Analisis dan Implementasi Penerapan Enkripsi Algoritma Kunci Publik RSA Dalam Pengiriman Data Web-form

Anton Rifco Susilo [1)]

1) Jurusan Teknik Informatika ITB, Bandung 140132, email: if14046@students.if.itb.ac.id

Abstract – *Metode pengamanan data dalam web yang umum dan luas digunakan adalah dengan menggunakan HTTP Secure (HTTPS) pada SSL untuk mengenkripsi data. Penerapan HTTPS pada SSL ini membutuhkan beberapa tahapan dalam pengamanannya, diantaranya ialah inisiasi jalur komunikasi, pengiriman kunci publik dari server ke client, enkripsi data menggunakan kunci publik, transportasi data, serta penutupan koneksi. Banyaknya tahapan yang ada membuat metode pengamanan ini tidak efisien. Oleh karenanya, diperlukan suatu mekanisme pengamanan lain yang relatif cepat tanpa mengorbankan faktor keamanan yang ada.*

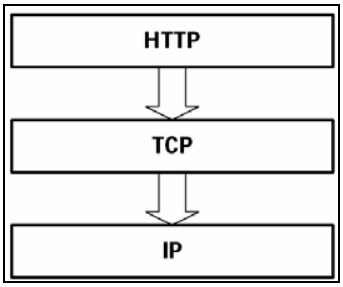
Salah satu alternatif yang ada ialah penerapan algoritma enkripsi RSA dalam transaksi data antara client-server. Ketika suatu halaman web yang berisi form dibangkitkan oleh web server, diberikan suatu fungsi javascript untuk mengenkripsi data yang ada menggunakan kunci publik dari web server. Di sini user akan mengisi data form-nya hingga selesai. Ketika pengguna menekan tombol submit, data-data yang ada akan dienkripsi sebelum dikirimkan.

Makalah ini akan membahas mengenai analisis dan implementasi algoritma enkripsi kunci publik RSA dalam transaksi web antara client dengan server. Bagian analisis dari makalah ini akan membandingkan metode yang dibuat dengan metode-metode pengamanan serupa yang telah ada, semisal : metode HTTPS pada SSL. Sementara itu, implementasi dari metode ini akan dipecah menjadi dua bagian, di sisi client dengan menggunakan javascript untuk enkripsi, serta di sisi server dengan menggunakan script php untuk dekripsi.

Kata Kunci*: Penyandian, Nir-penyandian, Algoritma RSA, Pemroses-form*

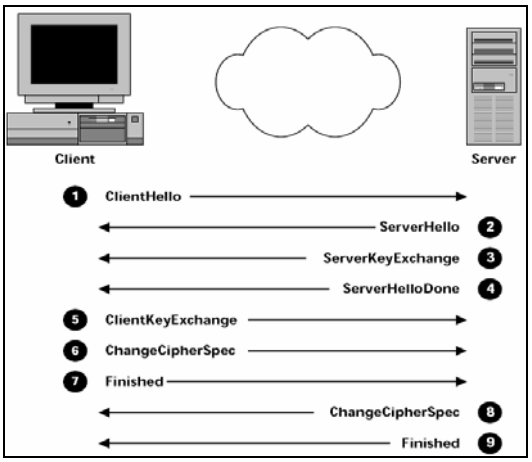
1. PENDAHULUAN

Dalam transportasi data melalui aplikasi web form, umumnya data dikirimkan melalui parameter ***Get*** ataupun ***Post*** pada koneksi HTTP yang diciptakan. Biasanya data tersebut dikirimkan begitu saja tanpa prosedur pengamanan yang berarti. Semua data, baik itu parameter autentikasi *login*, seperti *username* dan *password*, maupun parameter *request* lainnya dapat dilihat dengan mudah jika ada yang berhasil menyusup ke dalam jaringan internet. Saat ini, protokol pengamanan yang banyak digunakan oleh situs web yang menjadikan faktor keamanan sebagai prioritas utama, seperti : situs perbankan dan jual-beli, ialah protokol SSL melalui HTTPS (*Secure* HTTP). Pada penerapan protokol ini, pihak penyedia, atau web *server* wajib memiliki sertifikat autentikasi dari penyedia layanan terkait, seperti : ***Verisign***®. Kelemahan yang paling utama dari protokol ini ialah lambatnya dalam memproses autentikasi dikarenakan banyaknya tahapan yang terjadi. Oleh karena kelambatan ini, seringkali *client* yang mengakses dengan *bandwidth* internet kecil terputus koneksinya.



