Pengamanan HTTP Request GET Menggunakan Kunci Asimetrik

Nicholas Rio (13510024)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Nicholasrio2811@gmail.com

***Abstract*—HTTP adalah suatu protocol yang paling umum digunakan dalam berbagai pemrograman internet. Namun, karena kebanyakan request dikirimkan dengan menggunakan plaintext, maka request ini menjadi tidak aman. Untuk mengamankan HTTP request, kita dapat membuat skema pengamanan dengan menggunakan kunci asimetrik. Setelah dilakukan implementasi dan analisis lebih lanjut, dapat disimpulkan bahwa meskipun skema yang telah dibuat oleh penulis telah meningkatkan kemananan dari HTTP request, namun masih terdapat berbagai pengembangan lebih lanjut yang dapat diimplementasikan ke depannya.**

***Index Terms*—HTTP, Kunci Asimetrik, Request, URI**

I. LATAR BELAKANG

Dalam dunia informatika sekarang ini, tak dapat dipungkiri bahwa pemrograman yang dilakukan pada jaringan telah menjadi sangat penting. Salah satu jenis protocol pemrograman jaringan internet yang sangat banyak digunakan adalah HTTP Protocol yang digunakana oleh nyaris semua situs.

Karena penggunaan HTTP Protocol ini, timbul beberapa permasalahan baru. Di antaranya, pesan-pesan yang digunakan dalam protocol ini umumnya tidak dienkripsi sehingga rentan untuk disadap ataupun disalah-gunakan. Enkripsi dengan menggunakan SSL sekalipun lebih berguna untuk keperluan autentikasi situs alih-alih berfokus pada penyamaran pesannya.

Oleh karena itu, penulis tertarik untuk membuat suatu penelitian mengenai pengamanan HTTP Protocol dengan menggunakan kunci asimetrik. Dengan penggunaan kunci ini, diharapkan bahwa protocol HTTP dapat diamankan dari tangan-tangan jahil yang berusaha menjaili situs ataupun pengguna situs.

II. HTTP PROTOCOL

HTTP Protocol adalah suatu protocol yang dibuat pada application layer yang digunakan untuk mendistribusikan, mengkolaborasikan ataupun sekadar mengirimkan data melewati internet. Protocol ini menjadi dasar dari pemrograman World Wide Web (WWW).

HTTP Protocol memiliki suatu konsep dasar request – response; Client akan melakukan request kepada server, dan kemudian server yang memiliki informasi yang diinginkan oleh client akan membalas request tersebut dengan sebuah response. Contoh dari server adalah sebuah web server, sedangkan contoh dari client adalah suatu web browser seperti Mozilla Firefox, Google Chrome, Internet Explorer, dsb.

Alamat dari HTTP resources sendiri direpresentasikan melalui suatu bentuk yang disebut URI (Uniform Resource Identifiers), atau secara lebih spesifik, URL (Uniform Resource Location). Protocol ini juga dibuat dengan asumsi bahwa terdapat transport layer yang reliable, dan biasanya menggunakan TCP connection. Ada pula beberapa kasus dimana UDP digunakan.

Selain itu, HTTP bersifat stateless, sehingga tidak ada state yang disimpan di antara client dan server. Oleh karena itu, pada setiap pengiriman request, state yang sebenarnya harus ikut dikirimkan juga (bila diperlukan).

Terdapat beberapa jenis HTTP Request, yaitu :

- GET
  Digunakan hanya untuk mendapatkan data dari web server.
- HEAD
  Digunakan hanya untuk mendapatkan data header dari web server, tanpa response body seperti pada request GET.
- POST
  Digunakan untuk meminta server menerima data apapun yang dikirimkan melalui request ini.
- PUT
  Digunakan untuk meminta server menyimpan suatu resource yang dikirimkan melalui request ini pada URI tertentu.
- DELETE
  Menghapus suatu resource tertentu.
- TRACE
  Meminta server mengirimkan kembali request terakhir untuk mengecek apabila terdapat perubahan data yang dilakukan oleh intermediate server / tidak.
- OPTIONS
  Meminta server memberitahu metode-metode request yang didukung oleh server tersebut.
- CONNECT
  Mengkonversi koneksi request menjadi TCP / IP

tunnel, biasa digunakan pada proxy yang tidak dienkripsi.

