Studi dan Implementasi Cramer Shoup untuk Tanda Tangan Dijital

Ratna Mutia S / 13503086

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl Ganesha 10, Bandung
E-mail : if13086@students.if.it.ac.id

Abstrak

Tanda tangan digital merupakan suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan dan digunakan untuk menjamin integritas data, membuktikan asal pesan, dan anti-penyanggahan [1]. Pemberian tanda tangan dijital dapat memanfaatkan sistem kriptografi kunci-publik, salah satunya adalah Cramer Shoup. Cramer Shoup merupakan sistem kriptografi kunci-publik asimetris yang diajukan oleh Ronald Cramer dan Victor Shoup pada tahun 1998. Cramer Shoup terbukti sebagai skema efektif pertama yang tahan terhadap serangan *adaptive chosen ciphertext attack* dibandingkan dengan sistem kriptografi yang telah ada saat itu [2]. Di satu sisi, sebuah sistem kriptografi dapat dikatakan baik untuk digunakan dalam tanda tangan dijital apabila sistem tersebut sensitif terhadap modifikasi pesan. Dengan kemampuan Cramer Shoup yang tahan terhadap serangan diharapkan algoritma tersebut dapat dimanfaatkan untuk membentuk tanda tangan dijital yang baik.

Sebuah aplikasi bernama "Cramer Shoup on Digiture" dikembangkan untuk menguji implementasi Cramer Shoup pada tanda tangan dijital. Sebagai batasan, aplikasi tersebut hanya mampu menangani pesan teks. Pengujian kemudian dilakukan dengan melakukan modifikasi terhadap pesan maupun tanda tangan dijital. Pengujian juga dilakukan dengan membandingkan performansi antara penggunaan Cramer Shoup dengan algoritma kunci publik lain yaitu RSA.

Kesimpulan yang diperoleh adalah Cramer Shoup berhasil dalam pengujian modifikasi sehingga dapat digunakan sebagai dasar skema tanda tangan dijital yang aman

Kata kunci : *Cramer Shoup, tanda tangan digital, adaptive chosen ciphertext attack, kunci publik*

1. Pendahuluan

Tujuan dari tanda tangan dijital adalah menjaga integritas data dan memungkinkan penerima untuk melakukan otentikasi terhadap sumber pesan dan kebenarannya.

Algoritma kunci publik dapat digunakan untuk tanda tangan dijital. Cramer Shoup sebagai salah satu kunci publik yang terbukti aman dalam menghadapi serangan *adaptive chosen ciphertext attack* dapat menjadi salah satu alternatif.

Makalah ini akan meninjau tanda tangan dijital terlebih dahulu sebagai dasar pembahasan. Kemudian dilanjutkan dengan penjelasan algoritma Cramer Shoup pada proses enkripsi dan dekripsi. Selanjutnya dua buah skema tanda tangan dijital yang didasarkan pada algoritma Cramer Shoup akan diuraikan untuk memperlihatkan bagaimana algoritma Cramer Shoup diterapkan pada tanda tangan dijital.

Pembahasan berlanjut dengan menguraikan hasil implementasi program "Cramer Shoup on Digiture" dan menguraikan hasil pengujian serta analisa yang telah dilakukan.

Terakhir adalah membandingkan Naïve Cramer Shoup dengan RSAdari segi performansi.

2. Tanda Tangan Dijital

Kriptografi telah membantu dalam merahasiakan pesan dan juga keamanan data yang mencakup keabsahan pengirim, keaslian pesan, dan anti-penyanggahan. Tanda tangan dijital kemudian dikembangkan untuk mendukung keamanan data tersebut dalam bentuk otentikasi pesan. Teknik otentikasi dalam komunikasi data diartikan sebagai prosedur yang digunakan untuk membuktikan keaslian pesan atau identitas pemakai [1].

Tanda tangan yang telah digunakan dalam hal pengesahan dokumen kertas seperti ijazah, sertifikat, maupun surat kemudian menjadi dasar pengembangan tanda tangan dijital dalam dokumen elektronis.

Sebuah tanda tangan dijital perlu untuk memenuhi kriteria keamanan tertentu sehingga dapat menjamin kebenaran proses otentikasi. Seiring dengan berkembangnya berbagai macam algoritma, berbagai skema tanda tangan dijital diajukan oleh para ahli. Dibandingkan dengan algoritma kriptografi simetri yang tidak dapat digunakan untuk mengatasi masalah anti-penyanggahan pada keamanan data, tanda tangan dijital lebih cocok menggunakan sistem kriptografi kunci publik.

2.1 Pengertian tanda tangan dijital

Alice mempunyai sebuah pesan yang akan diberikan kepada Bob, Carla, dan David. Alice ingin agar setiap penerima benar-benar menerima pesan yang ia kirimkan dan setiap penerima dapat membuktikan bahwa Alice benar-benar pengirim pesan tersebut. Alice juga berharap tidak ada pihak lain yang memodifikasi pesan tersebut. Kalaupun ada pihak yang memodifikasi pesa, pihak penerima dapat mengetahui hal tersebut.

Sedangkan bagi Bob, Carla, mupun David hal yang mereka butuhkan adalah Alice tidak dapat menyanggah seandainya suatu saat mereka menerima pesan dengan Alice yang teridentifikasi sebagai pengirimnya.

Untuk memenuhi kebutuhan-kebutuhan tersebut, Alice perlu menandatangani setiap pesan yang ia kirimkan dan pihak penerima dapat melakukan autentifikasi tanpa Alice membagi kunci rahasia yang ia miliki serta tanpa adanya interaksi antara Alice dan pihak penerima.

Ilustrasi Alice tersebut merupakan bagaimana tanda tangan dijital diterapkan untuk keamanan data. Tanda tangan dijital bukanlah tanda tangan yang di-dijitasi dengan *scanner* kemudian di-*embed* ke dalam pesan. Tanda tangan dijital merupakan suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan [1]. Dengan demikian tanda tangan dijital dapat menjamin integritas data dan membuktikan asal pesan.

Skema tanda tangan dijital terdiri dari sebuah tuple (*M, K, G, S, V*) dengan

- *M* - pesan
- *K* - kunci privat dan publik (*pk, sk*)
- *G* - PPT algoritma pembangkit kunci
- *S* - PPT algoritma untuk tanda tangan dengan masukan kunci privat dan pesan
- V - pemverifikasi dengan Vpk(m,Ssk(m)) = 1 jika (pk,sk) adalah kunci valid

[5] Setelah didapat pasangan kunci privat dan publik melalui G, pesan dikenakan S dengan kunci privat sk. S kebanyakan merupakan algoritma enkripsi kunci publik. Pesan yang telah ditandatangani sampai ke penerima. Untuk mengetahui asal pesan dan apakah pesan masih asli, dapat dilakukan proses verifikasi dengan algoritma V dengan parameter kunci publik pengirim pk.

