 | 0 | 1 | 1 | Tidak valid |

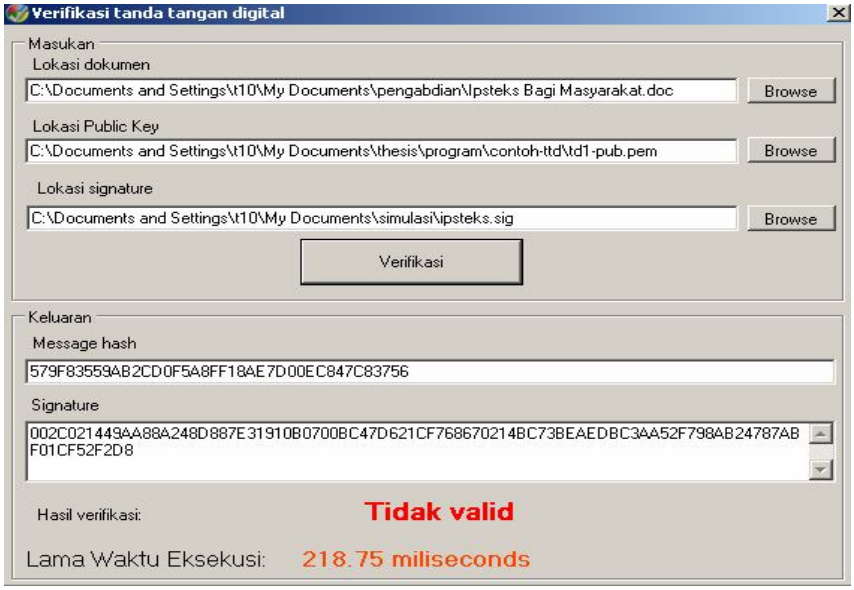
Keterangan : S = sah, TS = tidak sah, 1 = ya, 0 = tidak.

Dari tabel 4.8 dapat disimpulkan bahwa semua hasil verifikasi "Tidak valid" jika menggunakan *signature* yang rusak dengan e-dokumen sah atau tidak sah serta kunci publik sah atau tidak sah.

Pada kasus berikut menjelaskan hasil verifikasi dengan kunci publik dan *signature* yang rusak dan e-dokumen yang sah dan tidak sah.

**Kasus i :**

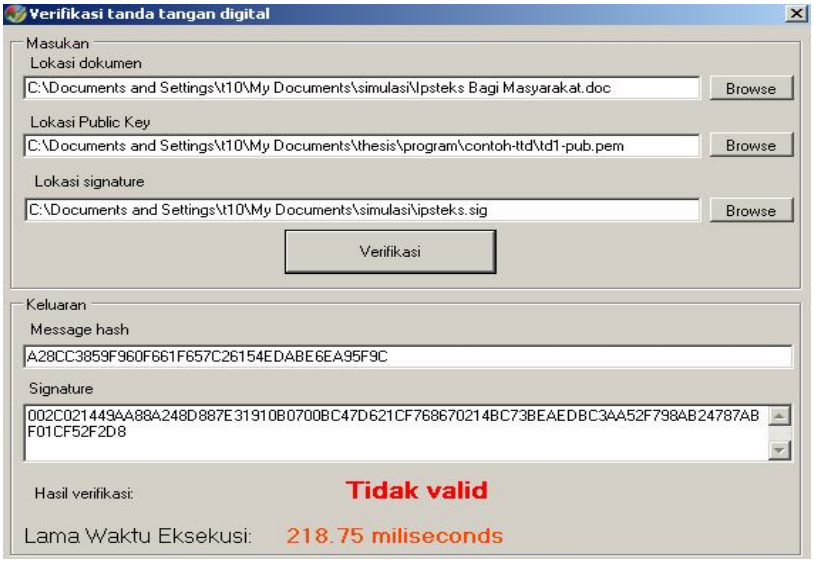
Gambar 4.45 menunjukkan hasil verifikasi contoh e-dokumen sah, kunci publik rusak dan *signature* rusak dengan jendela peringatan yang sama dengan jendela peringatan pada gambar 4.35.




**Gambar 4.45** *Contoh Hasil Verifikasi Kasus i*

**Kasus j :**

Gambar 4.46 menunjukkan hasil verifikasi contoh e-dokumen tidak sah, kunci publik rusak dan *signature* rusak dengan jendela peringatan yang sama dengan jendela peringatan pada gambar 4.35.




**Gambar 4.46** *Contoh hasil verifikasi kasus j*

Hasil verifikasi pada kasus i dan j diringkas pada tabel 4.9.

**Tabel 4.9** *Rekapitulasi Hasil Verifikasi dari Satu Signer dengan Kunci Publik dan Signature Rusak*

| Kasus No | e-dokumen | | Kunci Publik | Signature | Hasil verifikasi |
|---|---|---|---|---|---|
| | S | TS | Rusak | Rusak | |
| i | 1 | 0 | 1 | 1 | Tidak valid |
| j | 0 | 1 | 1 | 1 | Tidak valid |

Keterangan : S = sah, TS = tidak sah, 1 = ya, 0 = tidak.

Berdasarkan tabel 4.9 dapat disimpulkan bahwa hasil verifikasi “Tidak valid” jika *signature* dan kunci publik keduanya rusak walaupun e-dokumen sah atau tidak sah.

Hasil verifikasi berdasarkan tabel 4.4, 4.6, 4.8 dan 4.9 dituliskan bersama pada tabel 4.10.

**Tabel 4.10** *Rekapitulasi dari Tabel 4.4, 4.6, 4.8 dan 4.9*

| No. | Kasus No. | e-dokumen | | Kunci publik | | | *signature* | | | Hasil Verifikasi |
|---|---|---|---|---|---|---|---|---|---|---|
| | | S | TS | S | TS | R | S | TS | R | |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | Valid |
| 2 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | Tidak valid |
| 3 | e | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | Tidak valid |
| 4 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | Tidak valid |
| 5 | 4 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | Tidak valid |
| 6 | f | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Tidak valid |
| 7 | a | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Tidak valid |
| 8 | b | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Tidak valid |
| 9 | i | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Tidak valid |
| 10 | 5 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | Tidak valid |
| 11 | 6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | Tidak valid |
| 12 | g | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | Tidak valid |
| 13 | 7 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | Tidak valid |
| 14 | 8 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | Tidak valid |
| 15 | h | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Tidak valid |
| 16 | c | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | Tidak valid |
| 17 | d | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | Tidak valid |
| 18 | j | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Tidak valid |

Keterangan 1 = ya, 0 = tidak, S = sah, TS = tidak sah, R = rusak.

Berdasarkan tabel 4.10, pihak *verifier* dapat yakin bahwa e-dokumen masih utuh/ asli/ sah dan *signer/ sender* adalah *signer* sebenarnya/ sah jika dan hanya jika hasil verifikasi "Valid". Jika hasil verifikasi "Tidak valid " berarti *verifier* dapat yakin jika e-dokumen sudah tidak utuh/ tidak sah dan atau *signer* bukan *signer* sebenarnya/ sah.

