Pengamanan Situs dengan Enkripsi *Head* dan *Body* HTML Menggunakan Algoritma RC4

Yudi Haribowo

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
e-mail: if14111@students.if.itb.ac.id

ABSTRAK

Dalam makalah tugas akhir ini dikembangkan aplikasi situs berbagi *file* yang menggunakan sistem enkripsi-dekripsi halaman HTML sebagai metode otentikasinya. Sistem ini dikembangkan dikarenakan metode otentikasi yang sudah umum digunakan, yaitu sistem *password* memiliki celah sehingga penyadap bisa mencuri *password user*. Algoritma enkripsi-dekripsi yang digunakan dalam sistem ini adalah RC4 karena algoritma tersebut cepat sehingga aplikasi terhindar dari *bottleneck*.

Aplikasi ini terdiri dari dua bagian terpisah, yaitu *encryptor* dan *decryptor*. Modul *encryptor* sekaligus pembangun aplikasi situs berbagi *file yang* berada di sisi *server* menggunakan bahasa pemrograman PHP sebagai pembangunnya. Di sisi lain, modul *decryptor* yang berada di sisi klien menggunakan JavaScript sebagai bahasa pemrogramannya. Modul *decryptor* diintegrasikan menggunakan *web browser* Mozilla Firefox yang dilengkapi *plug-in* Greasemonkey sehingga bisa berkomunikasi dengan modul *encryptor*. Setelah dilakukan pengujian terhadap aplikasi, diperoleh hasil bahwa sistem yang dikembangkan berhasil mengamankan transmisi halaman HTML dari *server* ke klien sehingga bisa menjadi alternatif proses otentikasi selain sistem *password.*

Kata kunci: situs berbagi *file*, enkripsi-dekripsi halaman HTML, RC4, otentikasi

Makalah diterima 11 Juni. Revisi akhir 10 Juni 2008.

1. Pendahuluan

Perkembangan teknologi informasi dewasa ini tidak terlepas dari peran internet sebagai tulang punggung utamanya. Berbagai macam layanan dapat disediakan dalam sistem yang melibatkan jaringan seluruh dunia. Salah satu layanan utama yang disediakan dalam sistem internet adalah situs *web*.

Dalam sebuah situs, *server* bisa menyediakan berbagai layanan bagi kliennya, seperti berita, ilmu pengetahuan, riset, hiburan dan sebagainya. Sebagian besar situs di internet bisa diakses secara bebas oleh siapa saja yang membutuhkan layanan dari situs tersebut. Artinya, setiap klien mempunyai hak akses terhadap segala *content* situs. Akan tetapi, tidak semua situs bisa diakses secara bebas oleh setiap klien. Hal ini bisa disebabkan oleh kerahasiaan *content* situs tersebut sehingga *server* tidak ingin klien yang tidak berhak mengaksesnya.

Metode yang umumnya digunakan dalam pengamanan situs dari akses klien yang tidak terotentikasi adalah dengan sistem *password*, yaitu suatu cara dimana klien yang ingin mengakses halaman sebuah situs diminta memasukkan *username* dan *password* yang sah atau sudah disetujui oleh *server*. Sistem ini sudah dianggap cukup aman sehingga banyak diterapkan di berbagai situs. Namun, dalam sistem ini terdapat suatu celah yang bisa disusupi serangan oleh pihak ketiga, yaitu ketika *user* mengirimkan *username* dan *password* miliknya ke *server* dengan *method* POST yang digunakan. Hal ini bisa dimanfaatkan oleh pihak ketiga tersebut dengan melakukan *eavesdropping* (penyadapan) terhadap *username* dan *password* yang dikirimkan. Contoh situs yang menggunakan sistem ini antara lain situs-situs surat elektronik seperti Yahoo!, Gmail atau situs surat elektronik milik ITB (students.itb.ac.id).