2.2 Kriteria keamanan tanda tangan dijital

Sebuah skema tanda tangan dijital perlu mampu menangani masalah keabsahan pengirim, keaslian pesan, dan anti penyanggahan. Keabsahan pengirim berkaitan dengan kebenaran identitas pengirim untuk menjawab apakah pesan yang diterima tersebut berasal dari pengirim yang sesungguhnya. Dengan kata lain, jika pesan tersebut dikatakan berasal dari Alice, apakah pesan tersebut memang berasal dari Alice. Hal tersebut dapat dicek melakui penggunaan kunci privat pada proses penanda tanganan serta kunci publik pada proses verifikasi. Pada proses pembangkitan kunci, kunci yang dihasilkan merupakan kunci privat dan publik yang berpasangan secara unik. Misalkan terdapat pihak selain Alice yang mengirim pesan yang

bertanda tangan ke Bob, kemudian Bob membuka pesan tersebut dengan kunci privat Alice. Maka yang terjadi seharusnya proses otentikasi pesan gagal karena pesan bukan berasal dari Alice.

Keaslian pesan berkaitan dengan keutuhan pesan untuk menjawab apakah pesan yang diterima tidak mengalami perubahan. Jika pesan yang dikirim oleh Alice dimodifikasi oleh Carla dan sampai kepada Bob, maka proses otentikasi yang dilakukan Bob harus menyatakan bahwa pesan telah dimodifikasi sehingga proses otentikasi gagal. Walaupun modifikasi yang dilakukan Carla hanya sedikit perubahan. Yang dimaksud modifikasi misalkan, pada arsip teks, penambahan, penghapusan, maupun pengubahan satu karakter pada pesan. Modifikasi pada bagian tanda tangan dijital juga akan menyebabkan otentikasi gagal.

Anti-penyanggahan menjamin pengirim tidak dapat menyangkal tentang isi pesan atau fakta bahwa ia yang mengirimkan pesan. Jika keabsahan pengirim dan keaslian pesan dapat diverifikasi, maka pengirim tidak dapat melakukan sanggahan terhadap pesan yang dikirim.

Algoritma yang diterapkan pada saat penenda tanganan juga perlu tahan terhadap serangan-serangan kriptanalis. Semakin tahan suatu skema tanda tangan dijital maka keotentikan pesan semakin terjaga.

2.3 Tanda tangan dijital dengan algoritma kunci publik dan hash

Algoritma kunci-publik dapat digunakan dalam tanda tangan dijital. Konsep kriptografi kunci publik adalah bahwa kunci kriptografi dibuat sepasang, yaitu untuk enkripsi dan dekripsi. Kunci enkripsi merupakan kunci publik yang tidak bersifat rahasia. Sedangkan kunci dekripsi bersifat rahasia dan dikenal sebagai kunci privat.

Penggunaan kunci privat dan kunci publik untuk merahasiakan pesan sedikit berbeda dengan penggunaan sepasang kunci tersebut pada tanda tangan dijital. Untuk tujuan merahasiakan pesan, jika Alice ingin mengirim pesan pada Bob, maka Alice perlu mengetahui kunci publik Bob lalu menggunakannya untuk mengenkripsi pesan. Hanya Bob kemudian yang dapat mendekripsi pesan karena ia mempunyai kunci privat. Sedangkan dalam hal penandatanganan pesan, kunci privat Alice digunakan sebagai parameter dalam proses penandatanganan. Kemudian Bob memverifikasi pesan menggunakan kunci publik Alice.

Sebagai ilustrasi, misal Alice akan mengirim pesan M *M* adalah pesan yang akan dikirim. Sidik dijital *S* untuk pesan *M* diperoleh dengan mengenkripsikan *M* dengan menggunakan kunci rahasia (*SK*) Alice,

$$S = E_{SK}(M) \qquad (1)$$

E adalah fungsi enkripsi dari algoritma kunci-publik. Selanjutnya, S dikirim melalui saluran komunikasi.

Di tempat Bob sebagai penerima pesan, pesan dibuktikan kebenaran sidik dijitalnya dengan menggunakan kunci publik (*PK*) Alice,

$$M = D_{PK}(S) \qquad (2)$$

D adalah fungsi enkripsi dari algoritma kunci-publik. Sidik dijital *S* dikatakan valid apabila pesan M yang dihasilkan merupakan pesan yang mempunyai makna. [1]

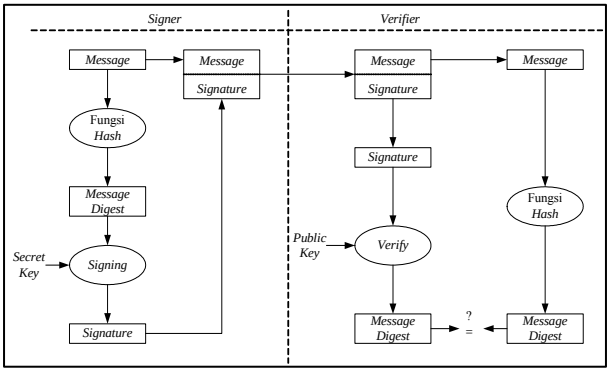
Algoritma kunci publik yang telah banyak diimplementasikan dalam tanda tangan dijital adalah RSA dan El Gamal.

Algoritma enkripsi dan dekripsi pada RSA identik dengan demikian proses penandatanganan dan verifikasi juga identik.

Terdapat pula algoritma yang dikhususkan untuk tanda tangan dijital, yaitu *Digital Signature Algorithm* (DSA). DSA merupakan bakuan (*standard*) untuk *Digital Dignature Standard* (DSS). Pada DSA, algoritma *signature* dan verifikasi berbeda.

Fungsi hash satu arah -seperti MD5, SHA, RIPE-MD, N-hash, dan lain-lain- dapat diterapkan pada pesan untuk menghasilkan message digest (MD) atau pesan terkompresi. Barulah MD tersebut dikenakan algoritma penandatangan. Dengan fungsi hash, tanda tangan yang

dihasilkan akan lebih singkat sehingga tidak terlalu banyak menambah ukuran pesan terkirim.



Gambar 1 Otentikasi tanda tangan dijital dengan fungsi *hash* satu-arah dan kunci publik

Proses penandatanganan serta verifikasi menggunakan kunci publik dan fungsi hash adalah sebagai berikut [1] :
Pesan diubah menjadi bentuk yang ringkas yang disebut *message digest* dengan mentransformasikan pesan *M* menggunakan fungsi *hash* satu-arah (*one-way*) *H*,

$$MD = H(M) \quad (3)$$

Sembarang pesan yang berukuran apapun diubah oleh fungsi *hash* menjadi *message digest* yang berukuran tetap (umumnya 128 bit).

Selanjutnya, MD dienkripsikan dengan algoritma kunci-publik menggunakan kunci rahasia (SK) pengirim menjadi sidik dijital *S*,

$$S = E_{SK}(MD) \quad (4)$$

Pesan *M* disambung dengan sidik dijital *S*, lalu keduanya dikirim melalui saluran komunikasi . Pesan *M* sudah ditandatangani oleh pengirim dengan sidik dijital *S*.

Di tempat penerima, pesan diverifikasi untuk dibuktikan keotentikannya dengan cara berikut:

1. Sidik dijital *S* didekripsi dengan menggunakan kunci publik (*PK*) pengirim pesan, menghasilkan *message digest* semula, *MD*, sebagai berikut:

   $$MD = D_{PK}(S) \quad (5)$$

2. Pengirim kemudian mengubah pesan *M* menjadi *message digest MD'* menggunakan fungsi *hash* satu-arah yang sama dengan fungsi *hash* yang digunakan oleh pengirim.
3. Jika *MD' = MD*, berarti pesan yang *diterima* otentik dan berasal dari pengirim yang benar.

