Penerapan Kriptografi Dalam Javascript Sebagai Alternatif Pengamanan Pada Aplikasi Berbasis Web

Tubagus Andhika Nugraha (13510007)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510007@std.stei.itb.ac.id

***Abstract*—The abstract is to be in fully-justified italicized text, at the top of the left-hand column as it is here, below the author information. The abstract is to be in 9-point, single-spaced type, and may be up to 8 cm long. Define all symbols used in the abstract. Do not cite references in the abstract. Do not delete the blank line immediately above the abstract; it sets the footnote at the bottom of this column. Leave two blank lines after the index terms, then begin the main text. All manuscripts must be in English.**

***Index Terms*—About four key words or phrases in alphabetical order, separated by commas.**

I. PENDAHULUAN

A. Latar Belakang

Penggunaan internet, terutama teknologi web, memegang peranan yang semakin penting dalam kehidupan manusia. Masyarakat semakin menjadi masyarakat yang terhubung (Sashongko, 2013) dan semakin bergantung pada internet. Teknologi web menjadi fokus perhatian utama karena menjadi platform aplikasi yang terbuka dan mudah digunakan, karena cukup menggunakan penjelajah untuk membuka situs-situs dan aplikasi-aplikasi web apapun. Web digunakan untuk pendidikan, penelitian, hiburan, budaya, bahkan politik dan kegiatan ekonomi. Transaksi-transaksi krusial seperti pembelanjaan daring membutuhkan pengamanan yang ketat, sehingga keamanan jaringan menjadi kebutuhan yang penting.

Umumnya, pengamanan jaringan web menggunakan protokol HTTPS yang memanfaatkan SSL (Secure Sockets Layer) atau TLS (Transport Layer Security). Protokol tersebut memiliki derajat pengamanan yang tinggi dan dimanfaatkan untuk aplikasi-aplikasi penting seperti perbankan dan pembelanjaan. Akan tetapi, protokol tersebut membutuhkan modal yang besar, dengan server khusus dan sertifikat pengamanan yang harganya mencapai jutaan rupiah untuk kelas termurah. Biaya yang tinggi ini mengakibatkan SSL tidak cocok untuk aplikasi sekala mikro dan kecil yang tetap membutuhkan pengamanan, seperti misalnya aplikasi percakapan (chatting) yang tidak ditujukan untuk peluncuran dan deployment skala sangat besar.

Fokus perkembangan teknologi web sekarang tertuju pada HTML5, CSS, dan Javascript, dan perkembangan menuju pemanfaatan teknologi web sebagai platform dasar pengembangan aplikasi yang sejajar dengan aplikasi native. Dengan demikian, perkembangan pemanfaatan Javascript untuk berbagai macam kebutuhan yang umumnya dilakukan aplikasi native, termasuk kriptografi, menjadi topik yang menarik untuk dibahas.

Makalah yang ditujukan akan memaparkan bagaimana Javascript dapat digunakan untuk kriptografi, dan bagaimana pemanfaatan tersebut dapat digunakan sebagai alternatif pengamanan pada aplikasi berbasis web.

B. Rumusan Masalah

Makalah ini akan membahas:

1. Pengembangan algoritma kriptografi dengan bahasa pemrograman Javascript.
2. Pemanfaatan kriptografi dalam Javascript untuk pengamanan pada aplikasi berbasis web/

C. Batasan Masalah

Makalah ini hanya akan membahas pengembangan satu buah algoritma kriptografi dan studi kasus pemanfaatan algoritma tersebut untuk satu jenis aplikasi. Untuk menunjang makalah yang diajukan, penulis akan mengembangkan pustaka Javascript untuk digunakan dalam studi kasus.

II. TELAAH PUSTAKA

A. JavaScript

JavaScript adalah bahasa pemrograman terinterprestasi (interpreted) berorientasi objek yang umumnya digunakan dalam pengembangan aplikasi berbasis web dan dijalankan oleh aplikasi penjelajah. Sintaks JavaScript distandarisasi sebagai bahasa ECMAScript dan menyerupai bahasa C. Meskipun sintaks JavaScript mirip Java, JavaScript tidak berhubungan langsung dengan Java, dan pertama kali diciptakan oleh Netscape, bukan Sun.