- PATCH
  Digunakan untuk melakukan modifikasi parsial pada resource tertentu.

Setiap server harus mengimplementasikan protocol GET dan HEAD agar dapat berjalan dengan baik.

Terdapat beberapa request method yang dianggap sebagai safe method, yaitu GET, HEAD, OPTIONS, dan TRACE. Maksud dari safe method di sini adalah bahwa request-request di atas hanya mendapatkan data saja, sehingga tidak akan mengubah data apapun yang ada di server. Berbeda dengan request-request seperti PUT, POST, ataupun DELETE yang dapat merubah data pada server sehingga menimbulkan efek samping yang tidak diinginkan.

Namun pada perkembangannya akhir-akhir ini, terdapat beberapa method yang dianggap tidak aman. Salah satunya adalah TRACE; hal ini disebabkan karena informasi yang didapatkan dari request ini dapat digunakan oleh pihak yang tidak bertanggung jawab untuk melakukan hal-hal yang tidak diinginkan seperti menembus keamanan, dsb. Oleh karena itu, pada protocol HTTP 1.1, request ini dianggap berbahaya.

Server pun memiliki format response yang disebut status codes. Beberapa status codes yang paling umum dijumpai di antaranya :

- OK 200
- Created 201
- Partial Information 203
- No Response 204
- Bad Request 400
- Unauthorized 401
- Payment Required 402
- Forbidden 403
- Page Not Found 404
- Server Internal Error 500
- Not Implemented 501

III. KUNCI ASIMETRIK

Kriptografi dengan kunci asimetrik merupakan suatu system yang menggunakan 2 kunci – public key dan private key. Sistem ini pertama kali ditemkuan oleh Diffie dan Hellman pada tahun 1976. Secara prinsip, cara kerjanya adalah sebagai demikian :

- Public key digunakan untuk mengenkripsi data yang hendak dikirimkan. Seperti namanya, public key dapat dimiliki oleh siapapun.
- Private key digunakan untuk mendekripsi data yang diterima. Private key hanya dimiliki oleh satu orang.

Pada umumnya, skema enkripsi yang digunakan bergantung pada sulitnya memecahkan suatu persamaan matematika tertentu, seperti persoalan matematika diskrit (algoritma ElGamal),

Penggunaan umum dari kunci asimetrik meliputi : pertukaran pesan biasa, pembuatan digital signature untuk mendapatkan prinsip authentication dan non-repudiation.