2. 5 Tanda tangan dijital pada dokumen elektronis

Dokumen elektronis dapat diberi tanda tangan dijital untuk keperluan otentifikasi. Tanda tangan dijital dapat diletakan pada awal atau akhir dokumen. Isi dokumen harus dapat diektrakai sehingga penerima dapat membaca isi dokumen seutuhnya. Untuk membedakan bagian tanda tangan dijital dengan isi dokumen, dapat digunakan tag pemisah.

Contoh penggunaan tanda tangan dijital pada dokumen elektronis adalah perangkat lunak Adobe Reader yang telah menyediakan fasilitas tanda tangan dijital sehingga file berekstensi pdf dapat diberi tanda tangan dijital pada area yang diinginkan pengguna.

Tujuan tanda tangan dijital bukanlah pada merahasiakan pesan pada dokumen elektronis, tapi memberi jaminan kevalidan dokumen dan asal dokumen.

Contoh pemberian tanda tangan dijital pda sebuah dokumen elektronis adalah sebagai berikut [1] :

Nilai *r*	#10 #13
Nilai *s*	#10 #7
Isi dokumen semula	
#26	← akhir arsip (EOF)

Pada contoh tersebut, skema yang digunakan adalah DSA. Tanda tangan dijital diletakan di awal dokumen yaitu nilai r dan nilai s. Perlu

diperhatikan bahwa tag pemisah antar tanda tangan dijital perlu dibuat sedemikian rupa sehingga mudah dikenali.

3. Cramer Shoup

Cramer Shoup merupakan algoritma kunci publik asimetris dan menjadi skema efektif pertama yang terbukti tahan terhadap serangan *adaptive chosen ciphertext attack* berdasar asumsi kriptografi standar.

Algoritma ini diciptakan oleh Ronald Cramer dan Victor Shoup pada tahun 1998. Pengembangan Cramer Shoup dikembangkan dari algoritma Elgamal. Berbeda dengan Elgamal yang masih rentan terhadap serangan, Cramer Shoup mempunyai elemen tambahan yaitu jaminan terhadap ketahanan dari berbagaii serangan keamanan. Ketahanan tersebut dicapai melalui penggunaan fungsi hash yang memiliki karakteristik *collision-resistant* dan komputasi tambahan sehingga menghasilkan ciphertext yang lebih panjang dibanding Elgamal.

3.1 Adaptive Chipertext Chosen Attacks

Keamanan Cramer Shoup secara formal didefinisikan sebagai "*indistinguishability under adaptive chosen ciphertext attack*" (IND-CCA2) atau dapat diartikan bahwa Crame Shoup tidak mungkin dapat diserang oleh *adaptive chosen ciphertext attack*. Definisi keamanan tersebut dikenal sebagai definisi keamanan yang terkuat pada algoritma kunci publik.

Adaptive chosen chipertext attack mengasumsikan bahwa pihak penyerang memiliki akses terhadap dekripsi oracle yang akan mendekripsikan chipertext manapun menggunakan skema kunci dekripsi rahasia. *Adaptive* memiliki arti bahwa penyerang memiliki akses terhadap dekripsi oracle baik sebelum maupun sesudah penyerang mengobservasi target chipertext yang spesifik untuk dikenakan serangan. Hal ini berbeda dengan sistem serangan yang lemah yang hanya memungkinkan penyerang mengakses dekripsi oracke sebelum mengobservasi chipertext yang menjadi target.

Jenis serangan *Adaptive chosen chipertext attack* ini pertama kali diperkenalkan oleh Daniel Bleichenbacher di akhir tahun 1990 dengan mendemostrasikan serangan terhadap server SSL yang menggunakan algoritma RSA.
Cramer-Shoup bukanlah skema pertama yang diajukan untuk menyediakan keamanan terhadap serangan *adaptive chosen ciphertext attack*. Naor-Yung, Rackoff-Simon, dan Dolev-Dwork-Naor mengajukan konversi skema standar (IND-CPA) menjadi skema IND-CCA1 and IND-CCA2 . Teknik tersebut aman dibawah kumpulan standar asumsi kriptografi. Akan tetapi pembuktian didasarkan pada teknik *zero-knowledge proof* sehingga tidak efisien dalam biaya komputasi dan ukuran chipertext.

Pendekatan lain dilakukan oleh Bellare/Rogaway's OAEP dan Fujisaki-Okamoto yang mencapai konstruksi yang efisien menggunakan random oracle. Hanya saja, untuk mengeimplementasika skema tersebut membutuhkan pengganti beberapa fungsi seperti fungsi hash. Pada akhirnya, skema tersebut dinyatakan tidak aman.

Proses *adaptive chosen chipertext attack* dapat diilustrasikan sebagai berikut :
Pertama, algoritma pembangkit kunci dijalankan.

Kedua, pihak penyerang membuat query yang bervariasi pada dekripsi oracle untuk mendekripsi chipertext yang telah dipilih.

Selanjutnya, penyerang memilih dua pesan yaitu m_o dan m_1. Kemudian kedua pesan tersebut dikirimkan pada enkripsi oracle. Enkripsi oracle kemudian memilih bit $b \in \{0,1\}$ secara random kemudian mengenkripsi m_b. Ciphertext yang berkorenponden kemudian dikirim ke penyerang.

Setelah pihak penyerang menerima chipertext dari enkripsi oracle, penyerang melanjutkan melakukan query pada dekripsi oracle, dengan output yang dihasilkan berbeda dengan enkripsi oracle.

Pada akhirnya, penyerang mengoutputkan $b' \in \{0,1\}$ sebagai terkaan terhadap nilai b. Jika probabilitas $b' = b$ adalah $\frac{1}{2} + e$ maka *advantage* penyerang didefinisikan sebagai e.

Sebuah *cryptosystem* dapat dikatakan aman terhadap *adaptive chosen ciphertext attack* jika

nilai *advantage* dari berapapun waktu polinomial penyerang dapat ditiadakan.

Cramer Shoup dengan berbagai teknik pembuktian, telah dinyatakan sebagai sebuah cryptosystem, dengan skema yang efektif, yang terbukti tahan terhadap serangan *adaptive chosen ciphertext attack*

3.2 Algoritma Pembangkitan Kunci

Pembangkit kunci memiliki proses sebagai berikut :

1. Alice membangkitkan elemen $g_1,g_2 \in G$. G merupakan kelompok bilangan prima q yang berukuran besar.
2. Alice memilih enam bilangan acak (x_1,x_2,y_1,y_2,z) dari {0,….,q-1}
3. Alice menghitung :

   $c = g_1^{x1} g_2^{x2}$
   $d = g_1^{y1} g_2^{y2}$
   $h = g_1^{z}$

4. Alice mempublikasikan (g_1,g_2,c,d,h,H) sebagai kunci publik. H adalah fungsi hash satu arah yang dipilih untuk digunakan pada proses selanjutnya.
5. Alice menyimpan (x_1,x_2,y_1,y_2,z) sebagai kunci privat.