Meski berupa bahasa pemrograman yang berorientasi objek, bahasa JavaScript tidak memiliki fitur kelas, atau disebut dengan *classless language*. Sebagai pengganti kelas, digunakan mekanisme yang disebut dengan *prototyping*, di mana setiap objek dapat dijadikan prototipe untuk objek lainnya. Prototipe sebuah objek

juga merupakan objek sendiri, sehingga dapat dilakukan manipulasi terhadap prototipe sebagaimana objek dapat dimanipulasi. Sintaks *prototype-based language* yang dimiliki JavaScript merupakan hasil pengaruh dari bahasa pemrograman Self. Bahasa lain yang menjadi pengaruh dalam pembuatan bahasa JavaScript adalah bahasa Scheme.

Pada bahasa JavaScript, fungsi diperlakukan sebagai *first-class*. Dengan kata lain, sebuah fungsi adalah sebuah variabel yang memiliki nilai bertipe fungsi. Hal ini berbeda dengan, misalnya, bahasa C, di mana nama fungsi adalah sebuah *keyword* yang tidak dapat diisi ulang. Pada JavaScript, sebuah fungsi dapat dijadikan parameter pada sebuah fungsi lainnya, dan menjadi nilai keluaran (*return*). Fungsi dapat juga bersifat anonim, tanpa nama. Karena fungsi sebenarnya hanyalah berupa nilai sebuah variabel, sebuah variabel bernilai fungsi dapat diganti dengan fungsi lainnya tanpa mengganggu berjalannya program.

Karena JavaScript bersifat *prototype-based* dan memiliki *first-class functions*, metode atau anggota objek yang berupa fungsi juga dapat diubah dengan metode lainnya bahkan setelah metode tersebut didefinisikan. Meski demikian, hal tersebut tidak berlaku untuk metode-metode yang diimplementasikan secara *native* oleh pengeksekusi, misalnya oleh browser. Metode-metode tersebut diimplementasikan dengan menggunakan *native code* yang dihubungkan ke JavaScript, dan tidak dengan JavaScript murni.

Sintaks umum dan fungsi-fungsi dasar JavaScript distandardisasi dalam ECMAScript. ECMAScript versi terbaru adalah versi 5, dan menambahkan fitur-fitur untuk memanipulasi *array*, objek, dan fungsi, seperti misalnya metode .forEach() dan .bind().

Umumnya JavaScript digunakan pada aplikasi web, sehingga aplikasi yang melakukan *parsing* dan mengeksekusi JavaScript adalah penjelajah web. Melalui JavaScript, pengembang web dapat mengakses dan memanipulasi elemen-elemen sebuah laman web secara terprogram (*programatically*). Model yang digunakan disebut sebagai DOM, atau Document Object Model, dan setiap elemen direpresentasikan sebagai sebuah objek. Setiap elemen juga memiliki *event* yang dapat ditangkap dan digunakan oleh JavaScript. Elemen HTML dapat dimanipulasi dengan mengubah isinya, atributnya, ataupun tampilannya dengan mengakses properti-properti CSS.

Sebagaimana standar web lainnya, penerapan JavaScript pada aplikasi web sangat bergantung pada dukungan piranti-piranti penjelajah web. Meski terdapat standar melalui ECMAScript oleh ECMA dan DOM oleh W3C, terkadang terdapat perbedaan antara satu penjelajah web dan penjelajah lainnya. Saat ini terdapat tiga buah *engine* penjelajah web yang paling berpengaruh dan paling signifikan tingkat penggunaannya, yaitu Trident buatan Microsoft yang digunakan pada Internet Explorer, WebKit yang merupakan turunan dari KHTML, sebuah komponen dari projek Konqueror namun kemudian dikembangkan oleh Apple untuk penjelajah Safari dan kemudian oleh Google untuk Chrome, serta Gecko yang digunakan Mozilla dalam Mozilla Firefox. Penggunaan *engine* tersebut berlaku untuk perangkat *desktop* maupun *mobile* terbitan perusahaan yang sama. Masing-masing *engine* memiliki tingkat dukungan yang berbeda-beda.