Gambar 1 Arsitektur Komunikasi web standar

Atas dasar itu, dibutuhkan suatu mekanisme pengamanan alternatif dalam transportasi data dalam dunia web. Suatu alternatif yang dapat menjawab tantangan kecepatan memproses tanpa mengorbankan faktor keamanan yang ada.



Gambar 2 Tahapan dalam protokol SSL

Mekanisme pengamanan yang akan dibahas pada makalah ini ialah dengan menerapkan konsep penyandian kunci publik RSA pada data form web.

2. ALGORITMA PENYANDIAN RSA

Pengenalan Algoritma RSA

Algortima RSA dijabarkan pada tahun 1977 oleh tiga orang: **Ron Rivest**, **Adi Shamir** dan **Len Adleman** dari ***Massachusetts Institute of Technology***. Nama RSA itu sendiri berasal dari inisial nama mereka (**Rivest—Shamir—Adleman**).

Algoritma ini diciptakan berdasarkan fakta sulitnya memfaktorkan suatu bilangan (bernilai besar) menjadi dua buah faktor primanya. Bekerja dengan konsep kunci publik dan kunci private. Kunci publik dapat disebarkan secara bebas; dengan kunci publik pengguna hanya dapat menyandikan data-untuk melakukan proses nirsandi, pengguna harus memiliki kunci private yang bersesuaian dengan kunci publik. Sangatlah sulit untuk membangkitkan kunci private dari kunci publik yang diberikan (walaupun secara teori dapat dilakukan). Hal tersebut membuat algoritma RSA sangat populer dalam penyandian data.

Cara Kerja Algoritma RSA

Berikut ini akan dijelaskan cara penggunaan algoritma RSA beserta contohnya – untuk kemudahan akan digunakan angka-angka yang kecil sebagai contoh.

- Bangkitkan dua buah bilangan prima, **p** dan **q**. Misalnya : p = 7; q = 19
- Inisiasi variabel **n**, sehingga **n = p . q**
  Berarti : n = 7 . 19 = 133
- Inisiasi variabel **m**, sehingga **m = (p - 1)(q - 1)**
  Berarti : m = (7 - 1)(19 - 1) = 108
- Inisiasi variabel **e**, dimana e merupakan ko-prima dari **m**, berarti bahwa **e** faktor persekutuan terbesar dari **m**
  Berarti : e = 5 , karena fpb(5, 108) = 1
- Cari **d** dimana **d.e % m = 1**, setara dengan mencari solusi integer dari **d = (1 + zm) / e**, dengan **z** mulai 0 hingga ditemukan solusi integer untuk **d**
  Berarti :
  z = 0 => d = 1 / 5 (real)
  z = 1 => d = 109 / 5 (real)
  z = 2 => d = 217 / 5 (real)
  z = 3 => d = 325 / 5 = 65 (integer) – dipilih

Dari variabel-variabel tersebut, didapat publik key : **n** dan **e** (133 dan 5), serta private key : **n** dan **d** (133 dan 65)

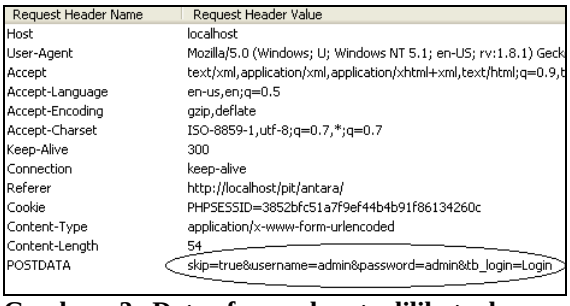
- Untuk melakukan penyandian, digunakan **C = P^e % n** ,dimana **P** adalah pesan yang akan dienkripsi dan C pesan yang telah terenkripsi.
  Misalkan pesan yang akan dienkripsi adalah 6
  C = 6^5 % 133 = 7776 % 133 = 62
- Sementara, untuk melakukan nir-penyandian, digunakan **P = C^d % n**
  Berarti :
  P = 62^{65} % 133 = 6

  Karena proses dekripsi dari data terenkripsi menghasilkan data awal, maka dapat dipastikan bahwa algoritma RSA ini valid.