3.3 Algoritma enkripsi

Untuk mengenkripsi pesan, Bob perlu melakukan proses sebagai berikut :

1. Bob mengkonversikan pesan m menjadi elemen G.
2. Bob memilih bilangan acak r dari {0,….,q-1} kemudian menghitung nilai u_1, u_2, e, α, dan v dengan menggunakan kunci publik Alice (g_1,g_2,c,d,h,H) sebagai berikut :

   $u_1 = g_1^{r}$
   $u_2 = g_2^{r}$
   $e = h^{r} m$
   $\alpha = H(u_1, u_2, e)$
   dengan H sebagai fungsi hash yang *collision-resistant*
   $v = c^{r} d^{r\alpha}$

3. Bob mengirim chipertext (u_1,u_2,e,v) pada Alice.

3.4 Algoritma dekripsi

Alice sebagai penerima pesan dari Bob kemudian melakukan dekripsi terhadap (u_1,u_2,e,v) menggunakan kunci privat Alice (x_1,x_2,y_1,y_2,z) dengan proses sebagai berikut :

1. Alice menghitung nilai α
   $\alpha = H(u_1, u_2, e)$
2. Jika hasil $u_1^{x1+y1\alpha} \; u_2^{x2+y2\alpha} = v$ maka $m = e/u_1^{z}$
   Akan tetapi jika hasil tidak sama dengan v, maka proses akan berhenti dan tidak menampilkan pesan.

3.5 Detail dan Variasi Implementasi Cramer Shoup

Terdapat beragam variasi yang dapat diterapkan pada skema dasar Cramer Shoup. Semua variasi tersebut dapat dipilih sesuai kebutuhan.

3.5.1 Implementasi sederhana

Bilangan prima yang besar yaitu p dipilih dengan p – 1 = 2q, q adalah prima. G merupakan *subgroup* q di Z_p^*.

Pesan yang akan dienkripsi dibatasi menjadi elemen dari {1,…q} dan mengencode pesan dengan mengkuadratkan dengan modulo p menhasilkan elemen G.

Untuk fungsi hash, SHA dapat digunakan dan menghasilkan ausmsi *collision resistance*. Aagr tidak menghasilkan biaya komputasi yang mahal, bit string yang akan dikenakan fungsi hash dibagi menjadi urutan $(a_1, a_2,…a_k)$ dengan setiap a_i bagian elemen {0,..q-1}. Kemudian pilih $h_1,…h_k$ pada G secara random untuk fungsi hash.

3.5.2 Implementasi hybrid

Implementasi hybrid digunakan untuk mencapai cara yang lebih efesien dan fleksibel dalam mengencode pesan.

Proses yang dilakukan adalah, pertama, diasumsikan terdapat kunci simetris C dengan

panjang kunci l bit. Kemudian pilih bilangan prima yang besar yaitu p dengan p - 1 = qm dengan q adalah 3l bit bilangan prima.

Untuk melakukan enkripsi terhadap pesan, algoritma enkripsi perlu dimodifikasi dengan menghitung e = $C_k(m)$ dengan K merupakan hasil fungsi hash h^r menjadi string sepanjang l bit.

Fungsi hash yang dapat digunakan adalah SHAdan implementasi secara hybrid ini terbukti aman.

3.5.3 Varian *hash-free*

Pada model implementasi ini, fungsi hash tidak diterapkan sehingg untuk keamanan didasarkan pada Diddie-Helman *decision problem*.

Perubahan skema yang terjadi adalah sebagai berikut :
Elemen d pada kunci publik disubsitusi dengan d_1, d_2, ….d_k Nilai d_i diperoleh melalui $d_i = g_1^{y_{i1}} g_2^{y_{i2}}$ dan y_{i1} dan y_{i2} merupakan nilai random yang disertakan pada kunci privat.

Untuk mengenkripsi pesan, hitung nilai v dengan rumus :

$$v = c^r \prod_{i=1}^{k} d_i^{a_i r},$$

Sedangkan saat mendekripsi pesan, hitung nilai v dengan rumus :

$$v = u_1^{x_1 + \sum_{i=1}^{k} a_i y_{i1}} u_2^{x_2 + \sum_{i=1}^{k} a_i y_{i2}}$$

3.5.4 Lunch-time attacks

Variasi ini digunakan apabila kebutuhan keamanan dibatasi dengan ketentuan bahwa skema hanya aman terhadap serangan lunch-time. Dengan demikian skema dapat diserhanakan dengan mengeliminasi parameter d, y_1, y_2, dan H pada kunci.

Kemudian, saat enkripsi, proses perhitungan menjadi sederhana dengan hanya menghitung v dengan rumus v = c^r.

Sedangkan saat dekripsi, nilai v diverifikasi dengan rumus :

$$v = u_1^{x_1} u_2^{x_2}$$

3.6 Performansi

Karakteristik Cramer-Shoup selain tahan terhadap serangan *adaptive chosen ciphertext attack* juga memerlukan waktu komputasi yang lebih lama karena menggunakan bilangan prima yang besar. Selain itu hasil encode pesan memiliki bit atau ukuran yang lebih panjang.

4. Skema tanda tangan dijital dengan Cramer Shoup

Sebuah tanda tangan dijital pada dokumen elektronis dapat diterapkan menggunakan algoritma kunci publik asimetris. Dengan demikian prinsip algoritma Cramer Shoup pun dapat diterapkan untuk menghasilkan tanda tangan dijital. Skema tanda tangan dijital telah banyak diajukan oleh para ahli, salah satunya skema yang didasarkan pada strong RSA assumption. Skema tersebut diajukan oleh Ronald Cramer dan Victor Shoup pada tahun 2000. Skema kedua pada paper ini adalah Naïve Cramer Shoup yang diajukan pada paper ini yang lebih sederhana untuk diimplementasikan.

4.1 Skema berdasar *strong* RSA *assumption*

Ronald Cramer dan Victor Shoup mendesain skema ini didasarkan pada prinsip bahwa sebuah skema tanda tangan dijital dapat dikatakan aman apabila :

- menggunakan fungsi hash yang collison resistant dan eksis untuk parameter l atau
  h : {0,1}* → {0,1}^{l}
- *strong RSA conjecture*
- Bilangan prima yang dibangkitkan adalah Sophie German Prime yaitu bahwa jika p adalah prima, maka 2p + 1 juga prima. [5]

Strong RSA assumption diperkenalkan oleh N. Baric dan B. Pfitzmann pada tahun 1997 dan digunakan untuk menganalisis sejumlah skema kriptografi. *Strong RSA assumption* merupakan asumsi yang diterapkan bahwa ada suatu permasalahan yang sulit untuk dicari solusinya.

Permasalahan tersebut adalah jika diberikan RSA modulus n dan sebuah bilangan acak z dengan z $\in Z_n^*$ maka cari r > 1 dan y $\in Z_n^*$ dimana $y^r = z$. Berbeda dengan *RSA assumption*, nilai r independen terhadap z sedangkan pada *RSA assumption* nilai r masih mungkin diturunkan dari nilai z.

Skema dibuat dengan dua *security parameter* yaitu l dan l' dengan l – 1 < l'. Salah satu pasangan nilai yang mungkin dipakai adalah l=160 dan l' = 512. Fungsi hash dipakai dengan output yang dapat direpresentasikan sebagai integer positif yang lebih kecil dari 2^l. Untuk nilai n yang positif integer, QR_n adalah *subgroup* Z_n^* dari nilai kuadrat.