Seperti disebutkan sebelumnya, penggunaan enkripsi bisa mencegah pencurian data oleh pihak luar. Algoritma enkripsi sendiri terdiri dari *cipher* blok dan *cipher* aliran. *Cipher* aliran memiliki keuntungan dibanding *cipher* blok dilihat dari sisi kompleksitas dan kecepatan komputasinya karena hanya bergantung dari algoritma yang digunakan. Selain itu, *cipher* aliran memiliki perambatan kesalahan yang kecil[BLE07]. Berdasarkan kunci yang digunakan, algoritma enkripsi terbagi menjadi algoritma kunci simetri dan kunci asimetri yang juga dikenal sebagai kunci publik. Salah satu algoritma kunci simetri yang dikenal adalah RC4. Algoritma ini merupakan salah satu algoritma *cipher* aliran. RC4 terkenal dengan kecepatan

prosesnya. Hal ini ditulis di [BUD98] dengan membandingkan RC4 dan Blowfish. Menurut hasil pengujian kecepatan algoritma kriptografi RC4 adalah 5.380,035 Kbyte/detik pada Pentium133 memori 16MB pada Windows 95 sedangkan algoritma Blowfish pada jenis komputer yang sama menghasilkan kecepatan 2.300 KByte/detik.

2. Kriptografi

Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya[RIN06]. Kriptografi menyediakan empat layanan, yaitu kerahasiaan, integritas data, otentikasi dan nipernyangkalan.

Algoritma kriptografi adalah aturan untuk melakukan cipherisasi dan decipherisasi atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi pesan[RIN06]. Algoritma kriptografi berkembang terus dan terbagi atas dua jenis yaitu algoritma kriptografi klasik dan algoritma kriptografi modern. Algoritma kriptografi klasik hanya menggunakan operasi sederhana untuk melakukan enkripsi terhadap ciphertext sehingga cukup mudah dipecahkan. Oleh karena itulah, algoritma ini sudah jarang digunakan saat ini. Akan tetapi, algoritma kriptografi klasik merupakan dasar dari algoritma kriptografi modern. Algoritma jenis ini terdiri dari dua tipe dilihat dari operasi dasar yang digunakan untuk melakukan enkripsi, yaitu *cipher* substitusi dan *cipher* tranposisi.

Selain algoritma kriptografi klasik, dikenal juga algorimta kriptografi modern. Tidak seperti algoritma klasik yang menggunakan pemrosesan sederhana, algoritma modern menekankan pada kompleksitas sehingga sulit dipecahkan. Algoritma kriptografi modern umunya beroperasi dalam mode bit. Operasi dalam mode bit berarti semua data dan informasi (kunci, *plaintext*, dan *ciphertext*) dinyatakan dalam bit biner 1 dan 0.

Algoritma kriptografi Rivest Code 4 (RC4) merupakan salah satu algoritma kunci simetris dibuat oleh RSA Data Security Inc. (RSADSI) yang berbentuk *stream cipher*. Secara umum algoritma RC4 terdiri dari tahap-tahap sebagai berikut:

1. Inisialisasi S-Box dan *array* kunci
2. Populasi S-Box
3. XOR S-Box dengan *Plaintext*/*Ciphertext*

3. Hypertext Transfer Protocol (HTTP)

Untuk menyampaikan halaman *web* dari *server* ke klien digunakan protokol yang dinamakan HTTP. Protokol HTTP bersifat *request-response*, yaitu klien menyampaikan pesan *request* ke *server* dan *server* memberikan *response* yang sesuai. Komunikasi protokol HTTP terdiri atas pesan *request* yang diberikan oleh *user agent* dan *response* yang diberikan oleh *server*.

Pesan HTTP terdiri atas baris mulai, *header* pesan, dan isi pesan beserta *entity* (opsional). *Header* pesan dan isi pesan dipisahkan oleh sebuah baris kosong, yaitu hanya berisi karakter CRLF. Baris mulai pada pesan *request* berisi pesan permintaan dari klien, sementara pada pesan *reponse*, baris ini berisi status *response* atas *request* yang diterima. *Header* pesan terdiri dari empat jenis, yaitu *header* pesan umum, *header request*, *header response*, dan *header entity* sedangkan isi pesan digunakan untuk mengirimkan *entity*. Dalam *header* dan *entity* inilah semua komunikasi antara klien yang mengakses suatu situs dengan *server* dilakukan.

