Implementasi Tandatangan Digital Kunci-Publik pada Berkas Gambar dengan Format JPEG

Luqman Abdul Mushawwir – NIM 13507029

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
abdulmushawwir@gmail.com

ABSTRAK

Makalah ini membahas penerapan dan implementasi tandatangan digital kunci-publik untuk berkas gambar berformat JPEG. Implementasi tandatangan digital ini dapat mempermudah seseorang untuk mengotentikasi suatu berkas gambar apakah sudah diubah atau belum, selain itu juga dapat mengetahui pemilik dari suatu berkas gambar, sehingga dapat menghindari pembajakan terhadap berkas gambar dan juga gambar-gambar *hoax*.

Salah satu metode tandatangan digital kunci-publik yang popular adalah mengenkripsi *message digest* dari suatu berkas dengan algoritma kunci-publik RSA. Fungsi *hash* yang dipakai untuk membuat *message digest* pun bermacam-macam, namun dalam makalah ini akan dipakai fungsi *hash* SHA-1.

Berkas gambar yang akan dipakai dalam implementasi tandatangan digital kunci-publik ini adalah berkas gambar dengan format JPEG (Joint Photographic Experts Group) karena ukuran berkas yang kecil dan struktur berkas yang sederhana.

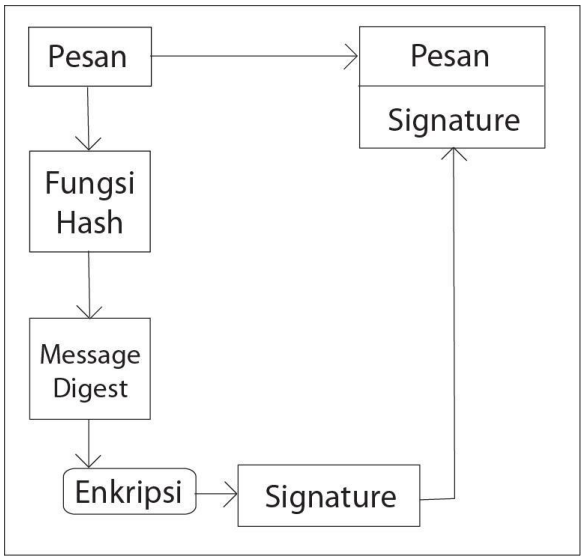
Kata Kunci: Tandatangan Digital, Gambar, JPEG, Algoritma Kunci-Publik, SHA-1, RSA

1. PENDAHULUAN

1.1 Tandatangan Digital

Tandatangan digital adalah suatu metode untuk menandai suatu berkas dengan sesuatu yang unik (sebagaimana definisi tandatangan pada dunia nyata). Tandatangan digital ini berbeda-beda untuk setiap berkas dan pemilik.

Tandatangan digital dibuat dengan membuat *message digest* dari suatu berkas dengan fungsi *hash*. Dengan metode ini, tandatangan setiap berkas yang berbeda adalah unik. Setelah itu, *message digest* ini dienkripsi dengan algoritma kunci-publik, tentu saja dengan kunci privat, sehingga tandatangan ini akan unik untuk setiap pemilik juga. Alur pembuatan tandatangan digital adalah sebagai berikut:

