Pengaplikasian Protokol Enkripsi dan Tanda Tangan Digital dengan Memanfaatkan Speech Recognition

Bernardino Madaharsa Dito Adiwidya / 13507089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if17089@students.if.itb.ac.id

***Abstrak*—Saat ini terdapat banyak metode atau langkah-langkah untuk menjaga aspek keamanan dari suatu pengiriman pesan yang disebut dengan protokol kriptografi. Dalam makalah ini, diperkenalkan suatu protokol kriptografi enkripsi dengan tanda-tangan digital untuk mengamankan pesan dengan menjamin kerahasiaan pesan sekaligus otentikasi dan keaslian pesan. Dalam prosesnya, digunakan media surat elektronik (*e-mail*) yang diintegrasikan dengan penggunaan media audio. Pesan yang dikirimkan dienkripsi dengan kriptografi kunci-publik lalu diberikan hasil fungsi *hash*-nya berupa suatu *message digest*. *Message digest* ini kemudian digabungkan dengan suatu sandi-lewat lalu dilakukan pemberian fungsi *hash* yang sama menjadi tanda-tangan digital. Nantinya yang akan dikirimkan kepada penerima melalui *e-mail* adalah pesan terenkripsi beserta tanda-tangan digital, sedangkan kata kunci pengirim akan disampaikan lewat media lain berupa audio (misalnya telepon). Sebelum akan mendekripsi, penerima yang dianggap telah mengetahui kunci privat pengirim akan membuat *message digest* dari pesan terenkripsi tersebut dengan fungsi *hash* yang sama. Dengan memanfaatkan *speech recognition*, aplikasi akan menggabungkan *message digest* tersebut dengan kata kunci yang dikatakan oleh pengirim lewat media audio, lalu dengan fungsi *hash* yang sama akan menghasilkan suatu nilai *signature* yang seharusnya sama dengan yang diberikan oleh pengirim. Setelah terautentikasi, penerima dapat mendekripsi pesan yang diterimanya.**

***Kata kunci*—enkripsi, protokol kriptografi, tanda-tangan digital, *speech recognition*.**

I. PENDAHULUAN

Pada saat ini kriptografi sudah digunakan secara luas, terutama untuk menjamin keamanan suatu pesan. Mulanya, sesuai dengan asal katanya, yaitu κρυπτο dan γραφη yang artinya tulisan rahasia, kriptografi merupakan ilmu atau seni untuk hanya menjaga kerahasiaan pesan saja. Namun saat ini, kriptografi digunakan untuk menjaga keamanan pesan, tidak hanya kerahasiaan pesan saja. Aspek keamanan yang disediakan oleh kriptografi adalah kerahasiaan pesan (*confidentiality*/*secrecy*), otentikasi (*authentication*), keaslian pesan (*data integrity*), dan anti-penyangkalan (*nonrepudiation*).

Ada banyak algoritma kriptografi yang telah diciptakan. Algoritma-algoritma yang telah ada terus-menerus dikembangkan akibat adanya kemajuan teknologi, terutama kemajuan *hardware*. Perkembangan tersebut juga meliputi pemanfaatan lebih dari satu macam algoritma untuk suatu pekerjaan. Penggunaan beberapa algoritma kriptografi tersebut diatur dalam rangkaian langkah-langkah yang melibatkan dua atau lebih orang yang disebut dengan protokol kriptografi.

Protokol-protokol kriptografi telah banyak dikembangkan saat ini. Akan tetapi, tidak ada satu pun yang benar-benar efisien dan sangat cocok untuk segala macam kebutuhan keamanan pesan. Protokol yang digunakan selalu tergantung kebutuhan dan keadaan pengirim dan penerima pesan.

Dalam makalah ini dikembangkan suatu protokol yang memanfaatkan enkripsi dan tanda-tangan digital. Tidak seperti protokol pada umumnya, pada makalah ini protokol memanfaatkan media lain yang cepat untuk menyampaikan suatu sandi-lewat, yaitu lewat media audio, yang dalam hal ini adalah telepon. Pesan asli yang terenkripsi dan tanda-tangan digitalnya dikirimkan lewat *e-mail*. Dengan memanfaatkan teknologi pengembangan aplikasi pada aplikasi pembaca *e-mail*, saat ini dimungkinkan melakukan validasi dengan memanfaatkan *speech recognition* untuk menerima sandi-lewat. Untuk pembuktian teknologinya, dalam pembuatan makalah ini dilakukan pembuatan contoh aplikasi *add-in* untuk menyimulasikan langkah-langkahnya. Tentu saja ada kelebihan dan kekurangan pada metode ini, tetapi setidaknya dapat memberikan suatu alternatif atau ide pengembangan protokol kriptografi masa depan.