4. Analisis Masalah

Masalah yang ingin diselesaikan dalam makalah tugas akhir ini adalah bagaimana mengimplementasikan sistem otentikasi di sisi klien sehingga *user* tidak perlu mengirimkan *username* dan *password* ke *server* dalam *entity* HTTP sehingga tidak bisa disadap. Dalam sistem ini, s*erver* yang menerima *request* langsung mengirimkan halaman yang sebelumnya hanya bisa diakses setelah klien melakukan otentikasi. Akan tetapi, halaman tersebut dikirimkan dalam bentuk cipherteks. Setelah diterima oleh *web browser* di sisi klien, halaman cipherteks tersebut di dekripsi sebelum ditampilkan.

4.1. Penanganan Enkripsi di Sisi *Server*

Dalam sistem baru yang diajukan dalam makalah tugas akhir ini, sistem tidak mengirimkan halaman *login* kepada *user* melainkan langsung halaman yang diminta dengan terlebih dahulu mengenkripsi halaman tersebut dengan *password user* yang bersangkutan. Oleh karena itu, dibutuhkan modul enkripsi di sisi *server web*. Pada sistem ini, komunikasi berjalan searah yaitu dari *server* ke klien sehingga modul yang dibutuhkan di sisi *server* hanyalah modul enkripsi. Modul ini dapat diimplementasikan dalam bentuk sebuah sub program (sebuah *class server side scripting)* yang bertugas melakukan enkripsi seluruh halaman *web* yang akan dikirim.

Algoritma yang digunakan untuk melakukan enkripsi adalah algoritma RC4. Algoritma ini dipilih karena dalam pengaksesan situs *web*, hal yang menjadi pertimbangan utama adalah kecepatan akses dan keamanan. Algoritma RC4 sendiri sangat cepat dalam melakukan enkripsi-

dekripsi. Hasil percobaan menggunakan komputer Pentium Core 2 Duo E4500 2.2 GHz dan memori 1GB menghasilkan kecepatan enkripsi 175-215 Mbps. Dengan asumsi kecepatan *bandwidth server* umumnya adalah 100Mbps, hasil percobaan membuktikan tidak terjadi *bottleneck* di sisi *server*. Hasil percobaan secara lengkap dapat dilihat pada **Tabel 1**. Dari sisi keamanan, RC4 juga aman karena sampai saat ini hanya bisa dipecahkan secara *brute force*.

4.2. Penanganan Dekripsi di Sisi Klien

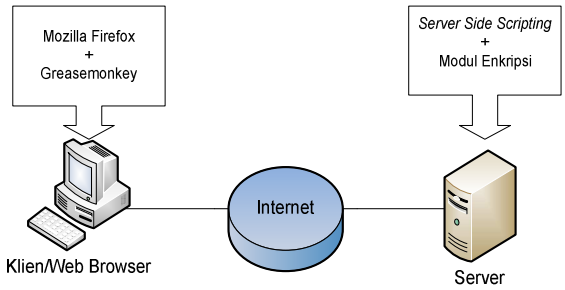
Setelah melakukan *request* ke *server*, klien akan menerima *response* berupa halaman *web* yang dimintanya dalam bentuk terenkripsi. Oleh karena itu, dibutuhkan modul dekripsi di sisi klien. Hal ini dilakukan dengan memanfaatkan *plug-in* untuk *web browser* yang sudah ada, yaitu Mozilla Firefox. Firefox memiliki banyak *plug-in* yang memiliki kemampuan bermacam-macam. Salah satuya adalah Greasemonkey. *Plug-in* ini memiliki kemampuan untuk menyertakan *user script* dimana *user script* ini adalah JavaScript yang bisa dibuat sesuai keinginan *user*. Oleh karena itu, solusi ini menjadi pilihan dalam membuat modul dekripsi di sisi klien karena kemudahan menggunakannya dan aman dari interupsi pihak ketiga (*file* JavaScript tidak ditransfer melalui jaringan karena sudah berada di sisi klien).