4.1.1 Pembangkitan kunci

Dua buah kunci publik dan privat diperoleh melalui proses sebagai berikut :

1. Dua buah bilangan prima sebesar l'-bit dipilih, yaitu p dan q. Nilai p dan q haruslah bilangan Sophie German Prime dengan $p = 2p'+1$ dan $q = 2q'+1$, p' dan q' adalah prima.
2. Hitung n = p.q
3. Pilih bilangan acak g dan x $\in QR_n$
4. Ambil bilangan acak l+1-bit prima yaitu $\tilde{e}$

Bilangan (p,q) digunakan sebagai kunci privat. Sedangkan (n, g, x, $\tilde{e}$) digunakan sebagai kunci publik.

4.1.2 Pemberian tanda tangan

Untuk pesan *m*, $S(m) = (e,y,\tilde{y})$ diperoleh melalui cara sebagai berikut :

e : bilangan acak prima l+1

$\tilde{y}$: bilangan acak $\in QR_n$

$\tilde{x} = \tilde{y}^{\tilde{e}} . g^{-h(m)} \bmod n$

$y = (x . g^{h(\tilde{x})})^{e^{-1} \bmod \Phi(n)} \bmod n$

4.1.3 Verifikasi

Untuk melakukan verifikasi yang perlu dilakukan adalah :

1. Cek apakah e merupakan bilangan ganjil (l + 1) bit selisihnya dengan $\tilde{e}$
2. Hitung $\tilde{x} = \tilde{y}^{\tilde{e}} . g^{-h(m)} \bmod n$
3. Cek apakah $x = {}_n y^e . g^{-h(\tilde{x})}$
   Jika benar maka pesan yang diterima valid.

4.1.4 Performansi

Skema yang diajukan Ronald Cramer dan Victor Shoup ini dklaim sebagai skema efektif yang terbukti aman. Uji coba dilakukan dengan mengimplementasikan skema dalam bahasa C. [2]. Pada implementasi ini, fungsi hash yang digunakan adalah SHA-1.

Proses penanda tanganan diimplementasikan dalam dua fase yaitu inisialisasi penggunaan kunci (*key set up*) serta proses penandatanganan utama. Fase *key set up* dijalankan sekali untuk menghasilkan tabel berdasarkan masukan kunci yang akan mempercepat proses penandatanganan maupun verifikasi.

Hasil pengujian menyatakan bahwa waktu yang dibutuhkan untuk menadatangani adalah 50 ms, untuk melakukan verifikasi adalah 57 ms, dan waktu untuk fase *key set up* sekitar 31 ms.

Pada fase penandatanganan, waktu banyak dihahiskan untuk memperoleh bilangan prima dan melakukan operasi eksponensial.

Jika dibandingkan dengan RSA, waktu yang dibutuhkan skema ini menghabiskan waktu lebih lama yaitu 1,4 kali dari waktu yang diperlukan skema RSA. Jika ditambah waktu untuk *key set up*, masa waktu untuk skema ini mengahabiskan waktu 2,3 kali lebih lama dari RSA.

Sedangkan untuk membandingkan waktu yang dihabiskan saat verifikasi antar skema ini dengan RSA sulit dilakukan. Hal ini dikarenakan perpangkatan yang diguinakan bervariasi. Akan tetapi dapat diprediksi bahwa jika RSA menggunakan pangkat yang kecil, skema ini akan menghabiskan waktu yang lebih lama dibanding RSA dalam verifikasi. Akan tetapi apabila pangkat yang digunakan besar pada RSA, maka waktu verifikasi skema akan lebih cepat.

Masalah besarnya waktu yang dihabiskan memang bukan satu-satunya parameter ukuran performansi. Paramter lain dapat berupa ukuran

tanda tangan dijital yang dihasilkan seta kunci publik. Dalam hal ini, harus diakui bahwa ukuran tanda tangan dijital serta kunci publik skema ini lebih besar dibanding dengan RSA.

4.2 Skema berdasar *Naïve Cramer Shoup*

Skema yang dinamakan Naïve Cramer Shoup ini menggunakan algoritma Cramer Shoup sepenuhnya untuk menghasilkan tanda tangan dan melakukan verifikasi. Skema ini juga menggunakan fungsi hash untuk menghasilkan message digest yang kemudian dienkripsi menggunakan algoritma enkripsi Cramer Shoup. Verifikasi, menggunakan algoritma dekripsi Cramer Shoup. Kunci yang digunakan merupakan kunci pemilik dokumen yang dihasilkan oleh algoritma pembangkit kunci Cramer Shoup.

4.2.1 Pembangkitan kunci

Pemilik dokumen yang akan ditandatangani mempublikasikan (x_1,x_2,y_1,y_2,z) sebagai kunci publik dan menyimpan (g_1,g_2,c,d,h,H) sebagai kunci privat. Kunci diperoleh melalui algoritma pembangkit kunci.

4.2.2 Pemberian tanda tangan

Isi dokumen dikenakan fungsi hash untuk menghasilkan message digest. Dalam hal ini, fungsi hash yang digunakan adalah SHA-1. Proses *hashing* yang dilakukan adalah sebagai berikut :
Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 264 bit (2.147.483.648 *gigabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh MD5. Yang dilakukan pertama kali adalah penambahan bit-bit pengganjal (*padding bits*) pada pesan. Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. Angka 512 ini muncul karena SHA memperoses pesan dalam blok-blok yang berukuran 512. Pesan dengan panjang 448 bit pun tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

Kemudian dilakukan penambahan nilai panjang pesan semula. Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi 512 bit.

Ketiga adalah inisialisasi penyangga (*buffer*) MD. Kelima penyangga yang diperlukan digunakan untuk menampung hasil antara dan hasil akhir.
Kelima penyangga ini diberi nama *A*, *B*, *C*, *D*, dan *E*. Setiap penyangga diinisialisasi dengan nilai-nilai dalam notasi HEX.

Dan terakhir adalah pengolahan pesan dalam blok berukuran 512 bit. Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit . Setiap blok 512-bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit. Proses ini terdiri dari 80 buah putaran dan masing-masing putaran menggunakan bilangan penambah *Kt*. Setiap putaran menggunakan operasi dasar yang sama yaitu :
$a, b, c, d, e \leftarrow (\text{CLS5}(a) + ft(b, c, d) + e + Wt + Kt), a, \text{CLS30}(b), c, d$

Setelah putaran ke-79, *a*, *b*, *c*, *d*, dan *e* ditambahkan ke *A*, *B*, *C*, *D*, dan *E*. Selanjutnya algoritma memproses untuk blok data berikutnya. Keluaran akhir dari algoritma SHA adalah hasil penyambungan bit-bit di *A*, *B*, *C*, *D*, dan *E*.

Hasil penyambungan bit-bit itulah yang disebut sebagai message digest (MD). Hasil yang diperoleh aman karena secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan.

Setelah diperoleh MD, MD dikenakan algoritma enkripsi Cramer Shoup dengan menggunakan kunci privat pengirim.

1. MD dikonversi menjadi elemen G.
2. Bilangan acak r dipilih dari {0,….,q-1} kemudian menghitung nilai u_1, u_2, e, α, dan v dengan menggunakan kunci

privat pemilik pesan (g_1,g_2,c,d,h,H) sebagai berikut :

$u_1 = g_1^r$
$u_2 = g_2^r$
$e = h^r MD$
$\alpha = H(u_1, u_2, e)$
dengan H sebagai fungsi hash yang *collision-resistant*
$v = c^r d^{r\alpha}$

3. S(m) = (u_1,u_2,e,v).

Tanda tangan dijital berupa S(m) yang kemudian di-*append* dengan pesan pada awal atau akhir dokumen.