II. ALGORITMA-ALGORITMA YANG DIGUNAKAN

Seperti yang telah disebutkan sebelumnya, protokol kriptografi melibatkan beberapa algoritma kriptografi. Protokol yang digunakan dibuat menggunakan algoritma kriptografi kunci-publik dengan enkripsi untuk *secrecy* dan fungsi *hash* satu arah untuk menandatangani pesan.

A. Algoritma Kunci-Publik dengan RSA

Algoritma kunci-publik merupakan suatu algoritma kriptografi yang menggunakan dua macam kunci:

a. kunci publik
   Kunci ini digunakan untuk publik sehingga tidak

rahasia dan dapat dipakai oleh siapapun. Dalam pengaplikasian dalam makalah ini, kunci publik digunakan untuk mengenkripsi pesan asli dari pengirim.

b. kunci privat
Kunci ini bersifat rahasia dan tidak boleh digunakan oleh pihak yang tidak berhak. Berbeda dengan kunci publik di atas, kunci ini digunakan untuk mendekripsi *ciphertext*.

Untuk melakukan enkripsi pesan, seperti yang telah disebutkan di atas, kunci publik digunakan untuk mengenkripsi pesan sedangkan kunci privat digunakan untuk mendekripsi pesan. Akan tetapi, jika algoritma ini digunakan untuk memeriksa keaslian pengirim, kunci publik digunakan untuk mendekripsi sedangkan kunci privat untuk mengenkripsi. Hal ini diperlukan karena setiap orang yang mengetahui kunci publik dapat mengetahui siapa pengirim pesan sebenarnya (untuk otentikasi dan anti-penyangkalan).

Saat ini, terdapat berbagai macam algoritma kriptografi kunci-publik yang dikembangkan. Algoritma-algoritma tersebut antara lain *RSA*, *Knapsack*, *Rabin*, dan *ElGamal*. Dalam aplikasi yang dibuat, digunakan algoritma *RSA*.

Algoritma *RSA* dipilih karena sangat populer saat ini dan memiliki kesulitan yang tinggi untuk memecahkannya karena harus menemukan faktor prima dari bilangan yang besar. Saat ini belum ada algoritma yang efisien untuk melakukannya untuk bilangan yang besar. Proses yang dilakukan pada algoritma ini adalah sebagai berikut.

a. Pembangkitan Kunci
Untuk melakukan pembangkitan kunci, dilakukan tahap-tahap sebagai berikut.
1) Pilih dua bilangan prima sembarang p dan q dengan nilai $p \neq q$.
2) Hitung $n = p \cdot q$. Karena nilai p dan q berbeda, dari nilai n tidak dapat diperoleh nilai p dan q dengan menarik akar kuadratnya.
3) Hitung $\phi(n) = (p - 1)(q - 1)$.
4) Tentukan kunci publik e yang relatif prima terhadap $\phi(n)$.
5) Bangkitkan kunci privat d dengan $e \cdot d \equiv 1 \pmod{\phi(n)}$.

b. Pengenkripsian Pesan
Seluruh pesan yang akan dikirim diubah menjadi kumpulan angka yang disepakati, misalnya setiap karakter diubah menjadi kode ASCII. Hasil kumpulan angka tersebut dikelompokkan menjadi blok-blok angka lebih kecil m_1, m_2, m_3, dst. Ambil nilai e dan n kemudian setiap blok m_i dienkripsi menjadi blok c_i dengan:
$c_i = m_i^{\,e} \bmod n$

c. Pendekripsian Pesan
Setelah mengambil nilai kunci privat d dan n, masing-masing blok *ciphertext* c_i didekripsi lagi dengan cara yang mirip dengan cara mengenkripsi sebagai berikut.
$m_i = c_i^{\,d} \bmod n$

Berdasarkan proses-proses di atas, untuk memecahkan algoritma *RSA* diperlukan pemfaktoran n menjadi p dan q. Jika berhasil, nilai $\phi(n)$ dapat diketahui. Karena e tidak dirahasiakan, maka kemudian nilai d dapat diketahui.