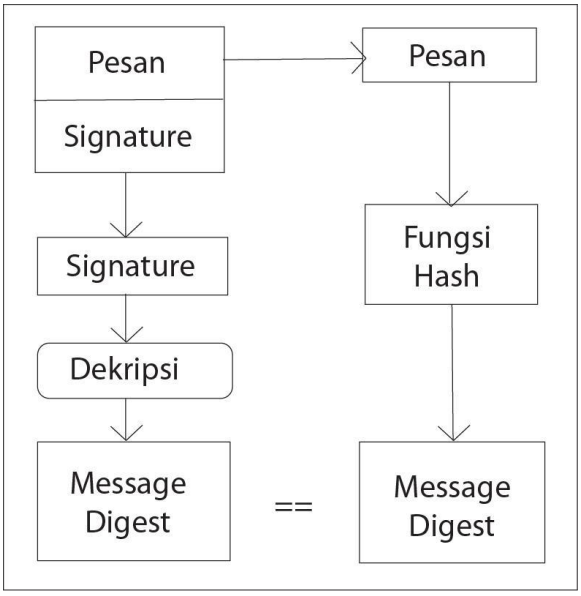


Gambar 1. Alur pembuatan tandatangan digital

Untuk mengecek apakah suatu berkas yang sudah dibubuhi tandatangan digital masih asli atau sudah diubah, berkas yang dibubuhi tandatangan digital diproses sebagai berikut.

Berkas dipisahkan dari tandatangan digitalnya, kemudian berkas yang sudah dipisahkan dibuat *message digest*-nya, sedangkan tandatangan digital yang sudah dipisahkan didekripsi dengan algoritma kunci-publik, dengan kunci publik tentunya, sehingga didapat *message digest* dari berkas saat dibuat tandatangannya (jika menggunakan kunci publik yang benar).

Setelah didapat keduanya, lalu dibandingkan. Jika hasil pembuatan *message digest* dari berkas sama dengan hasil dekripsi tandatangan, berarti berkas masih otentik. Sebaliknya, jika tidak sama berarti berkas sudah tidak otentik, atau kunci yang digunakan salah. Alur pengecekan tandatangan digital adalah sebagai berikut:



Gambar 2. Alur pengecekan tandatangan digital

Fungsi *hash* untuk membuat *message digest* dan algoritma kriptografi kunci-publik yang digunakan bermacam-macam. Fungsi *hash* yang dapat dipakai seperti SHA-0, SHA-1, MD5, dan lain-lain. Algoritma kriptografi kunci-publik pun bermacam-macam, seperti RSA, El Gamal, dll, namun penulis di sini menggunakan fungsi *hash* SHA-1 dan algoritma kriptografi kunci publik RSA.

1.1.1 Fungsi *Hash* SHA-1

Fungsi *hash* adalah suatu fungsi yang menerima masukan berupa *string* sembarang, lalu mentransformasikannya menjadi *string* keluaran yang panjangnya tetap.

Jika dituliskan dalam notasi matematis akan jadi seperti: MD = H(M) oleh fungsi hash sebuah string yang berukuran apapun diubah menjadi message digest yang berukuran tetap (128-512 bit).

Adapun sifat-sifat yang dimiliki oleh fungsi hash adalah sebagai berikut :

- Fungsi H dapat diterapkan pada blok data yang berukuran berapa saja.
- Nilai hash yang dihasilkan memiliki panjang yang tetap.
- Untuk setiap h yang diberikan, tidak mungkin menemukan suatu x sedemikian sehingga H(x)=h. Fungsi H tidak dapat mengembalikan nilai hash menjadi masukan awal.
- Untuk setiap x yang diberikan, tidak mungkin mencari pasangan $x \neq y$ sedemikian sehingga H(x)=H(y).

Ada dua jenis fungsi hash yang sering digunakan hingga sekarang. Fungsi hash yang pertama adalah MD5. Fungsi MD5 dibuat oleh Ron Rivest pada tahun 1994. Nilai hash yang dihasilkan oleh MD5 berukuran 128 bit. Fungsi hash yang lain adalah SHA (Secure Hash Algorithm) dikembangkan oleh NIST sebagai spesifikasi SHS (Secure Hash Standard). SHA sendiri memiliki banyak varian. Salah satunya adalah varian SHA-1. Varian SHA-1 ini menghasilkan nilai hash yang berukuran 160 bit.

Dalam makalah ini, untuk mempermudah, fungsi hash yang akan dipakai adalah algortima SHA-1.

1.1.2 Algoritma Kriptografi Kunci Publik RSA

Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (Massachussets Institute of Technology) pada tahun 1976, yaitu: Ron Rivest, Adi Shamir, dan Leonard Adleman.

Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor – faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Selama pemfaktoran bilangan besar menjadi faktor – faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin.