4.2.3 Verifikasi

1. Hitung nilai α
   $\alpha = H(u_1, u_2, e)$
2. Jika hasil $u_1^{x1+y1\alpha}\ u_2^{x2+y2\alpha} = v$ maka $MD = e/u_1^z$
3. Ambil pesan m pada dokumen
4. Hitung MD' yaitu MD' = H(m)
5. Jika MD = MD' maka otentikasi berhasil

5. Implementasi dan Pengujian

Implementasi dilakukan untuk menguji skema Naïve Cramer Shoup**.** Implementasi dilakukan dalam bahasa java versi 1.5 dan diuji di lingkungan pengembangan berbasis windows.

Program merupakan pengembangan dari Toy Implementation yang dibuat oleh M. Oliver M'o'ller pada tahun 2004. Toy Implementation merupakan implementasi algoritma Carmer Shoup pada pengiriman pesan melalui applet.
Kode program pada Toy implementation diubah sedemikian rupa sehingga dapat mendukung implementasi skema Naïve Cramer Shoup untuk tanda tangan dijital.

5.1 Implementasi Naïve Cramer Shoup dalam bahasa Java

Program dinamakan "Cramer Shoup on Digiture". Program ini dapat melakukan pembangkitan kunci public dan privat, menandatangani dokumen teks, serta melakukan verifikasi.

Berikut daftar nama kelas yang diimplementasikan untuk membangun program :

Base64Handler	:	kelas konversi bilangan
SecretKey	:	kelas kunci privat
PublicKey	:	kelas kunci publik
KeyGeneratorCS	:	kelas untuk membangkitkan kinci
Crypter	:	kelas form utama
CryptoModule	:	kelas yang mendefinisikan kunci
HashFunction	:	kelas untuk membangkitkan fungsi hash

KeyGenerator CS secara otomatis menghasilkan sepasang kunci public dan privat dan menyimpannya dalam file CryptoModule.java sebagai simulasi.

Cramer Shoup membutuhkan representasi builangan prima yang besar. Oleh karena itu tidak cukup menggunakan integer biasa. Untuk memenuhi kebutuhan tersebut, program ini memanfaatkan kelas Big Integer di Java. Sedangkan untuk mendapatkan bilangan secara acak, dapat memnafaatkan kelas Random pada Java.

5.1.1 Pembangkitan kunci

Kode program pembangkitan kunci sebagai berikut :

```
private        static        void
createNewKey(int bits){

createPrimePair(bits);
showFactorization();
create4Generator();
showGenerators();
Random rnd = new Random();
do{
x1=(new  BigInteger(nbits,  rnd))
   .mod(primeP);
}while(x1.equals(BigInteger.ZERO)
   );
do{
  x2=(new        BigInteger(nbits,
   rnd)).mod(primeP);
}while(x2.equals(BigInteger.ZERO)
   );
do{
y1=(new          BigInteger(nbits,
   rnd)).mod(primeP);
}while(y1.equals(BigInteger.ZERO)
   );
do{
y2=(new          BigInteger(nbits,
   rnd)).mod(primeP);
}while(y2.equals(BigInteger.
   ZERO));
    do{
z=(new           BigInteger(nbits,
   rnd)).mod(primeP);
}while(z.equals(BigInteger.ZERO))
   ;

c=((g1.modPow(x1,primeP)).multipl
y(g2.modPow(x2,primeP))).mod(prim
eP);

d=((g1.modPow(y1,primeP)).multipl
y(g2.modPow(y2,primeP))).mod(prim
eP);

h = g1.modPow(z, primeP);
}
```

Tuple (x1,x2,y1,y2, dan z) digunakan sebagai kunci yang digunakan untuk proses tanda tangan. Sedangkan (c,d,h) sebagai kunci untuk verifikasi. Parameter g1 dan g2 merupakan bilangan acak generator Z_p^* dengan p bilangan acak prima, output dari prosedur create4generator().

Contoh pemanggilan program pembangkitan kunci, kelas KeyGeneratorCS pada command prompt untuk basis teks :

```
java      KeyGeneratorCS      –f
kunci.java –o "ratna" –n KEYNAME
128
```

Maksud perintah tersebut adalah : jalankan KeyGeneratorCS dan simpan hasilnya pada file "kunci.java". Pemilik kunci adalah "ratna" dan nama kunci adalah "KEYNAME". Angka 128 menandakan jumlah random bit. Semakin besar jumlah random bit semakin panjang kunci yang dihasilkan.

Hasil tuple kunci yang dihasilkan pada perintah tersebut adalah :

```
p=
162475081987203281860167164511599309514076 9

x1 131895449472452046559653659635588781538719 5
x2: 208654775308969596566165394600690102036896
y1 926829876764713663964059430644871716124123
y2: 734865279631863956073583741316161068848199
z 139446504462075159056308346949957922043182 2

g1: 767056468418168454784114867689817090944396
g2: 506609454703099496253962412461452876354320
c  203196066236499680024296864285150929839710
d 922166998282248465540196023941868132396259
h: 518284389734133266940691060334749005924472
```

Bandingkan jika random bit yang digunakan adalah 16, tuple kunci yang dihasilkan adalah :

```
p = 509557

x1 247852
x2: 283889
y1 433195
y2: 237498
z 5747

g1: 93387
g2: 251993
c  198262
d 107203
h: 38650
```

5.1.2 Tanda tangan

Cuplikan kode program untuk mendapatkan S(m) = {u1,u2,e,v} adalah sebagai berikut :

```
r =
bigRandom(sk.k+1).mod(sk.p);
u1 = sk.g1.modPow(r,sk.p);
u2 = sk.g2.modPow(r,sk.p);
e  =
((sk.h.modPow(r,pk.p)).multipl
y(md)).mod(sk.p);
alpha =
hashBitList(sk.k,sk.p,sk.hash,
bitListThree(sk.k,u1,u2,e));
v =
((sk.c.modPow(r,sk.p)).multipl
y(sk.d.modPow(r.multiply(alpha
),sk.p))).mod(sk.p);
```

Sebelum S(m) dihitung, isi dokumen dibaca terlebih dahulu. Kemudian dikenakan fungsi sha(m) yang menghasilkan atribut `md` bertipe BigInteger.

Pada cuplikan kode program di atas, bilangan r dipilih secara random dengan fungsi bigRandom() kemudian baru dihitung nilai u1,u2 yang menggunakan masukan g1 dan g2 pada kunci. Kemudian proses berlanjut dengan menghitung e, alpha, dan v yang menggunakan parameter c, d, dan h pada kunci.