### 4.2.2 Simulasi Tandatangan Digital dengan Dua *Signer*.

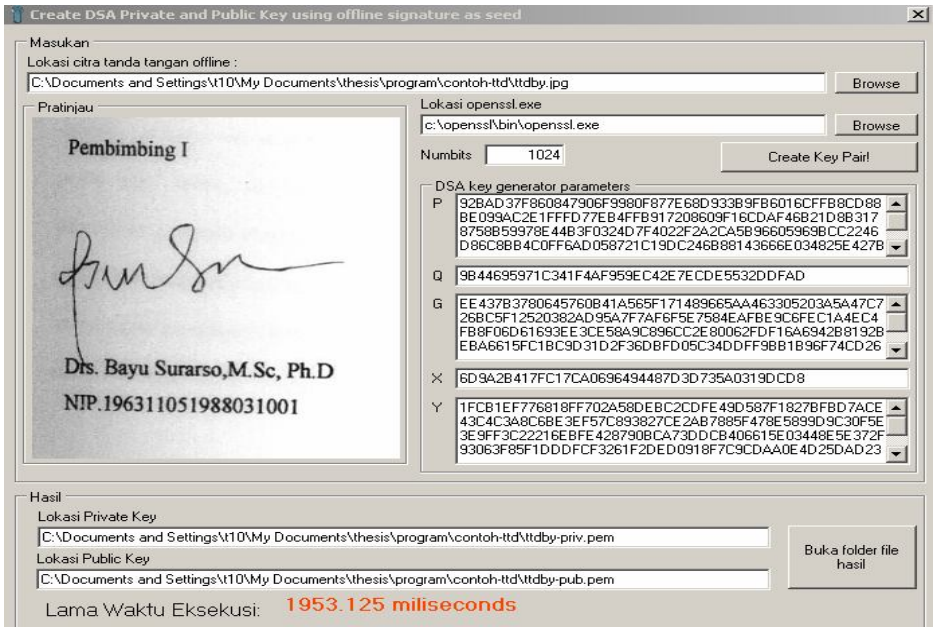
Pada beberapa kasus sering kali penandatangan/ *signer* suatu dokumen atau e-dokumen adalah dua atau lebih orang. Hal ini diperlukan untuk lebih menguatkan kepentingan e-dokumen tersebut. Pada sisi lain pemberian tandatangan *digital* lebih dari satu untuk satu e-dokumen lebih menguatkan keamanan keotentikan isi e-dokumen tersebut dan keabsahan penandatangan. Hal ini sebagai konsep baru yang mengijinkan tandatangan *digital* lebih dari satu sesuai banyak *signer* pada satu e-dokumen.

Pada penelitian ini dilakukan pemberian tandatangan *digital* oleh dua *signer*. E-dokumen yang diperlukan adalah file e-dokumen (*.*) dan *copy* dari file e-dokumen tersebut (copy of*.*). Jika penandatangan lebih dari dua misalnya sebanyak n maka diperlukan *copy* dari file e-dokumen tersebut sebanyak (n-1). Hal ini dilakukan karena hasil *signature* dari masing-masing *signer* akan dibuat dan disimpan oleh sistem secara otomatis sesuai nama file e-dokumennya untuk keperluan verifikasi nantinya. Jadi jika nama file yang sama dibuat dua *signature*, maka file .sig yang dihasilkan adalah *signature* dari *signer* 1 kemudian karena dengan nama file yang sama maka isi file sig tersebut diganti/ ditindih dengan *signature* dari *signer* 2. Jadi file sig tersebut adalah *signature*

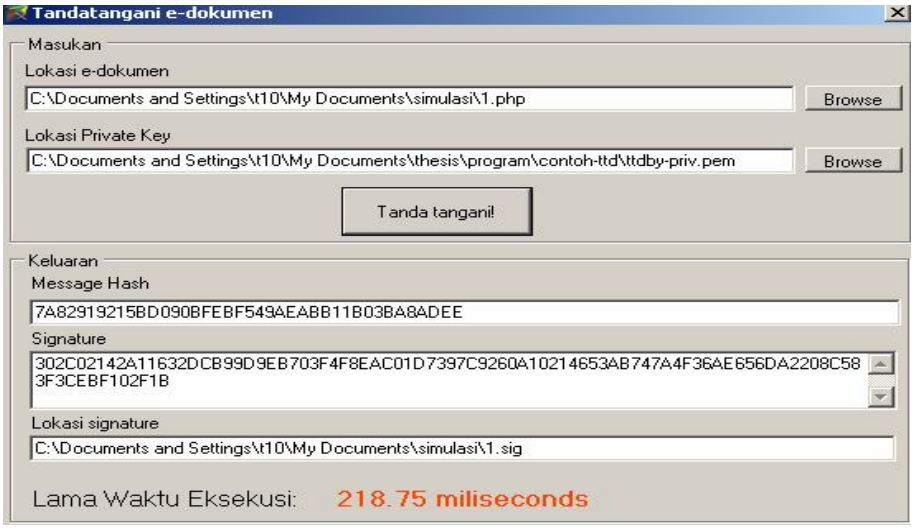
sah dari *signer* 2 dan *signature* tidak sah dari *signer* 1. Jika file .sig tersebut diverifikasi dengan asumsi e-dokumen adalah sah dan menggunakan kunci publik *signer* 2 maka hasil verifikasi adalah valid, jika menggunakan kunci publik *signer* 1 (analog pada kasus 1), hasi verifikasinya tidak valid (analog pada kasus 4).

Pada contoh simulasi ini terdapat dua *signer* yaitu *signer* 1 adalah Drs. Bayu Surarso, M.Sc, Ph.d, dan *signer* 2 adalah Aris Sugiharto, S.Si, M.Kom dengan file yang akan ditandatangani adalah 1.php dan copy of 1.php. Sebagai masukan adalah tandatangan *offline signer* beserta keterangan. Keterangan tersebut hanya contoh, jadi bisa diganti nomer dokumen, nama *signer* atau keterangan lainnya yang menunjukkan otorisasi (pemberi kuasa) pada dokumen fisik. Gambar 4.47 menunjukkan hasil pembuatan kunci *signer* 1 dan gambar 4.49 menunjukkan hasil pembuatan kunci *signer* 2. Gambar 4.48 menunjukkan *signature* dari *signer* 1 dan gambar 4.50 menunjukkan *signature* dari *signer* 2. Sedangkan contoh isi dari kunci privat, publik dan *signature* pada *mode base* 64 dari *signer* 1 dan 2 diberikan masing-masing pada lampiran 7 dan 8. Kemudian e-dokumen (1.php), *signature* 1.sig, copy 1.sig, dan kunci publik ttdars-pub.pem dan ttdby-pub.pem dikirim via *e-mail*. *E-mail* akan diterima oleh *verifier* ditunjukkan pada gambar 4.51, yang akan memverifikasi keaslian e-dokumen tersebut dan keaslian penandatangan. Pada e-mail tersebut diberi keterangan file *signature* dengan kunci pasangannya, pada contoh ini keterangannya adalah “file 1.php diverifikasi menggunakan *signature* 1.sig untuk kunci ttdby-pub.pem dan copy of 1.sig untuk kunci ttdars-pub.pem” yang diberikan pada gambar 4.51. File

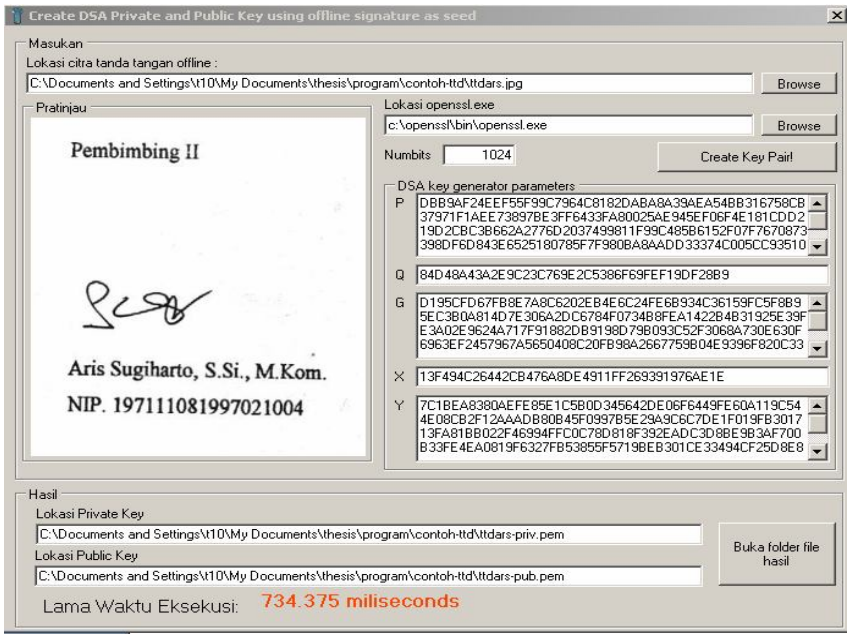
e-dokumen, kunci publik dan *signature*- nya dapat dikirim bersama atau terpisah lewat internet via *email.* Setelah file lampiran diterima dan diunduh *verifier* maka dapat dilakukan proses verifikasi.