3. MEKANISME PENGAMANAN

Dari penjelasan di atas, terbukti bahwa algoritma RSA bersifat valid, sehingga dapat kita gunakan untuk proses penyandian data dalam mekanisme pengamanan yang akan diimplementasi.

Sebelumnya akan dijelaskan mengenai bagaimana rapuhnya transportasi data web dengan cara konvensional. Pada cara konvensional, semua data yang dikirimkan dicantumkan pada variabel POST dan GET dari koneksi HTTP. Seperti variabel HTTP yang lain (seperti **Cookie, Content-Type**), semua data tersebut dapat dilihat sejelas-jelasnya dengan bantuan beberapa perangkat *hacking*.



Request Header Name	Request Header Value
Host	localhost
User-Agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1) Geck
Accept	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,t
Accept-Language	en-us,en;q=0.5
Accept-Encoding	gzip,deflate
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive	300
Connection	keep-alive
Referer	http://localhost/pit/antara/
Cookie	PHPSESSID=3852bfc51a7f9ef44b4b91f86134260c
Content-Type	application/x-www-form-urlencoded
Content-Length	54
POSTDATA	skip=true&username=admin&password=admin&tb_login=Login

Gambar 3 Data form dapat dilihat dengan mudah

Mekanisme pengamanan yang diusulkan ini tidak berusaha untuk mengamankan jalur komunikasinya, tetapi lebih kepada pengamanan terhadap data yang ditransmisikan.

Mengikuti algoritma RSA, dalam mekanisme pengamanan yang diusulkan juga terdapat dua (2) tahapan utama, yakni proses penyandian (*encryption*) serta nir-penyandian (*decryption*).

Berikut ialah penjelasan dari kedua tahapan tersebut:

- **Penyandian (*Encryption*)**
  Setelah *client* mengisi semua data pada web form dan menekan tombol submit, terdapat sebuah fungsi *javascript* yang akan mengenkripsi semua data-nya dengan menggunakan algoritma RSA -

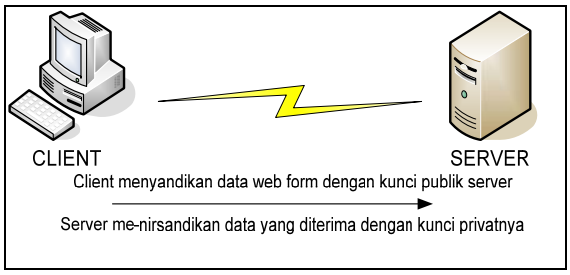
kunci publik milik *server*. Setelah semua data tersandikan, *browser* akan mengirimkannya kepada *server* dengan metode ***post*** ataupun ***get***.

Fungsi *javascript* penyandian dibangkitkan oleh web *server*, dengan parameter kunci publik yang dimiliki oleh *server* dan pesan yang akan dienkripsi dari web form yang diisikan oleh *client*. *Server* tidak perlu mengeluarkan kunci privatenya pada halaman web – hanya akan digunakan pada tahapan kedua yang berlangsung di sisi *server*, sehingga keamanan dapat dipertahankan.

Di sini, jika terdapat pihak ketiga yang berhasil menyadap data, ia hanya akan mendapatkan data yang telah tersandi. Dengan asumsi bahwa tanpa kunci private, pengguna tak dapat men-dekripsi data, maka data yang didapat pihak ketiga ini tidak bernilai.

- **Nir-sandi (*Decryption*)**
  Di sisi *server*, data yang diterima kemudian akan di nir-sandikan dengan algoritma RSA pula – kunci private. Sehingga data-data yang dimaksud oleh *client* dapat diterima oleh *server* dengan baik.

  Berbeda dari proses penyandian yang dibuat dengan script pemrograman *javascript*, pada proses nir-penyandian, fungsi akan diciptakan dengan menggunakan script *server*, seperti **php** ataupun **jsp**.