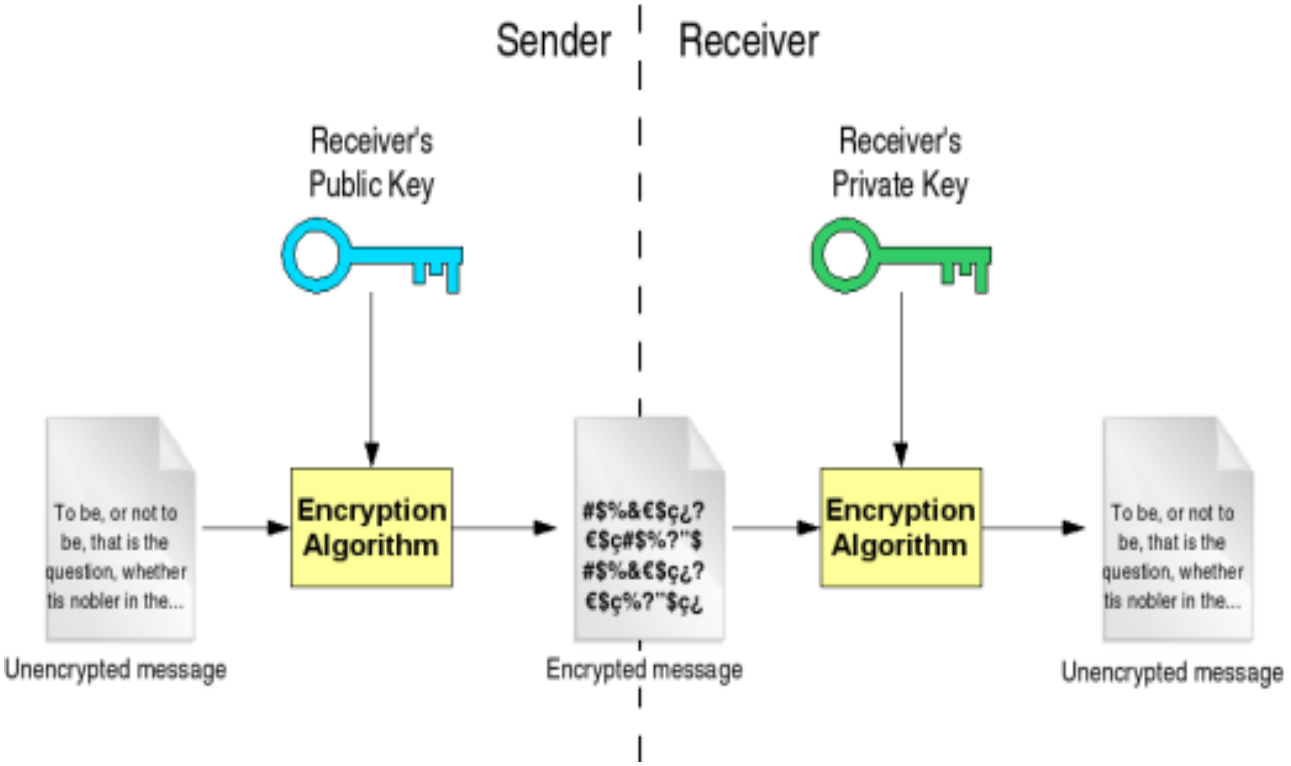


Figure 1 Skema Asimmetric Key

Keunggulan dari penggunaan kunci asimetrik, tentu saja, adalah bahwa selain sulit untuk dipecahkan, kriptanalis harus berusaha untuk memecahkan dua kunci sekaligus, yang jelas lebih sulit ketimbang memecahkan satu kunci saja. Selain itu, kedua belah pihak yang menggunakan skema ini tidak perlu saling bertukar kunci

privat, yang menyebabkan lebih sulit bagi pihak ketiga untuk memecahkan pesan yang dikirim diantara kedua belah pihak tersebut.

Namun, system ini tidak terlepas dari beberapa kelemahan tertentu. Di antaranya, skema ini masih dapat dipecahkan dengan metode Brute Force (meskipun memerlukan waktu yang sangat lama). Selain itu, waktu yang dibutuhkan untuk mengenkripsi ataupun mendekripsi pesan dengan menggunakan kunci asimetrik jauh lebih panjang ketimbang waktu yang dibutuhkan untuk mengenkripsi ataupun mendekripsi pesan yang menggunakan kunci simetrik. Hal ini karena kunci asimetrik umumnya memiliki panjang yang lebih besar dan pemrosesan yang lebih rumit dikarenakan oleh perhitungan matematis yang ada di belakangnya.

IV. Penyerangan Http Protocol Get

Pada umumnya, di browser, kita dapat membuat secara manual suatu GET request, contohnya dengan mengetikkan :

http://www.facebook.com/profile.php?id=10000329575496

URI di atas berarti : meminta data profile facebook dari user yang memiliki id 10000329575496.

Bagaimana percobaan serangan dapat dilakukan? Setelah kita mengetahui pola URI di atas, seorang attacker dapat saja mengubah-ngubah nilai id yang ada di belakangnya untuk mendapatkan data orang tertentu yang dia inginkan.

Secara sekilas, mungkin ini tidak terlihat berbahaya karena hanya menyasar profil www.facebook.com. Namun, bayangkan apabila yang diserang adalah suatu situs komersial yang secara kebetulan mengirimkan datanya menggunakan GET Protocol yang tidak dienkripsi. Selain itu, tidak hanya terbatas pada GET Protocol, hal ini juga dapat dimanfaatkan pada protocol-protocol lain seperti POST, di mana apabila atribut-atribut yang dikirimkan diketahui, attacker dapat dengan mudah mengubah data-data tersebut agar menjadi sesuai dengan keinginannya. Maka, terlihat sudah bahwa enkripsi protocol-protocol request tersebut menjadi penting dan krusial.