Sebenarnya, pemfaktoran n sendiri sudah cukup sulit, asalkan nilainya sangat besar. Penemu algoritma *RSA* menyarankan nilai p dan q memiliki panjang lebih dari 100 digit sehingga n memiliki lebih dari 200 digit. Karena belum ada tipe variabel *default* pada *compiler* pada umumnya yang dapat menampung bilangan sebesar itu, dapat dipergunakan tipe buatan baru *big integer*. Jika algoritma pemfaktoran yang digunakan adalah algoritma tercepat saat ini dan kecepatan komputasinya 1 milidetik, menurut Rivest dan kawan-kawan, usaha untuk memfaktorkan bilangan 200 digit memerlukan waktu 4 miliar tahun.

B. Fungsi Hash Satu Arah dengan SHA-1

Fungsi *hash* merupakan suatu fungsi yang menerima masukan suatu *string* dan menghasilkan suatu *string* keluaran yang panjangnya tetap. Keluarannya ini disebut dengan nilai *hash* atau *message digest*. Jika *message digest* tidak dapat diubah kembali menjadi pesan semula, fungsi ini disebut fungsi *hash* satu arah. Contoh fungsi *hash* satu arah antara lain *MD2*, *MD5*, *SHA*, *Snefru*, *RIPE-MD*, dan sebagainya.

Dalam makalah ini, digunakan algoritma *SHA* (*Secure Hash Algorithm*). Oleh NSA, algoritma ini dijadikan sebagai stadar fungsi *hash* satu arah. Algoritma ini dikembangkan menjadi beberapa versi seperti *SHA-0*, *SHA-1*, *SHA-224*, *SHA-256*, *SHA-384*, dan *SHA-512*. Algoritma *SHA* yang digunakan dalam makalah ini adalah *SHA-1*.

Fungsi SHA-1 menerima pesan dengan panjang maksimum 2^{64} bit dan menghasilkan 160-bit *message digest*. Prosesnya adalah sebagai berikut.

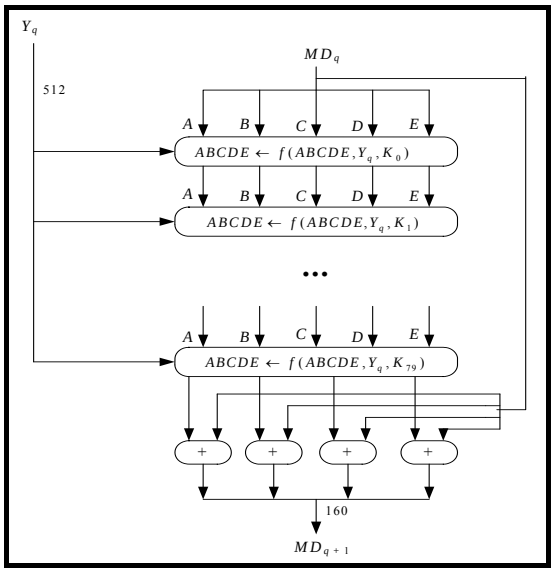
a. Melakukan inisialisasi variabel penyangga.
Variabel penyangga ini berjumlah lima buah yang panjangnya masing-masing adalah 32 bit. Kelima penyangga ini digunakan untuk menampung hasil antara dan hasil akhir. Nilai-nilainya yaitu:
A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476
E = 0xC3D2E1F0

b. Melakukan penambahan bit-bit pengganjal.
Pada pesan dilakukan penambahan bit-bit tertentu hingga panjang pesan menjadi kongruen dengan 448 modulo 512. Bit-bit yang ditambahkan terdiri atas bit 1 kemudian ditambahkan bit 0 hingga panjang pesan menjadi 64 bit kurang dari kelipatan 512.

c. Melakukan penambahan panjang pesan sebelum di-*padding*
Menurut poin (b) di atas, pesan di-*padding* dengan bit1 dan bit 0. Panjang pesan semula ini ditambahkan pada hasil *padding* pesan tersebut.