Hal lain yang membedakan satu *engine* dengan yang lain adalah *engine* JavaScriptnya. *Engine* inilah yang bertugas untuk *parsing* dan mengeksekusi JavaScript sebagai sebuah bahasa sebelum dihubungkan dengan DOM oleh *engine* penjelajah utama. Namun umumnya *engine* tersebut merupakan bagian tak terpisahkan dari *engine* utama sehingga kadang tidak disebutkan. Untuk Internet Explorer, namanya sama, Trident. WebKit yang digunakan untuk Safari dan untuk Chrome memiliki *engine* JavaScript yang berbeda; Chrome menggunakan sebuah *engine* yang dinamakan VP8. Firefox, di sisi lain, terus melakukan pembenahan terhadap *engine* JavaScript yang digunakan sehingga *engine* yang digunakannya pun berbeda-beda. Saat ini *engine* yang digunakan Firefox disebut dengan JagerMonkey.

Dengan berkembangnya peran web dan internet dalam kehidupan sehari-hari, pemanfaatan JavaScript pun meningkat. Awalnya JavaScript hanya digunakan untuk manipulasi objek DOM secara sederhana, namun kini JavaScript menjadi sendi yang penting untuk banyak aplikasi web tercanggih. Hal ini menyebabkan kinerja menjadi sebuah faktor yang penting untuk dipertimbangkan pengguna penjelajah web. Sebagai bahasa terinterpretasi yang dijalankan di dalam penjelajah, kinerja JavaScript pada dasarnya jauh lebih rendah dibandingkan dengan bahasa pemrograman seperti C yang dijalankan secara *native*. Namun karena paradigma JavaScript yang luas, fitur-fitur pada bahasa C bisa dijadikan sebagai sebuah *subset* dari JavaScript yang dapat dijalankan dengan jauh lebih cepat, namun tetap dalam JavaScript. Alhasil lahirlah projek Emscripten, sebuah *cross compiler* yang dapat mengompilasi kode C atau C++ menjadi kode JavaScript. Bersama dengan *engine* terbaru keluaran Mozilla, IonMonkey, kode hasil kompilasi Emscripten dapat berjalan dengan kinerja yang mendekati program *native*, hingga 0,5x kinerja program *native* untuk program yang sama.

Selain pada penjelajah web, JavaScript dapat digunakan juga untuk aplikasi yang bersifat *server-side*. Salah satu contohnya adalah NodeJS, yang memanfaatkan sifat dasar *event-based* dari JavaScript untuk membuat sebuah *web server* berbasis *event* untuk meningkatkan kinerja dan mengoptimalisasi skalabilitas.

B. HTML5

HTML5 adalah sekumpulan teknologi web yang meski dinamakan HTML5 tidak hanya menyangkut HTML. HTML5 bertujuan untuk memungkinkan teknologi web digunakan sebagai platform aplikasi yang layak. Spesifikasi HTML5 melingkupi perbaruan pada

spesifikasi bahasa HTML dan penambahan beberapa API dalam bahasa Javascript.

HTML5 memungkinkan pengembangan aplikasi-aplikasi berbasis web yang sebelumnya tidak mungkin untuk diwujudkan tanpa menggunakan native platform. Salah satu contohnya adalah penggunaan HTML5 untuk menciptakan game engine dengan API <canvas> dan WebGL.

C. XMLHttpRequest

Dalam konteks aplikasi web, salah satu API dalam HTML5 adalah XMLHttpRequest. XMLHttpRequest, atau XHR, adalah sebuah API untuk mengakses *resource* pada server secara asinkron dan dengan dipanggil melalui JavaScript tanpa memodifikasi DOM. Pada awalnya, XHR merupakan sebuah komponen ActiveX yang diciptakan oleh Microsoft, namun kemudian diadopsi oleh penjelajah web lainnya hingga sekarang menjadi kandidat rekomendasi untuk dijadikan standar baku oleh W3C. Pemanfaatan XHR dengan JavaScript sering disebut sebagai AJAX, yang merupakan singkatan dari *Asynchronous JavaScript and XMLHttpRequest*.