Algoritma RSA memiliki properti sebagai berikut:

1. p dan q bilangan prima
2. n = p.q
3. $\Phi(n) = (p - 1)(q - 1)$
4. e (kunci enkripsi)
5. d (kunci dekripsi)
6. m (plainteks)
7. c (cipherteks)

prosedur pembangkitan sepasang kunci yang yaitu kunci privat d dan kunci publik e dapat dijelaskan sebagai berikut:

1. Pilih dua buah bilangan prima sembarang, p dan q
2. Hitung n = p.q (sebaiknya $p \neq q$ karena apabila p = q maka n = p2 sehingga p dapat diperoleh dengan menarik akar pangkat dua dari n)
3. Hitung $\Phi(n) = (p - 1)(q - 1)$
4. Pilih kunci publik e yang relatif prima terhadap $\Phi(n)$
5. Bangkitkan kunci privat dengan menggunakan persamaan berikut: $e \cdot d = 1 \pmod{\Phi(n)}$,

perhatikan bahwa $e \cdot d = 1 \pmod{\Phi(n)}$ ekivalen dengan $e \cdot d = 1 + k\Phi(n)$, sehingga secara sederhana d dapat dihitung dengan $d = 1 + k\Phi(n) / e$

hasil dari algoritma di atas yaitu:

- Kunci publik sebagai pasangan (e, n)
- Kunci privat sebagai pasangan (d, n)

Untuk proses enkripsi dan dekripsi algoritma RSA, dapat dijelaskan sebagai berikut:

1. Enkripsi

- Ambil kunci publik penerima pesan e dan mod n
- Nyatakan plainteks m menjadi blok m1, m2, … sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n – 1]
- Setiap blok mi dienkripsi menjadi blok ci dengan

ci = mi e mod n

2. Dekripsi
- Setiap blok cipherteks ci didekripsi kembali menjadi blok mi dengan rumus mi = ci d mod n

1.2 Berkas JPEG

JPEG merupakan suatu standar kompresi berkas gambar yang dibuat oleh Joint Photographic Expert Group, yang mempunyai spesifikasi dari *codec*, yaitu mendefinisikan bagaimana sebuah gambar dikompres menjadi sebuah *stream of bytes* dan mendekompres *stream of* bytes tersebut menjadi sebuah gambar kembali.

Metode kompresi biasanya *lossy*, yang artinya beberapa informasi original dari gambar hilang dan tidak bisa kembali (dapat mempengaruhi kualitas gambar). Ada beberapa variasi pada standar JPEG yang *lossless*, namun tidak didukung secara luas.

Ada pula format "Progressive JPEG", yang datanya dikompres di banyak nilai detail yang semakin tinggi. Ini ideal untuk gambar besar yang akan ditampilkan sambil mengunduh melewati koneksi yang lambat, memungkinkan tampilan gambar yang wajar walaupun hanya menerima sebagian data, namun format ini tidak didukung secara luas.

Ada pula beberapa system pencitraan medis yang membuat dan memproses gambar JPEG 12-bit. Format JPEG 12-bit telah menjadi bagian dari spesifikasi JPEG untuk beberapa waktu, tapi sekali lagi, format ini tidak didukung secara luas.

JPEG mempunyai struktur *header* seperti yang dituliskan pada lampiran (1).

Contoh gambar JPEG adalah sebagai berikut:

Gambar 3. Gambar yang menunjukkan kompresi yang semakin *lossy* dari kiri ke kanan (sumber: Wikipedia)

2. PEMBAHASAN

Sebelum memulai pembahasan mengenai pembuatan, penempelan, dan analisis tandatangan digital, penulis merasa perlu memaparkan mengenai asumsi-asumsi yang dipakai dalam melakukan hal-hal tersebut.

Penulis menggunakan visual C# untuk membuat aplikasi pembuat tandatangan digital. Gambar yang dipakai untuk pengujian tandatangan digital ini adalah gambar berukuran 83.9 KB berformat JPG.