Panjang pesan semula dinyatakan dalam bilangan 64 bit sehingga panjang pesan akhir menjadi kelipatan 512.

d. Melakukan pengolahan pesan dalam blok 512 bit. Pesan yang telah ditambahkan tadi dibagi dalam blok-blok dengan panjang 512 bit. Setiap blok ini dipecah lagi menjadi 16 blok (W_1, W_2, ..., W_{16}) dengan panjang 32 bit. Masing-masing blok 512 bit diproses dengan variabel penyangga sehingga menghasilkan keluaran 128 bit. Proses ini disebut proses H_{SHA}.



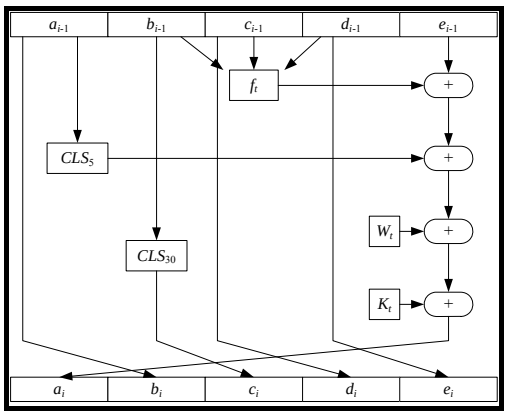
Gambar 1 Proses H_{SHA}

Proses ini menggunakan 80 putaran. Setiap putaran menggunakan bilangan penambah K_t yaitu:

- Putaran $0 \le t \le 19$: K_t = 0x5A827999
- Putaran $20 \le t \le 39$: K_t = 0x6ED9EBA1
- Putaran $40 \le t \le 59$: K_t = 0x8F1BBCDC
- Putaran $60 \le t \le 79$: K_t = 0xCA62C1D6

Setiap putaran menggunakan operasi yang sama, yaitu f_t yang rumusnya sebagai berikut.

- Putaran $0 \le t \le 19$: $(b \wedge c) \vee (\sim b \wedge d)$
- Putaran $20 \le t \le 39$: $b \oplus c \oplus d$
- Putaran $40 \le t \le 59$: $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
- Putaran $60 \le t \le 79$: $b \oplus c \oplus d$



Gambar 2 Operasi Dasar SHA-1 dalam satu putaran

Operasi pada gambar tersebut dapat ditulis dalam persamaan:

$a, b, c, d, e \leftarrow (\mathrm{CLS}_5(a) + f_t(b,c,d) + e + W_t + \mathrm{K}_t),$
$a, \mathrm{CLS}_{30}(b), c, d$

dengan:

- a, b, c, d, e adalah lima variabel penyangga
- t adalah putaran (0 s.d. 79)
- f_t adalah fungsi logika
- CLS_s adalah *circular left shift* sebanyak s bit
- W_t adalah blok kecil berukuran 32 bit
- K_t adalah konstanta penambah

Nilai W baru (dari W_1 s.d. W_{16}) diperoleh dari:

$$W_i = W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}$$

Hasil dari penyambungan akhir bit-bit A, B, C, D, dan E menjadi keluaran dari algoritma *SHA-1*.

III. PENERAPAN PROTOKOL

A. Latar Belakang

Protokol ini diterapkan dalam contoh kasus seperti ini. Bob adalah seseorang yang memiliki beberapa rekan kerja dari berbagai perusahaan yang sering mengirimkan pesan rahasia kepadanya. Perusahaan-perusahaan tersebut kadang-kadang berkonflik atau berkompetisi satu sama lain. Dalam pengiriman pesan, Bob selalu ingin menggunakan algoritma kunci-publik supaya ia cukup hanya mengetahui satu kunci privat saja. Kunci publik ia berikan kepada rekan-rekan kerjanya tersebut. Karena konflik atau kompetisi tersebut, Bob selalu hati-hati dalam menerima pesan rahasia sehingga ia selalu meminta konfirmasi si pengirim lewat telepon ketika ia menerima kiriman pesan rahasia lewat *e-mail*.

Alice adalah salah satu rekan kerja Bob dari perusahaan A. Pada suatu saat ia ingin mengirim pesan rahasia kepada Bob. Sayangnya, Eve mengetahui rencana tersebut. Eve berasal dari perusahaan E yang sedang bertikai dengan perusahaan A. Ia kebetulan merupakan rekan kerja Bob sehingga ia mengetahui kunci publik algoritma yang dipakai Bob.