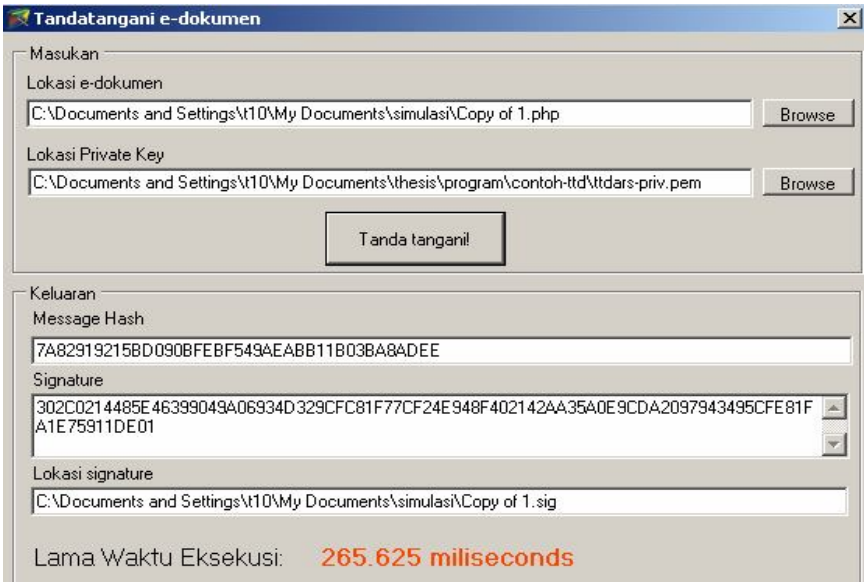

**Gambar 4.47** *Contoh Tampilan Pembuatan Kunci oleh Signer 1*




**Gambar 4.48** *Contoh Tampilan Hasil Tandatangan Digital dari Signer 1*




**Gambar 4.49** *Contoh Tampilan Pembuatan Kunci oleh Signer 2.*

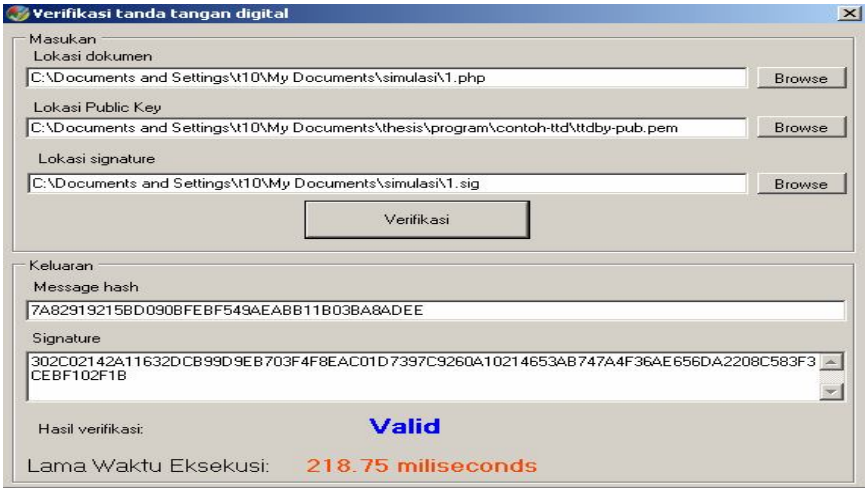



**Gambar 4.50** *Contoh Tampilan Hasil Tandatangan Digital dari Signer 2.*

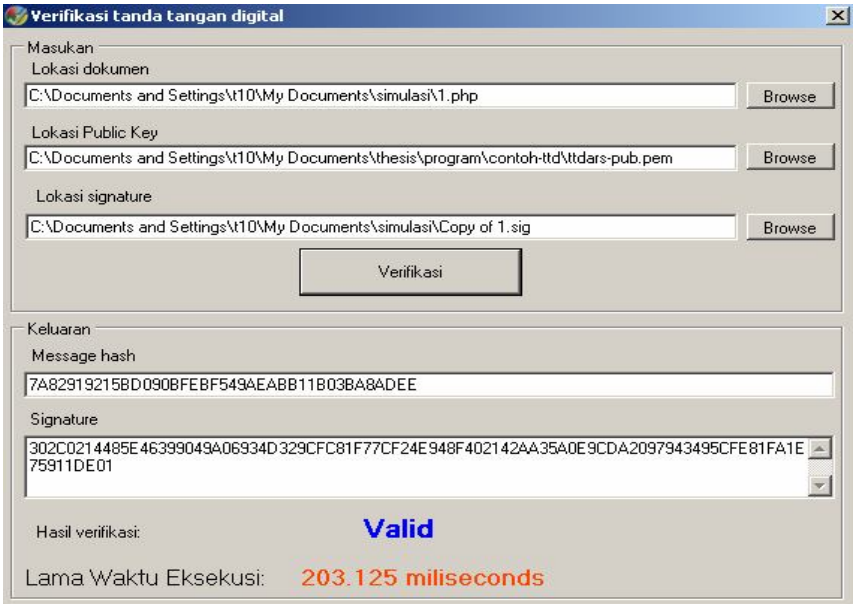

**Gambar 4.51** *Contoh Tampilan e-mail yang Diterima Verifier.*

Pada hasil verifikasi analog pada subbab 4.2.1.2 yaitu hanya bernilai *true* (dalam penelitian ini disebut valid) jika e-dokumen, *signature*, dan kunci publik ketiganya *true* atau dalam penelitian ini disebut "sah".

Jika isi e-dokumen tidak mengalami peubahan/ masih otentik dan kunci publik yang digunakan benar maka hasil verifikasi dari kedua pengguna tersebut adalah valid. Sebagai contoh hasil verifikasi untuk kasus ini, untuk masing-masing *signer* diberikan pada gambar 4.52 dan 4.53.




**Gambar 4.52** *Contoh Tampilan Hasil Verifikasi dari Pengguna 1*




**Gambar 4.53** *Contoh Tampilan Hasil Verifikasi dari Pengguna 2*

Berdasarkan hasil verifikasi pada subbab 4.2.1.2, verifikasi dilakukan dengan masukan satu *signature* dan satu kunci publik. Jadi verifikasi dengan dua *signer* dilakukan untuk masing-masing *signature* dan kunci publik *signer*. Sehingga kombinasi kemungkinan sah atau tidaknya e-dokumen, *signature* dan kunci publik beserta verifikasinya, analog pada subbab 4.2.1.2 untuk masing-masing *signer*. Pada kasus ini, *verifier* menerima/ yakin dengan keotentikan e-dokumen dan keabsahan *signer* jika hasil verifikasi dari kedua *signer* adalah "valid". Hal ini berlaku pula untuk *signer* sebanyak n, n = 1,2,3,…jika sebanyak (semua) n hasil verifikasi adalah "valid" maka e-dokumen otentik/ sah dan n *signer* adalah sah/ penandatangan dokumen sebenarnya.

### 4.2.3 Analisa Lama Waktu Eksekusi

Tabel 4.11 menunjukkan lama waktu eksekusi pada proses pembuatan kunci dengan waktu minimum 687,5 milidetik, maksimum 3031,25 milidetik dan rata-rata 1317,729 milidetik. Tabel 4.12 menunjukkan lama waktu eksekusi pada proses pembuatan tandatangan *digital* dengan waktu minimum 218,75 milidetik, maksimum 296,875 milidetik dan rata-rata 265,625 milidetik.. Tabel 4.12 menunjukkan lama waktu eksekusi pada proses verifikasi dengan waktu minimum 203,125 milidetik, maksimum 671,875 milidetik dan rata-rata 261,719 milidetik. Berdasarkan tabel 4.11, 4.12 dan 4.13 dapat diketahui aspek kenyamanaan pada waktu eksekusi ketiga proses tersebut maksimum 3031,25 milidetik, waktu minimum 218,75 milidetik dan waktu rata-rata 615,024 milidetik atau 1,025 menit. Jadi pengguna hanya menunggu mendapatkan hasil proses pembuatan kunci, *signature* dan verifikasi rata-rata selama 1,025 menit.