Meski bernama XMLHttpRequest, berkas yang diakses dari *server* tidak harus berupa berkas XML, dan protokol yang digunakan tidak harus berupa HTTP. Seringkali XHR digunakan untuk meminta berkas berbentuk JSON. JSON, atau JavaScript Object Notation, adalah sebuah bentuk serialisasi data *key-value* yang merupakan subset dari spesifikasi ECMAScript atau JavaScript. JSON menyediakan bentuk serialisasi data yang sederhana, dapat dibaca manusia, dan dapat dengan mudah diproses dengan JavaScript. JSON berbasis teks dan pemrosesannya pun menggunakan pemrosesan teks di sisi JavaScript.

Pemanfaatan lain AJAX yang umum adalah menggunakan AJAX untuk melakukan *loading* sebagian laman web tanpa harus mengubah dan melakukan *loading* seluruh laman web. Pada pendekatan ini, hasil *response* berupa teks fragmen HTML yang tinggal di-*insert* ke dalam dokumen tanpa pemrosesan lebih lanjut.

AJAX juga dapat digunakan untuk data yang bersifat biner, sebagai sebuah Blob. Namun Blob tersebut harus dikonversi terlebih dahulu ke dalam notasi base64.

D. LocalStorage

LocalStorage adalah sebuah API HTML5 yang memungkinkan sebuah laman web menyimpan informasi secara persisten di komputer pengguna. Sebelum LocalStorage, satu-satunya cara untuk menyimpan data pada komputer pengguna adalah melalui *cookies* yang ukurannya sangat kecil dan umumnya hanya digunakan untuk menyimpan informasi seperti Session ID. Untuk mengakses *cookies* pun tidak ada API khusus dalam JavaScript.

Informasi pada LocalStorage disimpan sebagai *key-value pair* dan disimpan dalam kontainer atau wadah di penjelajah yang terpisah antara satu situs dan situs lainnya. Dengan demikian, informasi untuk sebuah key yang sama pada situs berbeda akan berbeda, dan sebuah situs tidak dapat mengakses informasi LocalStorage pada situs lainnya. LocalStorage telah diimplementasikan pada kebanyakan piranti penjelajah.

E. jQuery

jQuery adalah pustaka Javascript yang berguna untuk memodifikasi DOM pada lama HTML5 dengan membungkus API HTML5 menjadi sekumpulan fungsi yang dapat dimanfaatkan tanpa harus melakukan pengkodean API dasar dengan berulang. jQuery memanfaatkan selektor CSS untuk membuat pengaksesan DOM lebih mudah dan intuitif. jQuery juga mendukung AJAX dalam pustakanya.

Implementasi jQuery memanfaatkan pembungkusan sebuah objek ke dalam objek lainnya, dan begitu pun yang digunakan untuk fitur AJAX dalam jQuery. Sebuah objek jqXHR menangani pemanggilan metode-metode XHR biasa dengan nama-nama metode yang lebih singkat. Misalnya, *event handler* untuk AJAX dipanggil oleh objek jqXHR dengan parameter yang dimanipulasi khusus untuk menyertakan fitur-fitur misalnya *parsing* JSON secara otomatis. Tanpa jqXHR dan jQuery, pengembang harus menggunakan metode *parsing* JSON sendiri, sehingga jQuery membungkusnya dalam sebuah pustaka yang mudah digunakan.

F. HTTPS, SSL, dan TLS

SSL dan TLS adalah lapisan pengamanan jaringan yang berada pada transport layer dalam protokol TCP/IP. TLS merupakan pembaruan dari SSL. HTTPS adalah protokol HTTP yang dijalankan di atas SSL/TLS.