B. Langkah-langkah Pengiriman Pesan

(1) Alice menyiapkan kunci publik yang pernah diberikan oleh Bob.
(2) Alice mengenkripsi *plaintext* yang ingin dikirimkan menjadi *ciphertext* dengan menggunakan kunci publik tersebut.
(3) Alice memproses *ciphertext*-nya dengan fungsi *hash* satu arah yang telah disepakati.
(4) Hasil *message digest*-nya digabungkan dengan *password* yang diinginkan Alice kemudian ditandatangani lagi.
(5) *Ciphertext* dan *signature* akhir inilah yang dikirimkan lewat *e-mail*.

C. Langkah-langkah Penerimaan Pesan

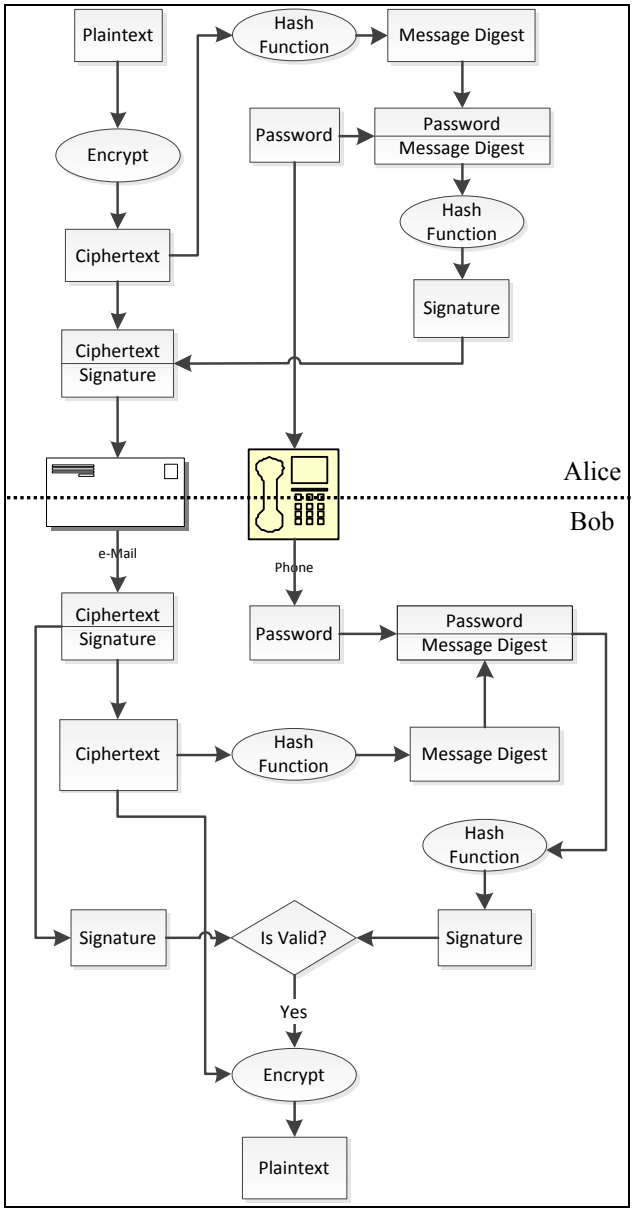
(1) Bob menerima *e-mail* dari Alice.
(2) Bob memproses *ciphertext* yang diterima dengan

fungsi *hash* yang sama.

(3) Bob meminta Alice untuk menyebutkan *password* tadi.

(4) Dalam aplikasi, hasil *message digest* tadi digabungkan dengan *password* yang otomatis diterima lewat telepon Alice lalu ditandatangani lagi dengan fungsi *hash* yang sama.

(5) Jika otentik, Bob dapat mendekripsi pesan yang diterimanya.



Gambar 3 Skema Protokol

IV. PENGAPLIKASIAN PROTOKOL

Berdasarkan yang telah disampaikan di atas, protokol yang dengan memanfaatkan teknologi *speech recognition* sudah dapat diterapkan dengan mudah saat ini. Perangkat keras yang diperlukan cukup dengan komputer dan telepon yang terhubung saja karena saat ini sudah ada perangkat lunak yang mendukung mendukung teknologi tersebut.

Saat ini sistem operasi Windows Vista dan Windows 7 mulai banyak digunakan di berbagai komputer. Dengan Speech API (SAPI) minimal versi 5.3 yang dibawanya dan.NET minimal versi 3.0, pengembang dapat membuat aplikasi yang dapat memanfaatkan perintah-perintah yang dikeluarkan oleh suara manusia.