**Tabel 4.11** *Lama Waktu Eksekusi pada Proses Pembuatan Kunci*

| Ditunjukkan pada gambar no. | Lama waktu eksekusi (miliseconds) |
|---|---|
| 4.4 | 3031,25 |
| 4.8 | 765,625 |
| 4.9 | 687,5 |
| 4.47 | 1953,125 |
| 4.48 | 734,375 |
| 4.49 | 734,375 |
| Rata-rata waktu eksekusi | 1317,729 |

**Tabel 4.12** *Lama Waktu Eksekusi pada Proses Pembuatan Tandatangan Digital*

| File Ipsteks Bagi Masyarakat.doc dengan ukuran file 1,14 MB | |
|---|---|
| Ditunjukkan pada no gambar | Lama waktu eksekusi (miliseconds) |
| 4.10 | 281,25 |
| 4.11 | 296,875 |
| File 1.php dengan ukuran file 1,87 KB | |
| 4.48 | 218,75 |
| 4.50 | 265,625 |
| Rata-rata waktu eksekusi | 265,625 |

**Tabel 4.13** *Lama Waktu Eksekusi pada Proses Verifikasi*

| File Ipsteks Bagi Masyarakat.doc dengan ukuran file 1,14 MB | |
|---|---|
| Ditunjukkan pada no gambar | Lama waktu eksekusi (miliseconds) |
| 4.17 | 234,375 |
| 4.18 | 218,75 |
| 4.20 | 234,375 |
| 4.22 | 234,375 |
| 4.25 | 343,75 |
| 4.27 | 234,375 |
| 4.29 | 218,75 |
| 4.31 | 234,375 |

| Ditunjukkan pada no gambar | Lama waktu eksekusi (miliseconds) |
|---|---|
| 4.34 | 234,375 |
| 4.36 | 234,375 |
| 4.37 | 234,375 |
| 4.38 | 234,375 |
| 4.41 | 375 |
| 4.42 | 218,75 |
| 4.43 | 671,875 |
| 4.44 | 218,75 |
| 4.45 | 218,75 |
| 4.46 | 218,75 |
| File 1.PHP dengan ukuran file 1,87 KB | |
| Ditunjukkan pada no gambar | Lama waktu eksekusi (miliseconds) |
| 4.52 | 218,75 |
| 4.53 | 203,125 |
| Rata-rata waktu eksekusi | 261,719 |

# BAB V

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan pembahasan mengenai aplikasi kriptografi untuk pengamanan e-dokumen dengan metode *hybrid* : Biometrik tandatangan dan DSA (*Digital Signature Algorithm*) dapat diambil kesimpulan :

1. Biometrik tandatangan manual *offline* pengguna dapat digunakan untuk membuat kunci privat. Kunci privat yang dihasilkan, digunakan untuk membuat kunci publik pasangannya. Panjang kunci antara 512 sampai 1024 bit sesuai standar keamanan yang dikeluarkan oleh FIPS (*Federal Information Processing Standard*). Dari masukan satu tandatangan *offline* dapat menghasilkan lebih dari satu pasang kunci. Hal ini menunjukkan pembangkitan kunci secara dinamis sebagai konsep baru pada penggunaan kunci untuk satu kali pakai.
2. Tandatangan *digital* dapat memenuhi kebutuhan ketidaktunggalan *signer*. Hal ini sebagai konsep baru pada penerapan tandatangan *digital* yang memenuhi kebutuhan otorisasi satu e-dokumen dengan lebih dari satu *signer*.
3. Keamanan implementasi Biometrik tandatangan dan DSA pada penelitian ini didasarkan atas satu e-dokumen jika ditandatangani oleh n *signer* maka harus diverifikasi sebanyak n kali. Jika semua n verifikasi bernilai valid berarti telah menembus n lapis keamanan

dalam hal verifikasi. Sebaliknya jika salah satu atau lebih hasil verifikasi tidak valid maka *verifier* dapat mengetahui jika e-dokumen yang diterima sudah tidak otentik dan atau salah satu atau lebih dari *signer* adalah bukan orang yang sebenarnya menandatangani e-dokumen tersebut. Selain itu keamanan pada kunci yang dihasilkan adalah pada sulitnya mencari pemfaktoran bilangan prima besar khususnya pada 1024 bit (Feng, H;& Chong Wa,C. 2002).

4. Pada implementasi tandatangan *digital* dengan metode *hybrid :* Biometrik tandatangan dan DSA terpenuhi kebutuhan keamanan e-dokumen dalam hal :
    - Kerahasiaan (*confidentiality*) *signature* hanya dapat didekrip oleh *verifier* dengan kunci publik pasangan kunci privat pada pihak *signer*.
    - Keutuhan atau keotentikan (*integrity*) e-dokumen yang ditransmisi, dijamin dengan hash SHA-1 dari e-dokumen tersebut.
    - Jaminan atas identitas dan keabsahan (*authenticity*) n *signer* dengan n *signature* yang dihasilkan serta hasil verifikasinya, dimana n = 1,2,3,....

## 4.1 Saran

Saran yang dapat diberikan terkait dengan implementasi tandatangan *digital* dengan metode *hybrid :* Biometrik tandatangan dan DSA yang telah diteliti yaitu :

1. Untuk lebih mengoptimalkan sisi keamanan transmisi e-dokumen lewat internet adalah dengan menggunakan tandatangan *offline* dan menambahkan algoritma pencocokan tandatangan. Algoritma pencocokan tandatangan *offline* misalnya dengan metode P-*tree* pada AHVS (*Automatic Handwritten Verification System*) (Najmul, 2006).
2. Program ini dapat dipakai sebagai masukan pada infrastruktur kunci publik di Indonesia dengan penambahan sertifikat *digital*, sehingga tujuan kriptografi dalam hal nir-penyangkalan (*non-repudiation*) dapat tercapai.
3. Pada kunci privat yang dihasilkan dapat tidak ditampilkan (disembunyikan) atau disimpan dengan menyandikannya, misalnya dengan algoritma 3DES atau algoritma enkripsi yang lain. Hal ini dilakukan untuk meningkatkan keamanan dalam hal penyimpanan kunci privat yang bersifat rahasia.

## DAFTAR PUSTAKA


**Chandra Pravir, Messier Matt, Viega John**. 2002. *Network Security With OpenSSL*, O'Relly

**Feng, H; & Chong Wa,C**. 2002. *Private Key Generation from On-Line Handwritten Signature*, Information Management and Computer Security, Emerald journal 10 (4).

**FIPS PUB 186-3**. 2009. *Digital Signature Standard (DSS)*. http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf. Diakses tanggal 3 Mei 2011

**FIPS PUB 183-3**. 2008. *Secure Hash Standard*, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf. Diakses tanggal 3 Mei 2011

**Hassler, V;& Helmut, B**.1999. *Digital Signature Management*, Internet Research, Emerald journal 9 (4).

**Kurniawan, Y.** 2004. *Kriptografi Keamanan Internet dan Jaringan Telekomunikasi*, Informatika, Bandung.

**Munir, R.** 2006. *Kriptografi*. Informatika, Bandung.

**Munir, R.** 2005. *Penggunaan Tanda Tangan Digital untuk Menjaga Integritas Berkas Perangkat Lunak*, Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005) ISBN: 979-756-061-6 Yogyakarta, 18 Juni 2005

**Najmul, A.K.M**. 2006. *Handwritten Signature Verification Using P-Tree*, Asian journal of information Technologi 5 (3)

**Pawan, K.J.;& Siyal, M. Y**. 2001. *Novel Biometric Digital Signature for Internet Based Applications*. Information Management & Computer Security, Emerald journal 9 (5).