G. Kriptografi Simetris

Kriptografi adalah sebuah pendekatan dalam kriptografi di mana sebuah *plaintext* dienkripsi jadi *ciphertext* dengan sebuah kunci, dan kunci yang sama digunakan untuk mendekripsi *ciphertext* kembali menjadi *plaintext*.

Kriptografi simetris merupakan bentuk paling sederhana dari kriptografi dan telah digunakan sejak zaman kuno. Algoritma kriptografi klasik seperti Caesar's Cipher, Vigenere Cipher dan Playfair Cipher semuanya menggunakan kunci yang simetris.

Meski sederhana, kriptografi simetris juga digunakan untuk enkripsi yang membutuhkan standar keamanan tinggi. Hal tersebut diwujudkan dalam algoritma *block cipher* seperti Rijndael dan Twofish.

III. PEMBAHASAN

A. Gambaran Umum

Pada bab ini akan dipaparkan perancangan sebuah

metode untuk mengamankan transmisi AJAX dengan memanfaatkan kriptografi dalam JavaScript.

A. Kriptografi dalam JavaScript

Sebagai bahasa pemrograman yang sudah dewasa dan dengan sintaks yang menyerupai C, tidak sulit untuk mengimplementasikan algoritma kriptografi umum dalam JavaScript.

Salah satu pustaka kriptografi JavaScript yang populer adalah SJCL, singkatan dari Stanford JavaScript Crypto Library. SJCL menyediakan antarmuka untuk enkripsi dengan menggunakan AES. Hasil enkripsi berupa sebuah objek yang dapat diubah ke notasi JSON dan mengandung variabel-variabel seperti initialization vector, pilihan *cipher*, dan lain-lain.

Selain SJCL terdapat juga pustaka-pustaka seperti CryptoJS dan JavaScrypt. Secara umum, pustaka apapun dapat digunakan untuk menerapkan kriptografi dalam JavaScript, asal *cipher* dan algoritma yang sama didukung oleh piranti yang terdapat pada *server*.

B. Kriptografi dalam AJAX

Untuk mengamankan pesan yang akan dikirimkan ke server, maka pola AJAX dapat dimodifikasi. Modifikasi dilakukan agar pesan yang akan dikirimkan dienkripsi terlebih dahulu sebelum melakukan *request*, dan teks hasil *response* didekripsi terlebih dahulu sebelum diproses lebih lanjut oleh program.

Karena JavaScript dan XMLHttpRequest berbasis model *event*, maka pemrosesan tersebut dapat dilakukan berdasarkan *event* yang di-*trigger* oleh aplikasi. Akan tetapi, bentuk objek XMLHttpRequest sendiri merupakan objek *native* sehingga mekanisme kerja dan isi metodenya tidak dapat diubah. Agar enkripsi dan dekripsi dapat dilakukan, maka pendekatan lain yang dapat diambil adalah dengan mengubah pustaka yang melayani penggunaan AJAX. Salah satu pustaka yang paling populer adalah jQuery, sehingga pustaka tersebut akan digunakan pada makalah ini.

Pada pustaka jQuery, seluruh pengelolaan AJAX dilakukan melalui objek jqXHR. Objek jqXHR merupakan *superset* dari XMLHttpRequest, dan untuk memasang *handler* pada *event* AJAX yang dibungkus dalam jqXHR, digunakan metode-metode jqXHR. Metode tersebut diberi parameter oleh objek jqXHR, sehingga parameter tersebut dapat dimodifikasi.

Terdapat dua tahapan yang penting dalam AJAX, yang ditangani oleh jqXHR: *send* dan *load*.

Untuk bagian *send*, aplikasi di luar pustaka jQuery yang memanggil sebuah metode di jqXHR, yang umumnya berupa .send(), .get(), atau .post(). Metode tersebutlah yang kemudian meneruskan kepada objek XMLHttpRequest untuk dikirim sebagai *request* AJAX yang asli.