Untuk menandatangani dokumen, perintah yang digunakan adalah :

```
java Crypter s mpeta.c d128.c
```

s menandakan perintah untuk melakukan signature atau tanda tangan dijital. mpeta.c merupakan dokumen yang akan ditanda tangani sedangkan d128.c adalah nama hasil dokumen yang setelah ditandatangani

Hasil pemanggilan program untuk menandatangani dokumen adalah sebagai berikut

Kunci yang digunakan :
g1:
767056468418168454784114867689817090944396
g2:
506609454703099496253962412461452876354320
c
203196066236499680024296864285150929839710
d
922166998282248465540196023941868132396259
h:
518284389734133266940691060334749005924472

Dokumen yang ditandatangani : mpeta.c
Ukuran file sebelum ditanda tangan : 15,183 byte
Message digest yang dihasilkan :

```
381661023037153717936361249262
4192175262211635696
```

Tuple (u1,u2,e,v) yang diperoleh adalah sebagai berikut :

```
r  : 92613307266440553358329415298283352714383 7
u1 : 30715586471446622041095839394870464131701 7
u2:12111083791272292151081389017194060461680 39
e : 37125185876604256376241644009096416636613 3
v : 14891593610590109893345836531576823713796 92

r : 14389961435685349453666143693585595490576700
u1 : 79441483372504783163545745125312900514588 8
u2 :13032046327290147993309223693776636529881 8
e :34256958661076848119978052714309882754104 1
v :77370077574512319346717097624921252254603 3

r :14741122136306643608979725095570891246014 47
u1 :36663218468956750545836264049494918047706 7
u2 :68407253946114796482762991341741017020310 0
e :12953044447133493413314719232064117807158 95
v :63499410278255051227159986198510303337383 5
```

Sedangkan tanda tangan dijital yang di-*append* pada dokumen berupa integer64 hasil konversi dari tuple (u1,u2,e,v) :

```
mR9qGQv/5TdpxNjRlX3GZWHHPhXa1KkXdrUMe
me4dlWoJz2rct9g6FNgUQjLy+5TdTbQMIhvBG
Lh
arBBvw25H2teon74MRBPlhn2wVZHsrXq/OquX
SyXiSCAEif5DuGsuK/90JIhDi+/QjFBeyIeNQ
Ul
```

```
9EDFq9L5tHXfEdhR+Ojfep5Xz4XZ/FmUe
bDi0XG/LzmErmQP8+iXqXJm9qwh3qOx6s
aq9kZdbCUK
HJr7kt87q6Wx1G+1IwvUH9vqe2DcXt2iU
9SLfBHz6h0PogYaW9xQpc
```

Ukuran dokumen yang telah ditandatangani : 15,635 bytes

5.1.3 Verifikasi

Cuplikan kode program untuk menghitung $MD = e/u_1^z$ adalah :

```
if(
(v.equals(((u1.modPow(pk.x1.ad
d(alpha.multiply(pk.y1)),pk.p)
).multiply(u2.modPow(pk.x2.add
(alpha.multiply(pk.y2)),pk.p))
).mod(pk.p))))
{

if(u1.equals(BigInteger.ZERO))
   md = BigInteger.ZERO;
else
   md =
(e.multiply((u1.modPow(pk.z,pk
.p)).modInverse(pk.p))).mod(pk
.p);
cChunk = bigInteger2bits(pk.k,
md);
              for(i = 0; i <
pk.k - 1; i++) // ignore most
significant bit

res[resIndex++] = cChunk[i];

}
else
  for(i = 0; i < pk.k-1;  i++)
    res[resIndex++] = true;
```

Tanda tangan dijital yang diekstraksi dari dokumen, diolah berdasarkan parameter k pada kunci sehingga mendapatkan tuple (u1, u2, e, v)

Kemudian dilakukan proses verifikasi terlebih dulu seperti pada algoritma dekripsi Cramer Shoup sampai mendapatkan nilai md.

Keluaran akhir dari program tersebut adalah md yang masih bertipe array of bit. Method untuk mendapatkan md tersebut dipanggil dalam method lain yaitu untuk melakukan verifikasi, dan mengubahnya menjadi tipe BigInteger.

Kemudian isi pesan dalam dokumen yaitu isi dokumen selain tanda tangan dijital diambil untuk kemudian dikenakan fungsi sha(m) yang akan menghasilkan md' bertipe BigInteger.

Kemudian md' dibandingkan dengan md. Jika bernilai sama, maka pesan lulus otentikasi. Pengecekan juga dilakukan terhadap tanda tangan dijital, apakah tanda tangan dijital pada dokumen merupkan format yang valid atau bukan.

Perintah yang digunakan untuk melakukan verifikasi adalah :

```
java Crypter v d128.c
```

v adalah option yang digunakan untuk melakukan verifikasi. Sedangkan d128.c adalah file dokumen yang telah ditandatangani dan akan dicek itentikasinya.

Kunci yang digunakan untuk melakukan verifikasi adalah :

x1 1318954494724520465596536596355887815387195
x2: 208654775308969596566165394600690102036896
y1 926829876764713663964059430644871716124123
y2: 734865279631863956073583741316161068848199
z 139446504462075159056308346949957922043182 2

MD yang diperoleh setelah dilakukan proses verifikasi terhadap tanda tangan dijital :

```
381661023037153717936361249262
4192175262211635696
```

Sedangkan MD' hasil perhitungan fungsi hash terhadap ekstraksi isi dokumen adalah :

```
381661023037153717936361249262
4192175262211635696
```

Sehingga MD = MD', dokumen d128.c lulus otentikasi.

5.2 Pengujian

Pengujian dilakukan dengan memodifikasi tanda tangan dijital pada dokumen yang telah ditanda tangani Kemudian hasil modifikasi diverifikasi untuk mengetahui apakah dokumen masih lulus otentikasi atau tidak. Hasil yang diharapkan adalah modifikasi yang dilakukan sesedikit pun pada tanda tangan dijital akan menghasilkan kegagalan pada proses otentikasi.

Pengujian dibatasi untuk mengetahui sensitifitas algoritma Cramer Shoup dan tidak untuk menguji SHA-1 sehingga pengujian tidak dilakukan dengan modifikasi isi dokumen.

5.2.1 Penambahan satu karakter pada tanda tangan dijital

Perubahan yang dilakukan yaitu menambah karakter "i" pada tanda tangan dijital :

```
mR9qGQv/5TdpxNjRlX3GZWHHPhXa1K
kXdrUMeme4dlWoJz2rct9g6FNgUQjL
y+5TdTbQMIhvBGLh
arBBvw25H2teon74MRBPlhn2wVZHsr
Xq/OquXSyXiSCAEif5DuGsuK/90JIh
Di+/QjFBeyIeNQUl
9EDFq9L5tHXfEdhR+Ojfep5Xz4XZ/F
mUebDi0XG/LzmErmQP8+iiXqXJm9qw
h3qOx6saq9kZdbCUK
HJr7kt87q6Wx1G+1IwvUH9vqe2DcXt
2iU9SLfBHz6h0PogYaW9xQpc
```

MD yang diperoleh setelah dilakukan proses verifikasi terhadap tanda tangan dijital :

```
381661023037153717936361249262
41921ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿð
```

Kesimpulan :
Pesan tidak lulus otentikasi karena MD ≠ MD′
Dengan demikian penambahan satu karakter pada tanda tangan dijital menyebabkan pesan tidak lulus otentikasi
Proses verifikasi Naïve Cramer Shoup sensitif terhadap penambahan satu karakter pada tanda tangan dijital.