**Pressman, S.R**. 1997. *Software Engineering a Practitioner's Approach*, USA Mc Grawhill. Inc

**Wu, Tzong-Chen; & Su, Ru-lan**.1997. *ID-based Group-Oriented Cryptosystem and its Digital Signature Scheme*, Computer Communications Elsevier Science B.V. 20.

**Whitten, L.J, Bentley, D.L & Dittman, C.K**. 2004. *System Analysis and Design Methods,* USA Mc Grawhill, Inc.

Lampiran 1.

Kode program form utama :

```
Public Class frmUtama

    Private Sub KeluarToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
KeluarToolStripMenuItem
        Application.Exit()
    End Sub

    Private Sub TentangToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TentangToolStripMenuItem
        frmAbout.ShowDialog()
    End Sub

    Private Sub
CreatePrivateAndPublicKeyToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CreatePrivateAndPublicKeyToolStripMenuItem
        frmCreateKey.Show(Me)
    End Sub

    Private Sub DSASignDocumentToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
DSASignDocumentToolStripMenuItem
        frmSign.Show(Me)
    End Sub

    Private Sub DSAVerifyDocumentToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
DSAVerifyDocumentToolStripMenuItem
        frmVerify.Show(Me)
    End Sub
End Class
```

Lampiran 2.

Kode program form crate key :

```
Imports System.IO
'untuk library input/output file semacam memeriksa apakah file
itu ada atau tidak
Imports System.Diagnostics
'untuk mengeksekusi program lain (openSSL)

Public Class frmCreateKey

    Private Sub btnBrowseLokasiTTD_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBrowseLokasiTTD
        If ofd1.ShowDialog = Windows.Forms.DialogResult Then
            'menampilkan open file dialogue (ofd1)
            txtLokasiTTD.Text = ofd1.FileName
            'menampilkan file terpilih di teks box
txtLokasiTTD.Text
            If File.Exists(ofd1.FileName) Then
                'fungsi dr sistem IO untuk memeriksa apakah file
terpilih ada/ tidak
                picTTD.ImageLocation = ofd1.FileName
                'jika ada maka file ditampilkan di picTTD
            Else
                MsgBox("Berkas citra tidak ditemukan!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal error")
                'jika tidak ada maka muncul MsgBox tersebut di
atas
            End If
        End If
    End Sub

    Private Sub btnBrowseOpenSSL(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnBrowseOpenSSL
        If ofd2.ShowDialog = Windows.Forms.DialogResult.OK Then
            'fungsi untuk menampilkan ofd2 yaitu lokasi
openSSL.exe
            txtOpenSSL.Text = ofd2.FileName
            'memanpilkan lokasi openSSL.exe
        End If
    End Sub

    Private Sub btnCreateKey_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCreateKey
        'deklarasi lokal variabel
        Dim dir_loc As String
        Dim dsaparamfile As String
        Dim ttd_file As String
        Dim obj As Process 'dari System.Diagnostics
        Dim success As Boolean
        Dim priv_name As String
        Dim pub_name As String
```

```
        If File.Exists(txtOpenSSL) And File.Exists(txtLokasiTTD)
Then
            'disable tombol "Create Key pair"
            btnCreateKey = "Sedang bekerja"
            'mengubah text create key pair! menjadi sedang
bekerja
            btnCreateKey = False
            'membuat tombol btnCreateKey tidak bisa diklik lagi

            'langkah pertama: bikin parameter DSA menggunakan
openssl dengan file citra tanda tangan
            'sbg seed random prime number generator
            dir_loc = Path.GetDirectoryName(txtLokasiTTD)
            'menampilkan path txtLokasiTTD tanpa filenya
            ttd_file =
Path.GetFileNameWithoutExtension(txtLokasiTTD)
            'menampilkan nama file tanpa ekstensi, berguna untuk
membuat file dan ekstensi pada DSAparam,priv, dan pub
            dsaparamfile = dir_loc + Path.DirectorySeparatorChar
+ ttd_file + "-dsaparam.pem"
            'nama file hasil openSSL

            'perintah: openssl dsaparam -outform PEM -out ds.pem
-rand ttd.jpg 1024
            'perintah: openssl dsaparam -outform PEM -out
e:\work\mba_ana\ttd1-dsaparam.pem -rand ttd.jpg 1024(PEM=privacy
enhanced mail)
            objCmd = New Process()
            'inisialisasi objekCmd dari tipe kelas process
            objCmd.StartInfo.FileName = txtOpenSSL.Text  'nama
dan lokasi executable nya = openssl.exe
            'kumpulan argumen untuk meng-generate dsaparam
            objCmd.StartInfo.Arguments = "dsaparam -outform PEM -
out """ + dsaparamfile + """ -rand """ + txtLokasiTTD.Text + """
" + txtNumbits.Text
            'kumpulan argumen untuk membuat file dsaparam
            objCmd.StartInfo.WindowStyle =
ProcessWindowStyle.Normal
            'menampilkan window cmd pada kondisi normal (tidak
min,max,hidden)unt.melihat proses openSSL

            'langkah kedua: bikin private dan public key dari
parameter td
            Dim dsa As New Chilkat.Dsa()
            'inisialisasi variabel dsa sebagai pemroses pub dan
priv key dsa yang dilakukan pada chilkatsoft.com

            If (success <> True) Then 'jika key invalid maka
muncul MsgBox dan keluar dari sub
                MsgBox(dsa.LastErrorText)
                Exit Sub 'keluar dari sub
            End If

            success = dsa.GenKeyFromParamsPemFile(dsaparamfile)
'ambil parameter dsa hasil dari openSSL
```

```
            If (success <> True) Then
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If

            'public key
            Dim pemStr As String
            pemStr = dsa.ToPublicPem() 'bikin pub key dengan
format PEM disimpan di variabel pemStr
            pub_name = dir_loc + Path.DirectorySeparatorChar +
ttd_file + "-pub.pem" 'tentukan tempat file menyimpan pub
            success = dsa.SaveText(pemStr, pub_name) 'simpan
variabelpemStr ke file pub_name
            'private key
            pemStr = dsa.ToPem() 'buat priv key pada format PEM
            priv_name = dir_loc + Path.DirectorySeparatorChar +
ttd_file + "-priv.pem" 'tentukan tempat menyimpan priv
            success = dsa.SaveText(pemStr, priv_name) 'proses
simpan pemStr ke priv_name

            'tampilkan parameter dsa dalam hexa
            txtP.Text = dsa.HexP
            txtQ.Text = dsa.HexQ
            txtG.Text = dsa.HexG
            txtX.Text = dsa.HexX
            txtY.Text = dsa.HexY
            txtPrivateKey.Text = priv_name 'tampilkan lokasi
penyimpanan priv key
            txtPublicKey.Text = pub_name 'tampilkan lokasi
penyimpanan pub key
            MsgBox("Selesai membuat pasangan private-public
key!") 'tampilkan pesan tersebut


            're-enable tombol create key
            btnCreateKey.Text = "Create Key Pair!" 'membuat
tombol btnCreateKey bisa diklik lagi,krn proses sudah selesai

            btnCreateKey.Enabled = True
        Else
            MsgBox("Berkas citra atau openssl.exe tidak
ditemukan!", MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal
error")
        End If
    End Sub

    Private Sub btnBukaFolder_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBukaFolder.Click
        Dim folder_hasil As String 'mendeklaraskan variabel
folder_hasil dengan tipe data String
        folder_hasil = Path.GetDirectoryName(txtPrivateKey.Text)
'ambil lokasi direktori priv key

        Process.Start("Explorer.EXE", folder_hasil) 'buka
explorer di foldel hasil
```

```
    End Sub
End Class
```

Lampiran 3.