Untuk bagian *load*, *response* yang diterima dari server dapat diperoleh dengan mengakses properti responseText, responseJSON, atau responseXML. Ketiganya pada jQuery asli mengambil nilai dari objek XMLHttpRequest, namun dapat diganti sehingga diisi dengan hasil dekripsi terhadap *response* dari *server*.

Agar tidak perlu ada yang diubah dari sisi aplikasi pemanafaat jQuery, maka penyimpanan kunci dan variabel untuk proses enkripsi/dekripsi perlu disimpan di luar agar tidak mengubah pola pemanggilan metode jQuery yang sudah ada. Kunci tersebut dapat didapatkan dari pengguna melalui sebuah *prompt* di awal eksekusi aplikasi, kemudian disimpan dalam *LocalStorage*. Apabila tidak diperlukan penyimpanan persisten, maka dapat juga digunakan sebuah variabel global. Kunci tersebut tidak dikirimkan sama sekali melalui jaringan dan hanya disimpan di dalam komputer.

C. Kriptografi pada Server

Untuk melakukan transaksi pengiriman dan penerimaan data berbentuk *ciphertext*, maka algoritma dan struktur data yang digunakan untuk *ciphertext* pada pustaka AJAX dan pada server juga harus sama. Berbeda dengan pada sisi *client*, tidak ada satu *framework* yang umum dan *ubiquitous* digunakan oleh pengembang web sisi *server* untuk penanganan AJAX. Umumnya *response* hanya didapatkan dengan mengeluarkan data melalui STDOUT.

Dengan demikian, diperlukan usaha yang lebih pada sisi *server* untuk menangani kriptografi. Pendekatan lain yang dapat digunakan adalah apabila menggunakan server berbasis NodeJS, yang mengandalkan JavaScript sehingga pustaka kriptografi yang sama dapat digunakan oleh klien dan server.

Kunci yang digunakan untuk enkripsi dan dekripsi data pada server tidak terhubung dengan yang digunakan oleh enkripsi dan dekripsi pada *client* sehingga diperlukan mekanisme untuk mentransmisikan kunci tersebut secara aman.

D. Analisis Risiko dan Penanggulangan

Kelemahan utama pada pengembangan aplikasi yang telah dipaparkan adalah transmisi kunci. Meski kunci tidak ditransmisikan pada saat komunikasi normal, kunci tersebut perlu disimpan oleh *server* sehingga transmisi kunci pertama tersebut perlu dilakukan dengan cara yang berbeda. Apabila layanan HTTPS tersedia, maka kunci dapat dikirimkan melalui HTTPS, karena HTTPS sudah terjamin keamanannya.

Keamanan lainnya terdapat pada JavaScript. Secara intrinsik, bahasa JavaScript tidak optimal untuk penanganan kriptografi. Sebagai bahasa terinterpretasi, pengembang tidak memiliki tingkatan kendali yang sama dengan program *native* untuk urusan pengelolaan memori, sehingga memori yang digunakan untuk enkripsi mungkin dapat ditebus dan dibaca oleh program yang berbahaya. Analisis kemananan secara mendalam mengenai kriptografi telah dilakukan oleh Matasano Security dan disimpulkan bahwa disarankan untuk tidak memanfaatkan

kriptografi JavaScript untuk penanganan keamanan secara serius.

Akan tetapi, jika eksekusi dilakukan pada lingkungan yang ter-*sandbox* dengan baik, maka risiko keamanan dari JavaScript dapat dikurangi.

III. KESIMPULAN

Berdasarkan pemaparan dalam makalah ini, dapat disimpulkan sebagai berikut:

1. JavaScript dapat digunakan untuk mengamankan trasmisi AJAX dengan sederhana tanpa harus menggunakan SSL.
2. Pemanfaatkan kriptografi AJAX tersebut memiliki titik lemah untuk transmisi kunci awal serta penanganan memori.

REFERENCES

[1] http://developer.mozilla.org/
[2] http://www.w3.org/
[3] http://jquery.com/
[4] http://www.matasano.com/articles/javascript-cryptography/

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, 26 Maret 2013

Tubagus Andhika Nugraha
13510007