Gambar 4 Mekanisme pengamanan dengan RSA

Dibandingkan dengan protokol SSL yang mempunyai sembilan (9) tahapan, pada mekanisme yang diusulkan ini hanya menerapkan dua (2) tahapan, sehingga diharapkan kinerja dari mekanisme ini akan lebih baik serta optimal. Ukuran kinerja dari mekanisme ini akan sangat bergantung dari kecepatan memproses penyandian dan nir-penyandian dari algoritma RSA yang diterapkan. Sementara untuk kekuatan penyandiannya, kita dapat mempercayakan kekuatan RSA, telah digunakan sejak 1983 dengan paten **U.S. Patent 4405829**. Selama belum ditemukan algoritma efektif yang dapat memfaktorkan suatu bilangan besar menjadi dua faktor prima-nya, algoritma ini akan berjalan dengan aman. Sementara itu, dengan cara *bruteforce*, RSA dengan 512 bit dapat dipecahkan dalam waktu 5,2 bulan di luar pemilihan waktu polinomial. Jika termasuk pemilihan polinomial maka waktu totalnya sekitar 7,1 bulan. Perlu diperhatikan bahwa pemilihan polinomial bisa mengurangi waktu total secara drastis dengan menggunakan lebih banyak lagi mesin.

Untuk algoritma RSA ini, semakin panjang jumlah bit kunci yang dipakai, akan semakin sulit dipecahkan menggunakan *bruteforce*. Bahkan dari pihak pencipta RSA sendiri berani mengeluarkan sayembara dengan hadiah sebesar $200.000 atau sekitar 1,9 milyar rupiah bagi siapa saja yang bisa memfaktorkan bilangan RSA-2048 bit (617 digit) – **http://www.rsasecurity.com/rsalabs/challenges/factoring/challengenumbers.txt**

4. STUDI KASUS

Dalam makalah ini, penulis akan memberikan sebuah studi kasus penerapan mekanisme pengamanan web. Skenario yang akan diterapkan ialah proses autentikasi *login*, yang akan mentransmisikan data *login* (*username* dan *password*). Dari skenario tersebut, akan diterapkan mekanisme pengamanan dengan protokol SSL serta penyandian data dengan algoritma RSA. Dari hasil studi kasus ini akan dibuat perbandingan antara kedua mekanisme dari segi kecepatan memproses – kecepatan menampilkan halaman web dan kecepatan mentransmisikan data (yang dapat diartikan sebagai seberapa cepat *server* memberikan halaman *response*). Dari perbandingan tersebut, kita dapat menyimpulkan mekanisme mana yang lebih baik (berdasarkan kriteria yang dibuat).

Dalam studi kasus ini, akan dilakukan pada *server* luar-dengan kecepatan *bandwidth* setara 40kbps, dengan menggunakan ***javascript*** dan **php**. Halaman utama yang berisi form *login* akan disamakan untuk kedua pengujian, yakni tampilan sederhana hitam putih (dimaksudkan agar tidak memperlambat pemrosesan). Satuan pengujian adalah *millisecond* (ms) yang akan dihitung oleh *script*. Masing-masing skenario akan dilakukan dua kali.

Pada proses penyandian, fungsi enkripsi *javascript* yang digunakan diambil dari fungsi enkripsi *javascript* yang sudah ada pada **http://www.cs-students.stanford.edu/~twj/jsbn** ,sedangkan fungsi nir-penyandian php yang digunakan dibuat oleh **Khaled Al-Shamaa**, yang diambil dari **http://php-classes.de/class/rsa**

Data yang ditransmisikan ialah : ***username=antonrifco&password=antonrifco*** , dengan nilai kunci publik **p = 9990454949 dan q = 9990450271**

Dengan parameter tersebut, didapat hasil penyandian dengan algoritma RSA ialah **4720037022084721594459396080047895563887**

Berikut adalah hasil pengujian yang dilakukan :

Skenario	Waktu SSL (millisecond)	Waktu RSA (millisecond)
Menampilkan halaman form	150	50
	140	55
Menampilkan response	500	350
	600	350

Hasil pengujian di atas menyatakan bahwa mekanisme pengamanan web form dengan penyandian RSA mempunyai efektivitas waktu lebih baik. Berdasarkan analisis, pada mekanisme pertama (SSL) waktu banyak terbuang untuk inisiasi koneksi dan penutupan koneksi HTTPS, dimana hal tersebut tidak terjadi pada mekanisme kedua (penyandian RSA). Mekanisme pertama menghabiskan waktu inisiasi untuk *loading* file *javascript* yang ada – RSA.js dan BigInteger.js

Pada transportasi data, mekanisme pertama melewatkannya melalui jalur yang diciptakannya sendiri, sementara mekanisme kedua melewatkannya melalui jalur TCP/IP umum. Untuk faktor kehandalan (*reliability*), mekanisme pertama mendapatkan nilai plus, karena ia akan memastikan bahwa *server* menerima data yang dimaksud oleh *client* – sementara mekanisme kedua tetap mengandalkan *unreliable connection* dari TCP/IP, yang berarti tidak ada kepastian bahwa *server* akan mendapatkan data yang dimaksud. Tetapi, untuk keunggulan ini mekanisme pertama dipastikan kehilangan banyak waktu.