4.3. Manajemen Kunci

Tugas akhir ini tidak menangani penyampaian kunci dari *server* ke sisi klien. Sistem mengasumsikan kedua pihak sudah memiliki kunci yang sama untuk melakukan enkripsi dan dekripsi. Penggantian kunci (*password*) juga tidak ditangani dalam tugas akhir ini karena prosedur penggantian kunci bisa menggunakan metode lain yaitu SSL. SSL bisa digunakan kali ini karena penggantian kunci jarang dilakukan. Selain itu, data yang dilewatkan juga sedikit, yaitu sebuah kunci baru sehingga proses bisa berjalan dengan cepat. Di sisi *server*, kunci disimpan dalam basis data yang diasumsikan aman sehingga *user* tidak perlu khawatir kuncinya tercuri.

5. Analisis dan Perancangan Perangkat Lunak

Sistem enkripsi-dekripsi HTML ini nantinya akan diterapkan dalam situs yang sebelumnya menggunakan metode otentikasi berupa *username-password*. Setelah melakukan analisis terhadap dua buah situs, yaitu situs *email* mahasiswa ITB dan situs berbagi *file* Rapidshare, ditentukanlah bahwa perangkat lunak yang akan dikembangkan dalam makalah tugas akhir ini adalah situs berbagi *file* seperti Rapidshare. Dimana dalam situs ini, terdapat mekanisme enkripsi-dekripsi HTML untuk melihat halaman yang berisi daftar *file* yang bisa diunduh. Perangkat lunak ini terdiri dari dua bagian, yaitu *encryptor* yang berada di sisi *server* dan *decryptor* yang berada di sisi klien seperti terlihat pada **Gambar 1**.



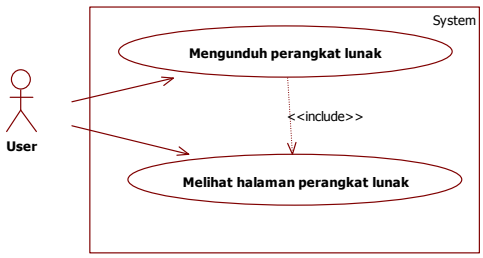
Gambar 1 Deskripsi Sistem

Bagian yang dienkripsi adalah seluruh teks yang berada dalam *tag* **<head>** dan **<body>** sedangkan *content* dari halaman tersebut seperti *image* tidak dienkripsi.

Perangkat lunak yang dikembangkan memiliki kebutuhan sebagai berikut:

1. Mampu mengenkripsi halaman *web* yang diminta oleh klien lalu mentransmisikannya ke klien tersebut.
2. Mampu melakukan dekripsi halaman *web* di sisi klien yang telah dienkripsi oleh *server*.
3. Mampu menampilkan halaman *web* di *browser* sesuai kondisinya sebelum dienkripsi.
4. Mampu menyediakan fasilitas unduh *file-file* aplikasi setelah halaman didekripsi.