Gambar 4. Gambar yang dipakai dalam pengujian (wilsumjg.jpg)

Algoritma yang digunakan untuk membuat tanda tangan digital adalah fungsi *hash* SHA-1 dan algoritma kriptografi kunci-publik RSA.

Kunci publik yang digunakan untuk algoritma kriptografi kunci-publik adalah kunci publik dengan n = 133, kunci publik = 5, kunci privat = 65.

2.1 Pembuatan Tandatangan Digital

Pembuatan tandatangan digital pada berkas JPEG dilakukan dengan langkah-langkah sebagai berikut:

1. Mengambil informasi dari berkas berupa *array of bytes*.
2. Membuat *message digest* dari berkas tersebut dengan menggunakan fungsi *hash* SHA-1, didapat 160 bit *message digest* dari berkas tersebut.
3. Mengenkripsi *message digest* tersebut dengan algoritma RSA, kunci privat = 65 dan n = 133.

Dengan langkah-langkah seperti di atas, didapat tandatangan digital dari berkas "wilsumjg.jpg" sebagai berikut:

0742385A08074F1B155A075976743908084F07801
5595A4F5A07078015072D598007385A4205384F

2.2 Penempelan Tandatangan Digital pada Berkas JPEG

Suatu berkas gambar dengan format JPEG mempunyai suatu penanda awal dan akhir gambar. Awal gambar, yang sering disebut *Start of Image* (SOI), ditandai dengan kode heksadesimal 0xFFD8. Sedangkan, akhir gambar, yang sering disebut *End of Image* (EOI), ditandai dengan kode heksadesimal 0xFFD9.

Begitu juga dengan berkas "wilsumjg.jpg", yang mempunyai SOI dan EOI seperti itu. Berkas ini dilihat kode heksadesimalnya dan mempunyai kode heksadesimal sebagai berikut:

FF-D8-FF-E0-00-10-4A-46-49-46-…-44-7E-EA-9B-53-F8-E2-8D-DF-FF-D9

Ide untuk menempelkan tandatangan digital adalah menempelkannya di luar kode yang akan dibaca sebagai sebuah gambar, yaitu sebelum SOI atau sesudah EOI. Pembuatan tandatangan digital pun mempunyai keuntungan tersendiri, yaitu hasil akhir tandatangan digital berupa heksadesimal, sehingga dapat "dibongkar pasang" dalam berkas dengan mudah.

Saat mencoba memasukkan tandatangan digital sebelum SOI, ternyata gambar menjadi tidak terbaca dan rusak, walaupun sebenarnya ketika tandatangan itu dihilangkan, gambar menjadi bisa terbaca kembali.

Tandatangan digital akhirnya dimasukkan setelah EOI. Setelah dibubuhkan tandatangan digital pun, berkas gambar tidak mengalami perubahan sama sekali. Di bawah ini adalah gambar "wilsumjg.jpg" setelah dibubuhi tandatangan digital.

Gambar 5. Berkas gambar setelah dibubuhi tandatangan digital setelah EOI-nya

Gambar yang dinamai "wilsumedit.jpg" ini mempunyai kode heksadesimal sebagai berikut:

FF-D8-FF-E0-00-10-4A-46-49-46-…-44-7E-EA-9B-53-F8-E2-8D-DF-FF-D9-07-42-38-5A-08-07-4F-1B-15-5A-07-59-76-74-39-08-08-4F-07-80-15-59-5A-4F-5A-07-07-80-15-07-2D-59-80-07-38-5A-42-05-38-4F

Jika dilihat, gambar tidak mengalami perubahan, namun di dalamnya menyimpan informasi mengenai tandatangan digital dari berkas tersebut.