Dalam makalah ini, disertakan contoh implementasi berupa aplikasi yang dibuat untuk mengaplikasikan protokol yang telah disebutkan sebelumnya dengan memanfaatkan teknologi *speech recognition*. Aplikasi yang dibuat dibangun dengan menggunakan Microsoft Visual Studio 2008 dengan .NET Framework 3.5 dengan sistem operasi Windows 7. Aplikasi yang dibuat adalah *add-in* dengan VSTO untuk Microsoft Outlook 2007 berupa *ribbon* yang tampak ketika membuka *e-mail* masuk atau ketika akan mengirimkan *e-mail*. Berikut ini desain aplikasi *add-in* yang dibuat.

Gambar 4 Desain Ribbon Add-in

Dalam implementasi sebenarnya, aplikasi harus dibuat terpisah untuk pengirim, yaitu hanya bisa mengenkripsi, dan untuk penerima. Namun pada gambar desain di atas, tidak diimplementasikan dalam aplikasi yang terpisah. Akan tetapi, dalam kasus di atas, Bob sebaiknya memiliki fitur aplikasi yang lengkap seperti gambar di atas, terutama untuk pembangkitan kunci dan pendekripsian, sedangkan untuk rekan-rekannya aplikasi hanya berupa pengenkripsi saja.

Komponen-komponen yang terdapat pada aplikasi ini yang digunakan untuk adalah sebagai berikut.

- Pembangkit kunci privat dan kunci publik
  Komponen ini digunakan untuk membangkitkan kunci privat dan publik baru dengan algoritma RSA. *User* dapat memasukkan panjang bit nilai *p* dan *q* (berdasarkan algoritma RSA di atas) lalu memperoleh kunci publik (*e* dan *n*) dan kunci privatnya (*d*).
  Jika pada kasus di atas Bob sudah memiliki kunci privat dan publik dan ingin menggunakannya tanpa harus membangkitkan kunci baru, ia dapat langsung memasukkan kunci privatnya untuk mendekripsi pesan, begitu pula Alice dapat langsung memasukkan kunci publiknya untuk mengenkripsi pesan.
- Pengenkripsi pesan *plaintext* menjadi *ciphertext*
  Jika komponen ini digunakan, secara otomatis *plaintext* pada *body e-mail* dienkripsi. Tentu saja *plaintext* dapat terenkripsi apabila kunci publik telah tersedia. Dalam *body e-mail*, perlu ada aturan atau tanda bahwa teks yang telah terproses merupakan *ciphertext*. Dalam aplikasi yang dibuat ini, *ciphertext* diawali dengan karakter "*" dan diakhiri dengan karakter "#". Tentu saja tidak perlu ada kekhawatiran isi *ciphertext* sama dengan karakter pembatas tersebut karena isi *ciphertext* berupa angka-angka

heksadesimal.

- Pembangkit *message digest* dari pesan terenkripsi
  Dari pesan terenkripsi pada *body e-mail*, dengan komponen ini user dapat memperoleh *message digest* dari *ciphertext* dengan fungsi *hash SHA-1*.
- Penyisipan *signature* ke dalam *body e-mail*
  Jika *ciphertext* telah ada pada *body e-mail*, *user* dapat menyisipkan *signature* pada bagian akhir teks. Dalam aplikasi ini, *signature* diawali dengan karakter "#" (yang merupakan bagian akhir *ciphertext*) dan diakhiri dengan karakter "$". *Signature* dapat diperoleh jika *user* telah memasukkan *password* dan sudah ada *message digest* hasil fungsi *hash* terhadap *ciphertext*. *Password* inilah yang akan diucapkan oleh pengirim ketika penerima akan mendekripsi *ciphertext* yang telah diterimanya. Aplikasi secara otomatis menggabungkan *message digest* dengan *password* kemudian membangkitkan *signature* dengan fungsi *hash SHA-1*.
- Pendeteksi kata yang diucapkan oleh *user*
  Dengan komponen ini, aplikasi pada penerima *ciphertext* akan memperoleh *password* yang diucapkan oleh pengirim melalui media audio.
- Pengecek keaslian dan pendekripsian *ciphertext*
  Keaslian pesan yang dikirimkan dapat dilakukan jika penerima telah mengetahui *password* dan *message digest* dari *ciphertext*. Dengan komponen ini, aplikasi secara otomatis mengambil *ciphertext*, lalu mencari *message digest* dari fungsi *hash SHA-1*, lalu menggabungkannya dengan *password*, dan terakhir mengecek keaslian pesan. Jika pesan asli, maka *body* email secara otomatis berubah menjadi pesan *plaintext*-nya.