Kode program form sign :

```
Imports System.IO

Public Class frmSign

    Private Sub btnBrowseDokumen_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBrowseDokumen.Click
        If ofd1.ShowDialog = Windows.Forms.DialogResult.OK Then
            If File.Exists(ofd1.FileName) Then
                txtDokumen.Text = ofd1.FileName
            Else
                MsgBox("Berkas tidak valid!",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal error")
            End If
        End If
    End Sub

    Private Sub btnBrowsePrivateKey_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBrowsePrivateKey.Click
        If ofd2.ShowDialog = Windows.Forms.DialogResult.OK Then
            If File.Exists(ofd2.FileName) Then
                txtPrivateKey.Text = ofd2.FileName
            Else
                MsgBox("Berkas private key tidak ditemukan",
MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal error")
            End If
        End If
    End Sub

    Private Sub btnSign_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSign.Click
        If File.Exists(txtDokumen.Text) And
File.Exists(txtPrivateKey.Text) Then
            Dim success As Boolean
```

```
            Dim crypt As New Chilkat.Crypt2() 'inisialisai klas
Crypt2 untuk membuat MD dari SHA-1
            If (success <> True) Then
                MsgBox(crypt.LastErrorText)
                Exit Sub
            End If

            'langkah pertama: hash message!
            crypt.EncodingMode = "hex" 'hasil MD dalam hexa
            crypt.HashAlgorithm = "sha-1" 'gunakan SHA-1 sbg
algoritma

            Dim hashStr As String 'delkarasikan variabel hashStr
untuk menyimpan hasil ke hashStr
            hashStr = crypt.HashFileENC(txtDokumen.Text) 'proses
hash
            txtMsgHash.Text = hashStr 'menampilkan hashStr di
txtMsghash supaya visibel dilihat user

            'langkah kedua: ambil private key
            'mendeklarasikan objek DSA untuk ttdigital
            Dim dsa As New Chilkat.Dsa()
            If (success <> True) Then
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If

            'mendeklarasikan priv key sebagai variabel string
            Dim priv_key As String
            priv_key = ""
            'tfs= text file string
            Dim tfs As TextReader
            tfs = File.OpenText(txtPrivateKey.Text) 'membuka file
priv key yg dipilih
            priv_key = tfs.ReadToEnd 'membaca isi file priv key &
masukkan isi file ke var priv key
            tfs.Close() 'tutup objek tfs agar efisien memori

            'success = dsa.LoadText(txtPrivateKey.Text, priv_key)
            'masukkan priv key sbg parameter priv key DSA
            success = dsa.FromPem(priv_key)
            If (success <> True) Then 'mengecek isi/panjangnya
priv key sesuai parameter DSA
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If
            'pastikan private key nya valid/polanya cocok aturan
priv key sebenarnya
            success = dsa.VerifyKey()
            If (success <> True) Then
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If

            'langkah ketiga: masukkan message hash ke algo
```

```
            success = dsa.SetEncodedHash("hex", hashStr)
'masukkan MD e-dok sbg par. DSA
            If (success <> True) Then
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If

            'langkah keempat: sign!
            success = dsa.SignHash() 'proses ttdig
            If (success <> True) Then
                MsgBox(dsa.LastErrorText)
                Exit Sub
            End If

            Dim hexSig As String
            txtSignature.Text = hexSig 'tampilkan ttdig ke teks
box ttdig
            'simpan ttdig ke file
            Dim path1 As String
            Dim nama_sig As String
            path1 = Path.GetDirectoryName(txtDokumen.Text)
'mengetahui lokasi/ folder e-dok
            nama_sig = path1 + Path.DirectorySeparatorChar +
Path.GetFileNameWithoutExtension(txtDokumen.Text) + ".sig"
'menentukan nama file ttdig
            dsa.SaveText(hexSig, nama_sig) 'simpan ttdig ke file
            txtLokasiSign.Text = nama_sig 'menampilkan full path
file ttdig ke teksbox txtLokasiSig

            MsgBox("Selesai menandatangani dengan private key!")
        Else
            MsgBox("E-dokumen atau berkas private key tidak
ditemukan!", MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal
error")
        End If
    End Sub
End Class
```

Lampiran 5 :

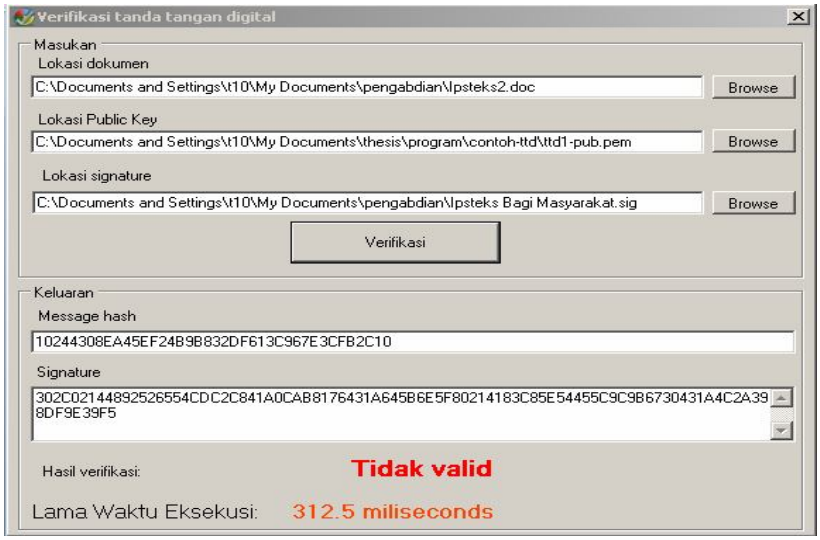
Beberapa contoh perubahan pada isi file e-dokumen dan hasil verifikasinya.

Beberapa contoh perubahan pada isi file “Ipsteks Bagi Masyarakat.doc” selain pada contoh kasus 5 dan hasil tampilan proses verifikasinya yaitu :

a. Perubahan dengan penggantian ukuran font yaitu pada “USULAN..” diganti “USULAN..” yang disimpan sebagai file Ipsteks2.doc :

USULAN PROGRAM IPTEKS BAGI MASYARAKAT
($I_bM$)

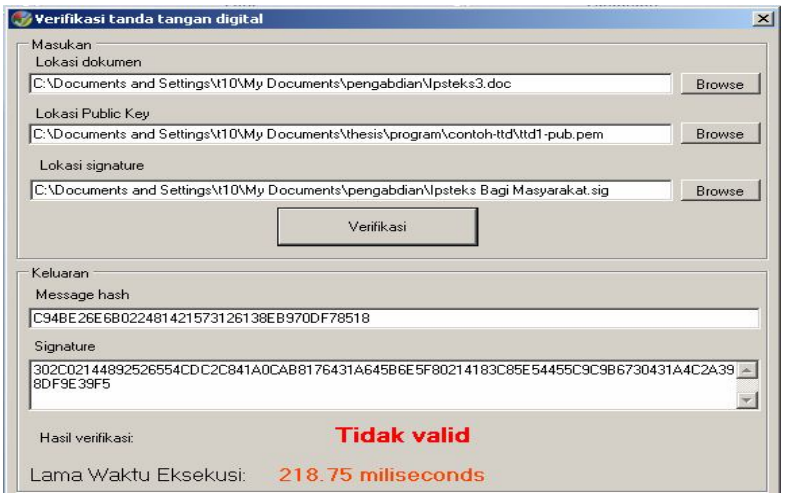
Hasil verifikasinya :



b. Perubahan dengan penggantian warna font yaitu pada “USULAN..” diganti “USULAN..” yang disimpan sebagai file Ipsteks3.doc :

USULAN PROGRAM IPTEKS BAGI MASYARAKAT
($I_bM$)