5.1. Model Use Case



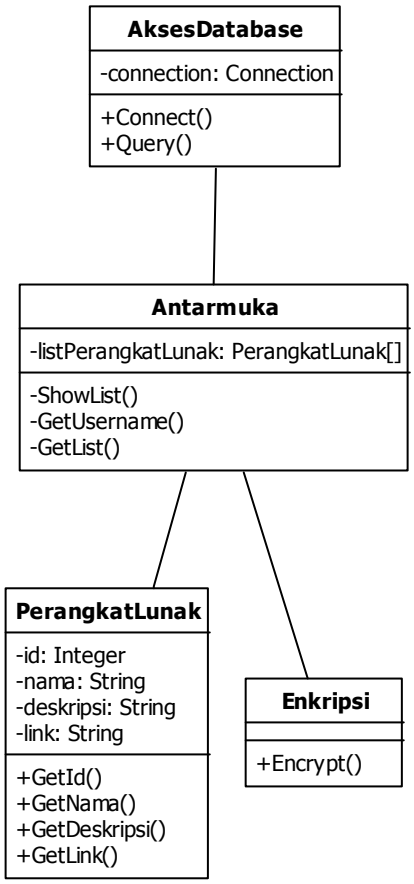
Gambar 2 Diagram *Use Case*

Perangkat lunak ini terdiri dari dua *use case*, yaitu Melihat Halaman Perangkat Lunak dan Mengunduh Perangkat Lunak. Dalam *use case* Melihat Halaman Perangkat Lunak, aktor bisa membuka halaman situs berisi daftar perangkat lunak yang disediakan aplikasi sedangkan dalam *use case* Mengunduh Perangkat Lunak, aktor bisa mengunduh perangkat lunak yang disediakan

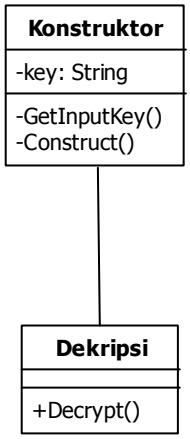
aplikasi. Diagram *use case* dapat dilihat pada **Gambar 2**.

5.2. Diagram Kelas

Identifikasi kelas dilakukan berdasarkan hasil analisis perangkat lunak. Karena menggunakan Greasemonkey yang memfasilitasi *user side scripting*, kelas perangkat lunak ini terdiri dari dua bagian, yaitu yang berada di sisi *server* dan yang berada di sisi klien. Kelas yang berada di sisi *server* antara lain kelas Antarmuka, Enkripsi, AksesDatabase, dan PerangkatLunak sedangkan yang berada di klien adalah kelas Konstruktor dan Dekripsi. Diagram kelas *server* dapat dilihat pada **Gambar 3** dan diagram kelas klien pada **Gambar 4**.



Gambar 3 Diagram kelas *server*



Gambar 4 Diagram kelas klien

6. Implementasi

Kelas-kelas analisis kemudian diimplementasikan sesuai posisinya, di *server* atau di klien. Kelas yang berada di sisi *server* diimplementasikan menjadi *file* PHP sedangkan kelas klien diimplementasikan menjadi *file* JavaScript. Implementasi ini sendiri dilakukan dalam lingkungan sebagai berikut:

1. Prosesor *Intel Core 2 Duo* E4500 2.2GHz.
2. Memori RAM 1GB.
3. Kapasitas *harddisk* 200 GB.
4. Sistem Operasi : *Microsoft Windows XP Professional Service Pack 2*.
5. *Server* basis data : *MySQL 5.0.27*.
6. *Server web* : *Apache PHP 2.2.3*
7. Bahasa Pemrograman : *PHP, HTML, JavaScript*.
8. Kakas pengembangan modul : *Notepad++*
9. *Browser* : *Mozilla Firefox 3 Beta 5* dilengkapi dengan *plug-in* Greasemonkey.

6.1. Implementasi Modul Perangkat Lunak

Modul perangkat lunak diimplementasikan dengan menggunakan kakas Notepad++ dengan bahasa pemgrograman PHP dan JavaScript. Implementasi kelas perangkat lunak dapat dilihat pada **Tabel 2**.

Tabel 2 Implementasi Kelas

No	Nama Kelas	Nama File Fisik
1.	Antarmuka	index.php & view.php
2.	Enkripsi	Enkripsi.php
3.	Akses Database	AksesDatabase.php
4.	Perangkat Lunak	PerangkatLunak.php
5.	Konstruktor	zapshare.user.js
6.	RC4	zapshare.user.js

6.2. Implementasi Antarmuka

Pengembangan antarmuka ini menggunakan pendekatan *scripting* PHP sehingga tidak ada kelas yang dihasilkan melainkan sebuah *file* PHP untuk masing-masin antarmuka. Daftar *file-file* fisik yang berisi implementasi antarmuka dapat dilihat pada **Tabel 3**.