Berikut ini skenario contoh implementasi protokol dengan menggunakan aplikasi ini.

1. Bob menentukan kunci privat dan publik:

```
d   = 463125
e   = 8221
n   = 2746363
```

2. Bob memberitahukan nilai *e* dan *n* kepada semua rekannya.
3. Alice ingin mengirimkan pesan rahasia kepada Bob. Ia menyiapkan *plaintext* dan *password* sebagai berikut.

Plaintext:

```
Dear Bob,

Bob, tolong ambil pesanan untuk Perusahaan A di
Jalan Sesama nomor 9.
Tolong jangan sampai Eve mengetahui pesanan
tersebut.
Terima kasih.

Regards,
--
Alice
```

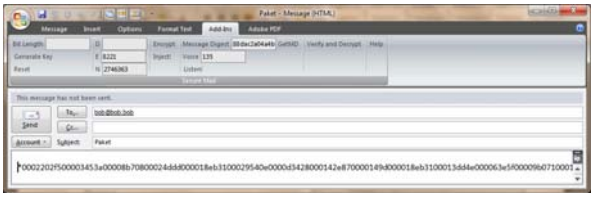
Password:

```
135
```

4. Alice melakukan enkripsi dan injeksi *signature* pesan yang akan dikirimkannya menjadi seperti berikut ini.

```
*0002202f500003453a00008b70800024ddd000018eb31000
29540e0000d3428000142e870000149d000013dd4e000063e
5f00009b07100013dd4e000063e5f00029540e0000d342800
0142e870000149d000018eb310001cace40000d3428000170
e440000d34280000cb9b200005951700018eb3100008b7080
0020bca8000142e8700004413300017 0e4400018eb310000b
b66200003453a00022ae4500008b7080000cb9b200008b708
0000cb9b200018eb3100025de6f0000cb9b20001cace40002
5de6f000294e5500018eb31000152f4500003453a00024ddd
000025de6f00022ae4500008b70800008a2bf00008b708000
08b7080000cb9b200018eb3100008cfa200018eb31000162d
f500004413300018eb3100009bd0d00008b708000170e4400
008b7080000cb9b200018eb31000228ca200003453a00022a
e4500008b70800020bca800008b70800018eb310000cb9b20
000d342800020bca80000d342800024ddd000018eb3100020
40e90000e37c000013dd4e000063e5f0001f3f050000d3428
000170e440000d34280000cb9b200005951700018eb310002
5dedf00008b7080000cb9b200005951700008b7080000cb9b
200018eb3100022ae4500008b70800020bca80000bb662000
08b70800004413300018eb3100003ad42000099b780000345
3a00018eb3100020bca800003453a0000cb9b200005951700
003453a0001cace400008b70800008a2bf00025de6f000044
13300018eb310000bb66200003453a00022ae4500008b7080
000cb9b200008b7080000cb9b200018eb310001cace400003
453a00024ddd000022ae4500003453a000142e8700025de6f
0001cace40000e37c000013dd4e000063e5f0001f3f050000
3453a00024ddd000004413300020bca800008b70800018eb3
1000294e5500008b70800022ae450000441330000 8a2bf000
0e37c000013dd4e000063e5f00009b07100013dd4e000063e
5f0000e0adb00003453a00005951700008b70800024ddd000
0162df500022ae450000149d000013dd4e000063e5f00014d
e8600014de8600013dd4e000063e5f00008cfa2000170e440
000441330001691980000 3453a#
```