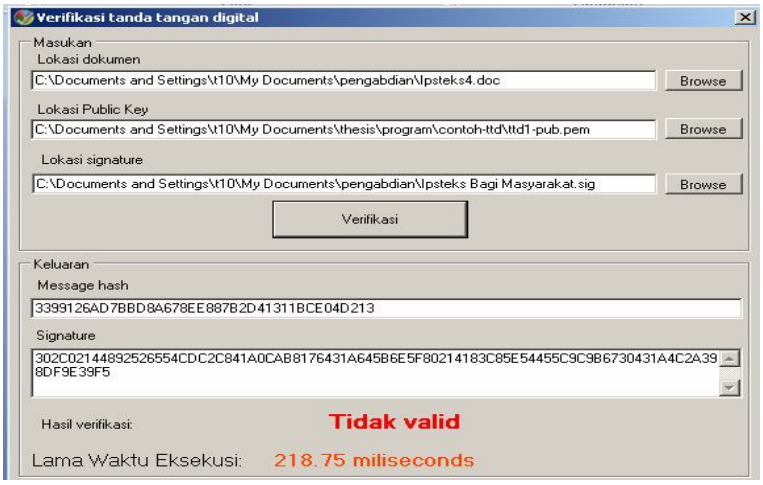
Hasil verifikasinya :




c. Perubahan dengan penghapusan/ penghilangan karakter yaitu penghapusan “$I_bM$” yang disimpan sebagai file Ipsteks4.doc :

USULAN PROGRAM IPTEKS BAGI MASYARAKAT

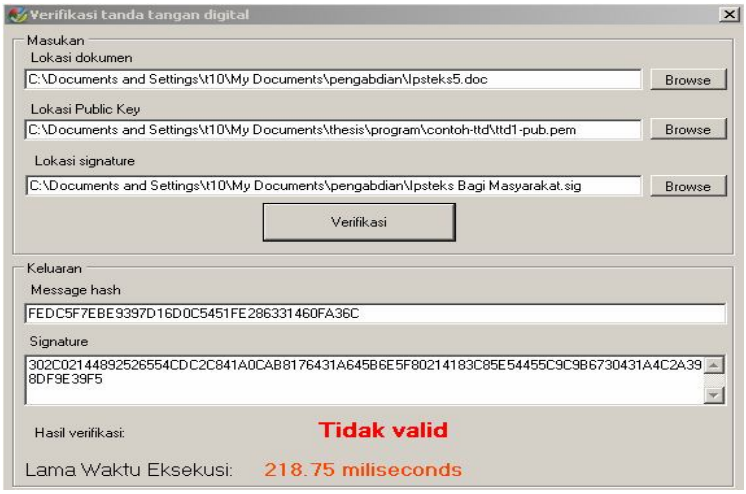
Hasil verifikasinya :




d. Perubahan dengan mengganti posisi karakter yaitu “USULAN PROGRAM…” diganti “PROGRAM USULAN...” yang disimpan sebagai file Ipsteks5.doc :

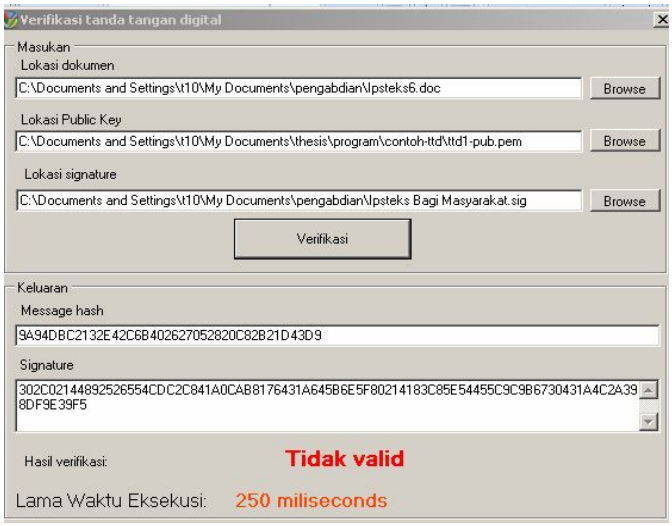
PROGRAM USULAN IPTEKS BAGI MASYARAKAT
(IbM)

Hasil verifikasinya :



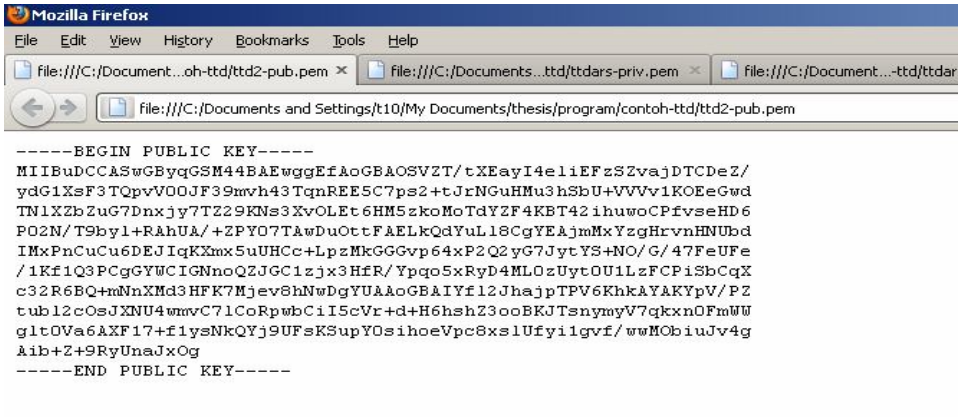
e. Perubahan dengan mengganti ukuran gambar “logo UNAKI” yang diperkecil yang disimpan sebagai file Ipsteks6.doc :


Hasil verifikasinya :



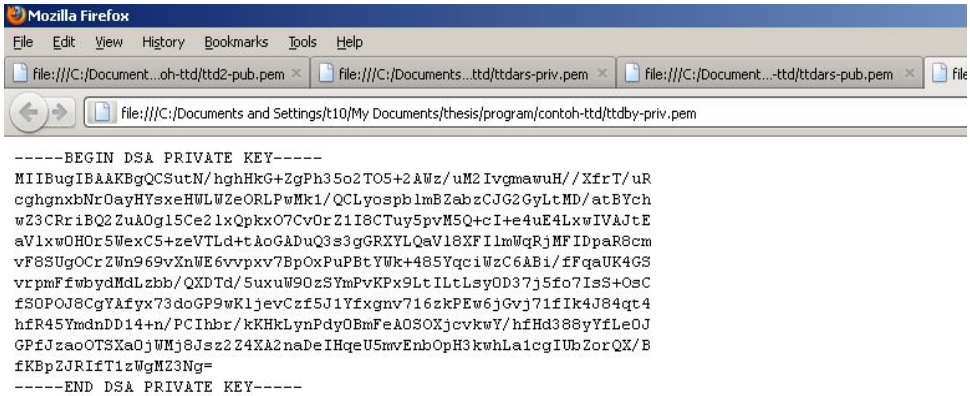
Lampiran 6 :

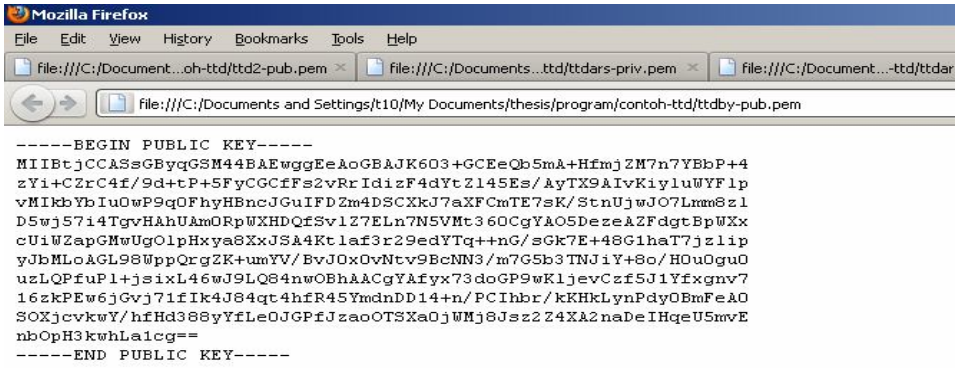
Isi kunci publik ttd2-pub.pem

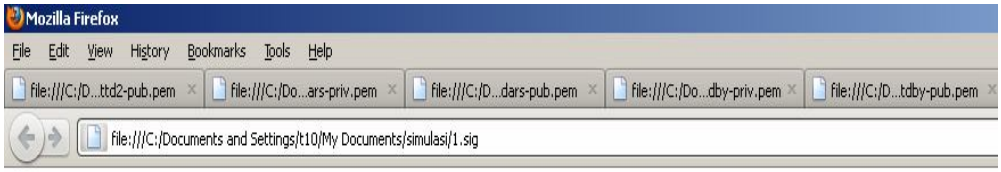



Lampiran 7 :