2.3 Otentikasi Berkas

Otentikasi pada berkas yang sudah dibubuhi tandatangan digital dilakukan dengan cara sebagai berikut:

1. Mengambil informasi dari berkas berupa *array of bytes*.
2. Memisahkan bagian tandatangan digital dari berkas: membaca berkas sampai EOI (FF-D9) lalu memisahkan bagian setelah EOI ke array yang berbeda.
3. Mendekripsi tandatangan digital yang telah dipisahkan dengan algoritma kriptografi kunci-publik RSA dengan n = 133 dan kunci publik = 5.
4. Membuat *message digest* dari bagian SOI sampai EOI dengan fungsi *hash* SHA-1, sehingga dari hasil (3) dan (4) sama-sama menghasilkan 160 bit *message digest*.
5. Membandingkan hasil dari (3) dengan (4). Jika hasilnya sama, berarti berkas gambar masih otentik, sedangkan jika tidak sama, berarti telah dilakukan perubahan pada berkas, atau kunci yang digunakan salah.

2.4 Pengujian dan Analisis pada Berkas dan Tandatangan Digitalnya

Untuk menguji tandatangan digital, maka penulis melakukan beberapa pengujian:

1. Melakukan pengubahan pada berkas yang sudah ditandatangani. Pengubahan ini dilakukan dengan menambahkan garis bawah pada tulisan ber-*aksara* Sunda.
2. Mengubah tandatangan digital dengan menghapus "4F" pada akhir tandatangan digital.
3. Menggunakan kunci publik yang tidak cocok dengan kunci privat yang digunakan untuk membuat tandatangan digitalnya, yaitu n = 3337 dan kunci publik = 3.

Gambar 6. Berkas gambar yang sudah diubah

Hasil pengujian pada masing-masing kasus adalah sebagai berikut:
Pada pengujian (2) dan (3), tandatangan digital pada berkas dapat didekripsi, namun *message digest* yang didapat dari hasil dekripsi berbeda dengan *message digest* yang didapat dari hasil *hash* dari berkas gambar.

Pada pengujian (1), tidak terdeteksi tandatangan digital pada berkas. Ternyata, setelah dicek, ketika berkas tersebut diubah dan disimpan, *header* pada gambar tersebut ditulis kembali, sehingga berkas tandatangan yang ditulis setelah SOI tidak ikut ditulis kembali.

Kode heksadesimal pada gambar yang sudah diubah adalah sebagai berikut:

FF-D8-FF-E0-00-10-4A-46-49-46-00-01-01-01-00-60-00-60-…-20-E3-68-A5-74-3D-8F-FF-D9

Seperti yang dapat dilihat, pada kode heksadesimal di atas, gambar tetap diawali dengan SOI (FF-D8) dan diakhiri dengan EOI (FF-D9). Hal ini membuktikan bahwa tandatangan digital yang ditempel dengan cara dibubuhkan setelah EOI tidak kuat, karena akan hilang hanya dengan pengubahan kecil.

2.5 Potensi Pengembangan

Ada beberapa hal yang dapat dikembangkan, baik dari metode pembuatan tandatangan digital, metode penempelan tandatangan digital ke berkas, dan lain-lain. Di bawah ini adalah beberapa potensi pengembangan metode implementasi tandatangan digital pada berkas JPEG yang dipikirkan oleh penulis.

1. Metode pembuatan tanda tangan digital dapat dikembangkan agar menjadi lebih simpel, aman, dan lebih cepat dalam pembuatannya.

2. Metode penempelan tandatangan digital pada berkas seperti yang dijelaskan pada subbab 2.2 tidak kuat, sehingga diperlukan pengembangan pada metode penempelan tandatangan digital pada berkas.

   Metode penempelan yang memungkinkan adalah dengan menggunakan *comment* pada struktur *header* berkas gambar yang ditempelkan tandatangan (kode FF-FE dst.), namun masih perlu dipikirkan bagaimana penempelannya, karena untuk membubuhkan *comment*, membutuhkan ukuran *comment*, dan lain-lain

3. Tandatangan digital dibubuhkan secara eksplisit ke gambar, jadi seperti *bar code* atau *watermark* yang dibubuhkan ke gambar, sehingga terlihat wujud tandatangan digital itu sendiri yang berbeda-beda setiap gambar dan pemilik.