5. Alice mengirimkan *e-mail* berisi *ciphertext* dan *signature*-nya kepada Bob.

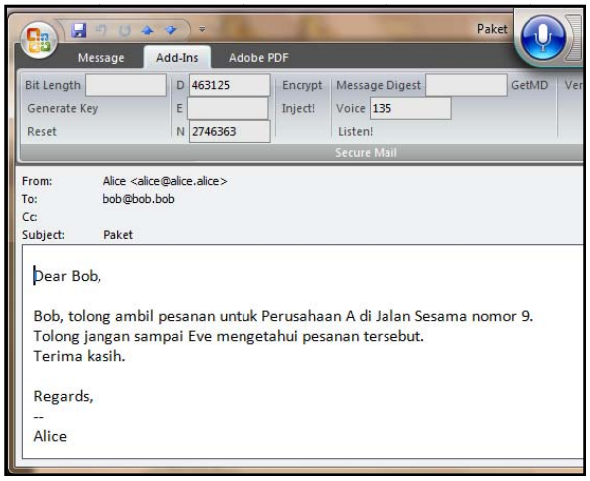


Gambar 5 E-mail yang Dikirimkan Alice

6. Bob menerima *e-mail* dari Alice lalu menelepon Alice untuk menyebutkan *password*. Ia kemudian memasukkan kunci privatnya.

Gambar 6 E-mail yang Diterima Bob

7. Bob mengecek kebenaran pesan lalu secara otomatis aplikasi akan mendekripsikan pesan jika benar.



Gambar 7 Hasil Plaintext yang Diperoleh Bob

IV. Analisis Kelebihan dan Kekurangan Protokol

A. Kelebihan

1. Menggunakan dua algoritma yang sangat baik dan banyak diimplementasikan untuk kriptografi saat ini.
2. Keamanan pesan tetap terjamin meskipun kunci publik diketahui oleh musuh.
3. Dapat membuktikan asal pengirim pesan dengan menggunakan media lain, yaitu audio, sehingga pengiriman pesan melalui *e-mail* palsu dapat dicegah.
4. Penggunaan fungsi *hash* dilakukan secara berlapis sehingga sulit ditemukan hubungan antara *ciphertext* dengan *signature* pada *body e-mail* jika *e-mail* tersebut tercuri.
5. Pengimplementasian protokol dengan memanfaatkan *speech recognition* saat ini dapat dilakukan dengan lebih mudah karena sudah ada *library* yang disediakan secara luas. Selain itu, aplikasi tidak hanya dapat dibuat dalam bentuk aplikasi tersendiri (*stand-alone*), melainkan juga dapat terintegrasi dalam aplikasi untuk *e-mail* dalam bentuk *add-in* atau *plugin*.
6. Di masa depan pemakaian pendeteksi percakapan akan semakin luas dan lazim sehingga protokol ini akan semakin berkembang.

A. Kekurangan

1. Terdapat tiga kali pemrosesan algoritma (satu kali untuk algoritma kunci-publik dan dua kali algoritmafungsi *hash*) sehingga bisa memakan waktu cukup lama dan memerlukan *resource* komputer cukup besar.
2. Tidak dapat mengetahui asal pengirim pesan dan melakukan anti-penyangkalan.
3. Memerlukan komputer yang terhubung dengan media komunikasi suara (misalnya telepon) dan memerlukan kualitas suara yang sangat baik dan aman.
4. Perlunya kesepakatan keadaan *password*, misalnya cara penyebutan, bahasa yang digunakan, dan jenis kata yang digunakan. Aplikasi dapat kesulitan menentukan kata yang dimaksud jika "kamus" yang digunakan terlalu luas dan akibatnya waktu yang diperlukan menjadi sangat lama.
5. Aplikasi memerlukan teknologi *speech recognition* yang saat ini belum bisa dikonfigurasi dengan mudah.

V. Kesimpulan

Protokol kriptografi dengan memanfaatkan *speech recognition* merupakan suatu alternatif yang baik untuk melakukan pengamanan pesan. Aspek keamanan yang dapat dijamin yaitu kerahasiaan pesan, otentikasi, dan keaslian pesan.

Daftar Pustaka

Munir, Rinaldi. 2005. Diktat Kuliah IF5054 Kriptografi. Bandung: Departemen Teknik Informatika Institut Teknologi Bandung.

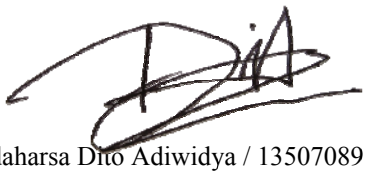
Pengantar Kriptografi
URL: http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/PengantarKriptografi2006.ppt
Waktu akses: 5 Februari 2010 17:19

System.Speech Code Samples
URL: http://www.brains-N-brawn.com/speechSamples
Tanggal akses: 16 September 2009 20:37

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



Bernardino Madaharsa Dito Adiwidya / 13507089