Berikut ini adalah hasil analisis perbandingan antara mekanisme pengamanan SSL dan penyandian RSA:

Tahap Inisiasi
SSL : Menciptakan koneksi HTTPS antara *client* dengan
RSA : Loading fungsi-fungsi penyandian javascript – RSA.js, BigInteger.js
Transportasi
SSL : Melalui jalur komunikasi yang telah diciptakan antara *client-server*
RSA : Melalui jalur komunikasi umum TCP/IP
Format Data
SSL : Data terenkripsi dengan algoritma kunci publik
RSA : Data terenkripsi dengan algoritma RSA
Reliability
SSL : Sangat *reliable*
RSA : Cukup *reliable*, sesuai dengan *reliability* protokol TCP/IP
Faktor Kesukaran
SSL : Untuk menerapkannya secara sah, *server* harus mempunyai sertifikasi SSL
RSA : Untuk menerapkannya, setiap halaman form yang dibangkitkan harus pula membangkitkan fungsi penyandian javascript

Berdasarkan beberapa faktor yang dibandingkan di atas, seharusnya terlihat bahwa mekanisme pengamanan data dengan penyandian RSA lebih unggul. Tetapi, walaupun begitu, bukan berarti bahwa mekanisme SSL benar-benar tidak dapat diandalkan. Karena selain pengamanan data, protokol SSL juga dapat digunakan untuk menguji keabsahan suatu situs web – untuk menghindari *phising web* (penyamaran situs web dengan membuat tampilan yang mirip)

5. MENINGKATKAN FAKTOR KEAMANAN

Untuk meningkatkan faktor keamanan pada mekanisme pengamanan yang dibahas, berarti kita harus meningkatkan keamanan penggunaan algoritma RSA, dapat dilakukan dengan cara :

- Menggunakan kunci dengan jumlah bit yang besar, misalnya 512 bit. Terbukti bahwa semakin besar bit kunci, semakin sulit untuk dicoba-coba dengan *bruteforce*.
- Mengganti kunci publik – dan oleh karenanya juga mengganti kunci private secara periodik, semisal dua bulan sekali diganti.

Selain mengamankan penggunaan algoritma RSA, juga dapat dilakukan dengan pengamanan tambahan lain, misal : penyandian fungsi *javascript* serta php yang digunakan. Hal ini bertujuan agar pengguna seminimal mungkin mengetahui mengenai prosedur pengamanan yang diterapkan.

6. KESIMPULAN

Dalam komunikasi data rahasia dalam dunia web, dibutuhkan sebuah mekanisme pengamanan data. Mekanisme tersebut harus dapat memenuhi kebutuhan akan kecepatan memproses serta handal dalam pengamanannya. Protokol SSL yang selama ini dipakai telah terbukti tidak dapat memenuhi kriteria pertama, kecepatan. Makalah ini membahas salah satu alternatif pengamanan yang dapat digunakan; dengan menerapakan konsep penyandian kunci publik RSA. Sebagai bahan perbandingan, disajikan pula studi kasus pengamanan data dengan skenario autentikasi *login*

menggunakan protokol SSL dan penyandian RSA.

7. DAFTAR REFERENSI

[1] AB. Format, “Manuscript Format”, *Proc. Of SITIA*, Jun 2002, Surabaya, Indonesia, pp.6-9.

[2] Munir, Rinaldi. Diktat Kuliah IF5054 Kriptografi. Institut Teknologi Bandung. 2005

[3] Ferguson, Niels dan Schneider, Bruce. Practical Cryptography, John Wiley. 2003

[4] Kelas penyandian kunci publik RSA, **http://php-classes.de/class/rsa** (diakses pada Desember 2007)

[5] Fungsi penyandian RSA Javascript, **http://www.cs-students.stanford.edu/~twj/jsbn** (diakses pada Januari 2008)

[6] Situs *official* algoritma RSA, **http://www.rsasecurity.com/rsalabs/**