   Sebagai tambahan, walaupun tandatangan digital itu dibubuhkan secara eksplisit, namun tetap saja tandatangan digital tersebut harus dibubuhkan secara implisit, menghindari penghapusan pada tandatangan digital yang eksplisit.

Mungkin masih banyak lagi yang bisa dikembangkan terkait implementasi tandatangan digital pada berkas JPEG, atau mungkin nantinya bisa merambah ke format berkas yang lain, atau bukan hanya gambar.

Potensi-potensi pengembangan di atas merupakan hasil pemikiran penulis yang mengeksplorasi terkait hal ini hanya sedikit dan masih di permukaan. Jika eksplorasi dilakukan lebih mendalam, pasti aka nada ide-ide yang lebih banyak untuk dikembangkan.

3. KESIMPULAN

Berbagai macam berkas dapat dibuat tandatangan digitalnya, yaitu dengan cara membuat *message digest* dengan fungsi *hash* dan mengenkripsi *message digest* tersebut dengan algoritma kriptografi kunci publik.

Pada berkas gambar, ada beberapa cara untuk membubuhkan tandatangan digital ke berkas secara implicit, salah satunya adalah membubuhkan tandatangan setelah EOI (*end of image*).

Namun, pada pembubuhan tandatangan setelah EOI, tandatangan pada berkas akan hilang saat gambar di-*edit.* Oleh karena itu, harus dicari metode untuk membubuhkan tanda tangan yang kuat.

UCAPAN TERIMA KASIH

Ucapan terima kash dan apresiasi yang sebesar-besarnya kepada:

1. Allah SWT.
2. Bapak Rinaldi Munir, M.T. yang telah memberikan kuliah IF3058 Kriptografi.
3. Anggrahita Bayu Sasmita dan Mochamad Reza Akbar, teman sekelompok tugas besar yang memberikan penulis inspirasi untuk topik makalah ini.

Dan semua pihak yang relah mendukung, baik secara materi ataupun moral, yang tidak bisa disebut satu per satu.

DAFTAR PUSTAKA

[1] Bruce Schneier, “Applied Cryptography - Second Edition”. John Wiley & Sons, Inc, 1996

[2] Menezes, P. Van Oorschot, S. Vanstone, “Handbook of Applied Cryptography”. CRC Press, Inc, 1997

[3] Munir, Rinaldi. Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung, 2004

[4] William B. Pennebaker, Joan L. Mitchell, “Jpeg: Still Image Data Compression Standard” (3rd ed.). Springer.

[5] www.**convision**.de

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2010

Luqman Abdul Mushawwir
13507029

LAMPIRAN

1. Struktur *header* sebuah file JPEG

Marker	Value	Size (bytes)	Description
SOI	0xFF,0xD8	2	Start Of Image
APP0	0xFF,0xE0	18	JFIF (JPEG File Interchange Format)
COM	0xFF,0xFE, 0x00, 0xFA	4, total 252	Comment start
		4	JPEG Flags bit field: Bit 1: Indicate if daylight saving time is used
		4	captured time (format time_t)
		4	captured ticks
		4	JPEG size
		4	Camera Number
		4	total motion
		24	camera name
		200	motion
DQT	0xFF,0xDB	69	Quantization Table - Luminance (Y)
DQT	0xFF,0xDB	69	Quantization Table - Chrominance (Cb/Cr)
SOF	0xFF,0xC0	19/13	Start Of Frame (19 color/13 black&white)
DHT	0xFF,0xC4	33	Huffman Table - Luminance (Y) - DC Diff
DHT	0xFF,0xC4	183	Huffman Table - Luminance (Y) - AC Coeff
DHT	0xFF,0xC4	33	Huffman Table - Chrominance (Cb/Cr) - DC Diff
DHT	0xFF,0xC4	183	Huffman Table - Chrominance (Cb/Cr) - AC Coeff
SOS	0xFF,0xDA	14/10	Start Of Scan (14 color/13 black&white)
...	...	...	Huffman coded image
EOF	0xFF,0xD9	2	End Of Image