Tabel 3 Implementasi Kelas

No	Antarmuka	Nama File Fisik
1.	*Index*	index.php
2.	*View*	view.php

7. Pengujian

Pengujian perangkat lunak merupakan aktivitas menjalankan perangkat lunak dengan berbagai cara yang bertujuan untuk melakukan evaluasi terhadap perangkat lunak yang dibuat serta mendeteksi atau menemukan kesalahan perangkat lunak. Tujuan pengujian disesuaikan dengan kebutuhan perangkat lunak, yaitu apakah perangkat lunak mampu:

1. Mengenkripsi halaman *web* yang diminta oleh klien lalu mentransmisikannya ke klien tersebut.
2. Melakukan dekripsi halaman *web* di sisi klien yang telah dienkripsi oleh *server*.
3. Menampilkan halaman *web* di *browser* sesuai kondisinya sebelum dienkripsi.
4. Menyediakan fasilitas unduh *file-file* aplikasi setelah halaman didekripsi.

Berdasarkan hasil pengujian, diperoleh hasil bahwa perangkat lunak berhasil memenuhi seluruh kebutuhan awal pengembangan karena perangkat lunak berhasil menampilkan halaman situs seperti kondisinya sebelum dienkripsi dan mengirimkan *file* yang diminta *user*.

8. Kesimpulan dan Saran

Kesimpulan yang dapat diambil antara lain:

1. Perangkat lunak berhasil melakukan enkripsi halaman HTML yang dihasilkan melalui *server side scripting*.
2. Perangkat lunak berhasil melakukan dekripsi halaman HTML terenkripsi yang diterima dari *server*.
3. Perangkat lunak berhasil menyediakan fasilitas *download file* kepada *user* yang berhak.
4. Perangkat lunak berhasil mengimplementasikan sistem otentikasi di sisi klien.
5. Perangkat lunak tidak berhasil menampilkan setiap *tag* HTML yang ada.

Saran yang bisa diberikan untuk pengembangan perangkat lunak lebih lanjut antara lain:

1. Pada perangkat lunak perlu diimplementasikan cara untuk mengamankan transmisi data dari klien ke *server*.
2. Perangkat lunak selain enkripsi halaman HTML, perangkat lunak hendaknya dilengkapi juga dengan enkripsi *content* dari halaman *web* tersebut.

REFERENSI

[BUD98] http://www.bimacipta.com/rc4.htm, diakses 23 April 2008 20.40 WIB

[RIN06] Munir, Rinaldi. *Kriptografi*. Program Studi Teknik Informatika. 2006.

DAFTAR PUSTAKA

B. Schneier, *Aplied Cryptography 2rd*, John Wiley,& Sons, 1996.

Ferguson, Niels dan Bruce Schneier, *A Cryptographic Evaluation of IPsec*, dapat diperoleh di http://www.schneier.com/paper-ipsec.pdf

Munir, Rinaldi. *Kriptografi*. Program Studi Teknik Informatika. 2006.

Purbo, Onno W. *TCP/IP-Standar, Desain dan Implementasi*. Elex Media Komputindo. 2001.

Tanenbaum, Andrew S. *Computer Networks*. Prentice Hall. 2003.

Tabel 1 Kecepatan enkripsi RC4

Jumlah	Kecepatan Thread ke (dalam Mbps)										
Thread	1	2	3	4	5	6	7	8	9	10	Total
1	91.42										91.42
2	87.67	86.48									174.15
3	58.71	61.53	58.71								178.95
4	41.29	56.63	48.48	50.39							196.79
5	36.99	37.42	43.53	41.02	36.36						195.32
10	17.77	25.09	17.97	20.64	19.16	24.24	19.1	20	20.77	30.47	215.21