Isi kunci privat, publik dan signature dari masukan tandatangan offline ttdby.jpg

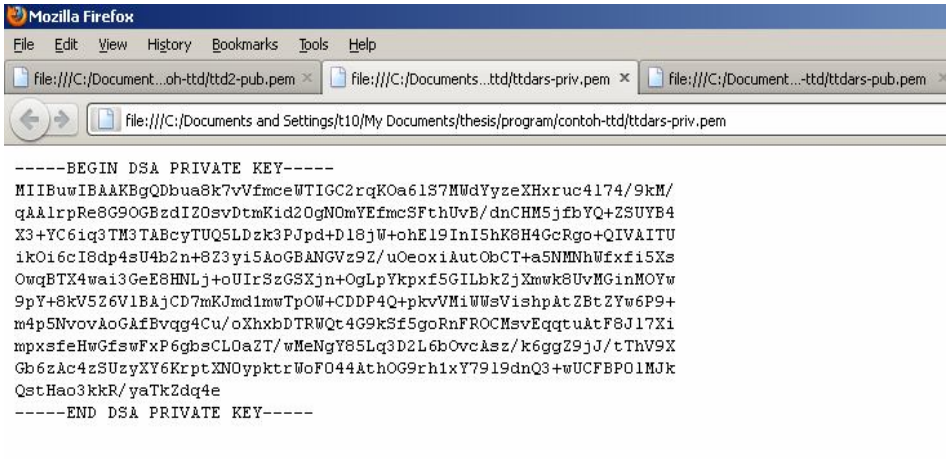


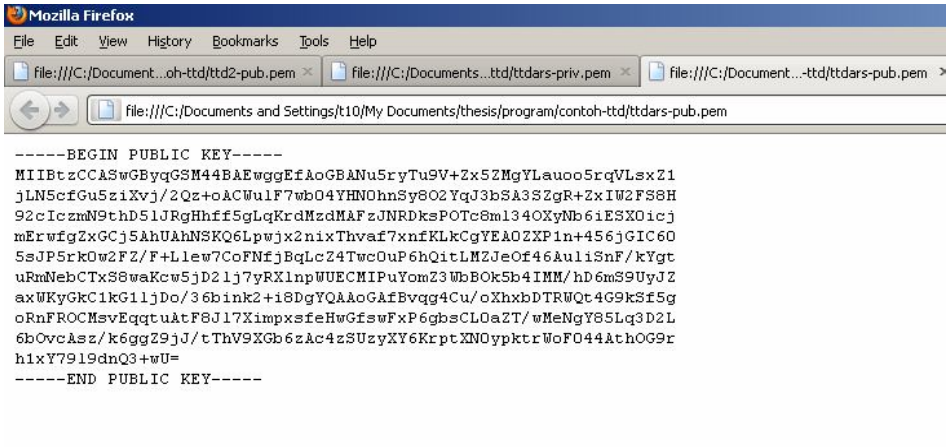



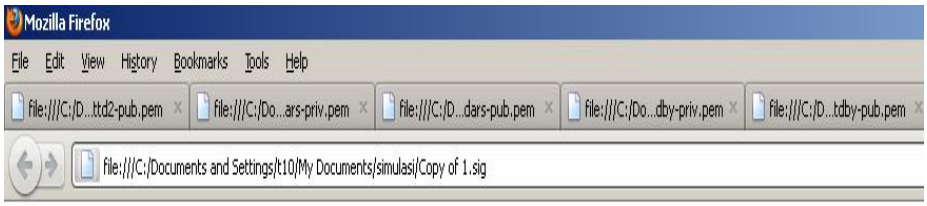



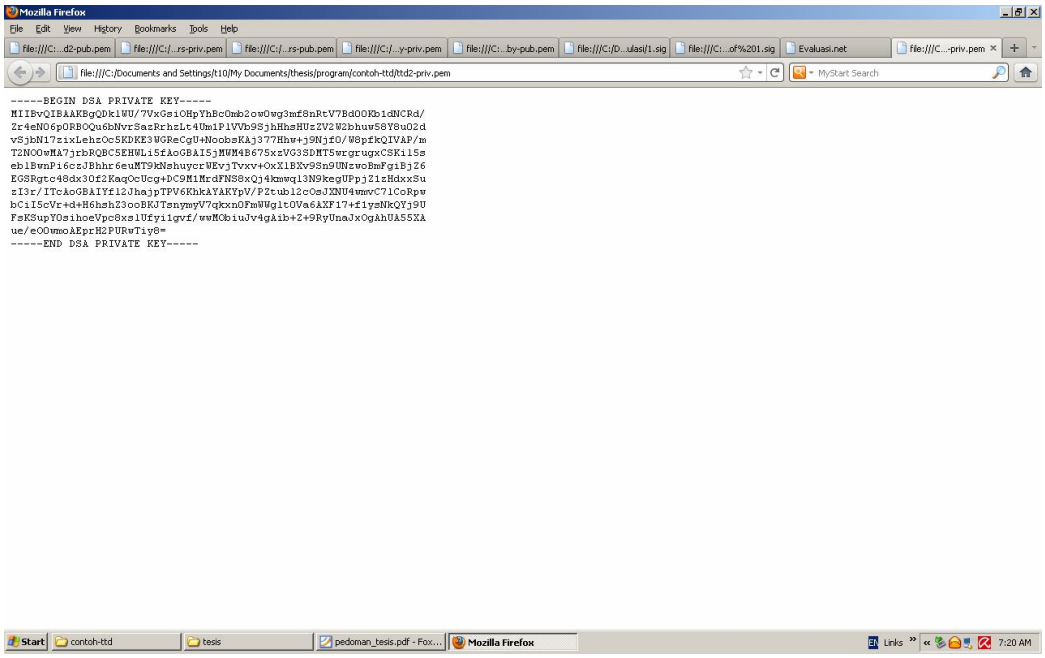

Lampiran 8 :

Isi kunci privat, publik dan signature dari masukan tandatangan offline ttdars.jpg




Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents and Settings/t10/My Documents/thesis/program/contoh-ttd/ttd2-priv.pem
-----BEGIN DSA PRIVATE KEY-----
MIIBvQIBAAKBgQDkiWU/7VxGs1OHpYbBcGmb2ow0wg3mf8nRtV7Bd0OKb1dNCRd/
2r4eNO6pORBOQu6bNvrSazPrhzLt4Um1PlVVb9SjhHhmHUrZV2W2bhuw58Y8uO2d
vSjbN17zixLehzOc5KDKE3WGReCgU+NoobsKAj377Hhw+j9NjfO/W8pfkQIVAP/m
T2NOOwMA7jrbRQBC5EHWLi5fAoGBAI5jMWM4B675xzVG3SDMT5wrgrugxCSKi15s
eblBwnPi6czJBhhr6euMT9kNshuycrWEvjTvxv+OxXlBXv9Sn9UNzwoBmFgiBjZ6
EGSRgtc48dx30f2KaqOcUcg+DC9M1MrdFNS8xQj4kmwq13N9kegUPpjZ1zHdxxSu
zI3r/ITcAoGBAIYfl2JhajpTPV6KhkAYAKYpV/P2tubl2cOsJXNU4wmvC7lCoRpw
bCiI5cVr+d+H6hshZ3ooBKJTsnymyV7qkxnOFmWWglt0Va6AXF17+fiymNkQYj9U
FsKSupYOsihoeVpc8xslUfyiIgvf/wwMObiuJv4gAib+Z+9RyUnaJxOgAhUA55XA
ue/eODwmoAEprH2PURwTiy8=
-----END DSA PRIVATE KEY-----
Start
contoh-ttd
tesis
pedoman_tesis.pdf - Fox...
Mozilla Firefox
Links
7:20 AM