5.2.2 Pengurangan satu karakter pada tanda tangan dijital

Perubahan yang dilakukan yaitu menghapus karakter "s" pada tanda tangan dijital :

```
mR9qGQv/5TdpxNjRlX3GZWHHPhXa1K
kXdrUMeme4dlWoJz2rct9g6FNgUQjL
y+5TdTbQMIhvBGLh
arBBvw25H2teon74MRBPlhn2wVZHsr
Xq/OquXSyXiSCAEif5DuGSuK/90JIh
Di+/QjFBeyIeNQUl
9EDFq9L5tHXfEdhR+Ojfep5Xz4XZ/F
mUebDi0XG/LzmErmQP8+iXqXJm9qwh
3qOx6saq9kZdbCUK
HJr7kt87q6Wx1G+1IwvUH9vqe2DcXt
2iU9SLfBHz6h0PogYaW9xQpc
```

MD yang diperoleh setelah dilakukan proses verifikasi terhadap tanda tangan dijital :

```
381661023037153717?ÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿ
```

Kesimpulan :
Pesan tidak lulus otentikasi karena MD ≠ MD′
Dengan demikian pengurangan satu karakter pada tanda tangan dijital menyebabkan pesan tidak lulus otentikasi
Proses verifikasi Naïve Cramer Shoup sensitif terhadap pengurangan satu karakter.

5.2.3 Pengubahan satu karakter pada tanda tangan dijital

Perubahan yang dilakukan yaitu mengubah karakter V menjasi karakter w pada tanda tangan dijital :

```
mR9qGQv/5TdpxNjRlX3GZWHHPhXa1K
kXdrUMeme4dlWoJz2rct9g6FNgUQjL
y+5TdTbQMIhvBGLh
arBBvw25H2teon74MRBPlhn2wWZHsr
Xq/OquXSyXiSCAEif5DuGsuK/90JIh
Di+/QjFBeyIeNQUl
9EDFq9L5tHXfEdhR+Ojfep5Xz4XZ/F
mUebDi0XG/LzmErmQP8+iXqXJm9qwh
3qOx6saq9kZdbCUK
HJr7kt87q6Wx1G+1IwvUH9vqe2DcXt
2iU9SLfBHz6h0PogYaW9xQpc
```

MD yang diperoleh setelah dilakukan proses verifikasi terhadap tanda tangan dijital :

```
38166102303715371?ÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿ75262211635696
```

Kesimpulan **:**
Pesan tidak lulus otentikasi karena MD ≠ MD′
Dengan demikian penambahan satu karakter pada isi dokumen menyebabkan pesan tidak lulus otentikasi
Proses verifikasi Naïve Cramer Shoup sensitif terhadap pengubahan satu karakter.

5.2.4 Menggunakan kunci publik yang berbeda

Pengujian juga dilakukan dengan menggunakan kasus seandainya penerima menggunakan kunci yang berbeda pada saat melakukan verifikasi.

Misal kunci yang digunakan adalah :

x1 247852
x2: 283889
y1 433195
y2: 237498
z 5747

Maka MD yang dihasilkan adalah :

```
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÞ
```

Hasil tersebut menunjukan adanya *error*dan mengakibatkan dokumen tidak lulus otentikasi.

5.2.5 Kesimpulan pengujian

Berdasarkan pengujian yang telah dilakukan dengan mengujicoba beberapa kasus modifikasi pada dokumen, hasil yang diperoleh sesuai dengan yang diharapkann. Yaitu tanda tangan dijital Naïve Cramer Shoup sensitive terhadap modifikasi. Dengan demikian dapat disimpulkan bahwa skema Naïve Cramer Shoup menghasilkan tanda tangan dijital yang aman terhadap serangan modifikasi.

Jika dianalisis, hal ini disebabkan algoritma dekripsi pada Cramer Shoup yang digunakan pada saat verfikasi akan menghasilkan plaintext yang selalu berbeda sesuai dengan parameter masukan chipertext.

Dapat dilihat pula pada hasil penambahan maupun pengurangan karakter bahwa *error* yang terjadi pada satu bagian akan menyebabkan *error* pada bagian selanjutnya. Pengujian juga dilakukan untuk memperkuat pernyataan tersebut dengan menghapus tanda tangan dijital sebanyak lebih dari satu karakter. Hasil yang diperoleh ternyata bilangan MD yang dihasilkan menjadi tidak terdefinisi.

5.3 Perbandingan performansi dengan RSA

Untuk mengetahui perbandingan performansi skema Naïve Cramer Shoup dengan skema naïve RSA, dilakukan dengan cara membandingkan performansi program “Cramer Shoup on Digiture” yang dibandingkan dengan program Digiture. Program Digiture diimplemetasikan oleh R. Fitri, Dini A., dan Ratna M. pada tahun 2006. Program Digiture ini menggunakan algoritma SHA-1 dan RSA dalam menghasilkan tandatangan dan melakukan otentikasi.

Berbeda dengan “Cramer Shoup on Digiture”, Digiture dikembangkan dalam bahasa C# dengan kakas visual studio .net 2005.

Pengujian berbagai kasus modifikasi telah dilakukan terhadap Digiture dengan hasil yang sama, yaitu bahwa modifikasi yang dilakukan satu karakter sudah dapat mengakibatkan dokumen tidak lulus otentikasi. Dengan demikian perbandingan performansi antara skema Naïve Cramer Shoup dan Naïve RSA dilakukan dengan membandingkan ukuran tanda tangan dijital yang dihasilkan, kunci, serta waktu yang diperlukan.

Hasil pengujian sebagai berikut :
Dengan menggunakan file masukan yang sama, yaitu mpeta.c, tanda tangan dijital yang dihasilkan program Digiture adalah sebagai berikut :

```
<ds>287A7C03410E312191DA165FDC19ADC0
414D7D6B22079B8A9D98</ds>
```

Sedangkan kunci yang dihasilkan Digiture untuk 128 random bit adalah :
Kunci privat = 1297
Kunci publik = 109

Kesimpulan sementara yang diperoleh melalui kedua hasil pengujiantersebut adalah :
1. ukuran tanda tangan dijital yang dihasilkan RSA lebih kecil dibandingkan dengan Naïve Cramer Shoup
2. Panjang kunci RSA lebih kecil dibanding kunci Naïve Cramer Shoup

Mengenai perbandingan waktu, dilihat dari kompleksitas algoritma pada Naïve Cramer Shoup untuk mendapatkan bilangan prima yang acak dan pada saat tanda tangan maupun verfikasi, algoritma Naïve Cramer Shoup membutuhkan waktu yang lebih lama dari RSA. Akan tetapi kesimpulan tersebut masih perlu diuji untuk berbagai macam kasus lagi.

6. Kesimpulan

1. Tanda-tangan digital dapat digunakan untuk menjaga integritas suatu data, dan juga mampu digunakan untuk membuktikan asal pesan dan nirpenyangkalan.
2. Algoritma Cramer Shoup dapat dimanfaatkan untuk tanda tangan dijital menjadi skema Naïve Cramer Shoup
3. Tanda tangan dijital yang dihasilkan Cramer Shoup aman terhadap serangan berbagai kasus modifikasi tanda tangan dijital
4. Dibanding RSA, Cramer Shoup memiliki kompleksitas dan ukuran tanda tangan dijital yang lebih tapi menghasilkan kunci yang lebih aman. Dengan kelebihan Cramer Shoup yang tahan terhadap serangan *adaptive chosen message attack*, Cramer Shoup dapat menjadi alternatif skema tanda tangan dijital yang lebih aman.

7. Referensi

[1] Rinaldi Munir. Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung. 2006.

[2] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. Mei 1998.

[3] Ronald Cramer and Victor Shoup. Signature Schema Bases on Strong RSA Assumption. 9 Mei 2000.

[4] Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. 12 Desember 2001

[5] Zeph Grunschlag, Prof.. Digital Signature.2001