V. Skema Pengamanan HTTP Protocol

Pengamanan HTTP Protocol yang penulis buat menggunakan kunci asimetrik. Mengapa hal ini dilakukan? Pertimbangan-pertimbangan untuk menggunakan kunci asimetrik adalah sebagai berikut :

- Jika menggunakan kunci simetrik, kunci yang ada pada server = kunci yang ada pada setiap page / resource. Dengan kata lain, tidak dapat ditemukan cara yang aman untuk mengirimkan kunci tersebut karena dapat dengan mudah dilakukan intersepsi di tengah-tengah proses pendistribusian kunci.
- Hanya server yang memiliki wewenang untuk mendapatkan data dari HTTP Request.
- Meskipun penggunaan kunci asimetrik memerlukan computational cost yang besar, namun karena pada umumnya data yang dikirimkan pada protocol GET berukuran kecil, maka computational cost yang digunakan masih dapat dihandle dengan baik. Selain itu, data hasil enkripsinya pun masih tidak terlampau besar.

Konsep pengamanan HTTP Protocol Get dengan kunci asimetrik adalah sebagai berikut :

1. Server pusat autentikasi menyimpan private key
2. Server pusat mendaftarkan setiap page yang boleh membuat request ke dalam server tersebut di sebuah database
3. Seluruh page yang diizinkan membuat request ke dalam server diberikan sebuah public key

Secara implementasi, hal ini akan menjadi sedikit lebih rumit. Pertama-tama, struktur sebuah get request dalam URI adalah sebagai berikut :

id=12345&login=yes

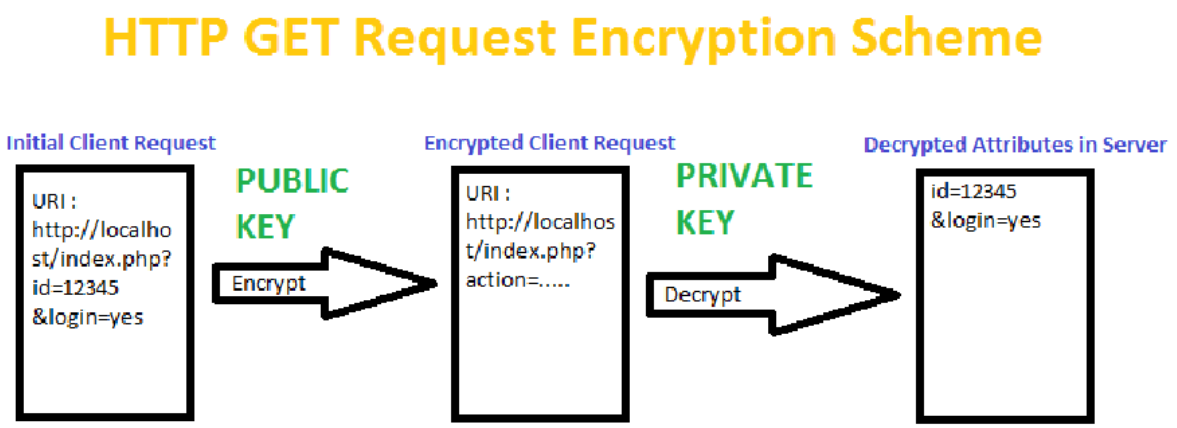
Yang berarti, bahwa dalam URI tersebut, terdapat data berupa id yang bernilai 12345 dan data login yang bernilai yes.

Dalam skema pengamanan yang penulis buat, penulis ingin melindungi 2 hal utama dari request ini, yaitu :

- Penggunaan nama atribut sebenarnya dalam GET request. Mengapa? Agar user tidak dapat mengetahui data apa yang sebenarnya dikirimkan dalam suatu GET request.
- Nilai dari suatu atribut.

Oleh karena itu, dalam penggunaan request selanjutnya, semua page yang terautentikasi hanya boleh menggunakan satu atribut yaitu action. Atribut ini yang nantinya akan mengandung nilai pesan yang sebenarnya dan telah dienkripsi menggunakan public key yang diberikan oleh server. Dengan cara ini, sangat sulit bagi user untuk mencoba-coba segala kombinasi kemungkinan yang ada karena hasil dari enkripsi akan sangat panjang; memerlukan computational cost yang sangat tinggi untuk mendapatkan kombinasi yang diinginkan.

Setelah itu, ketika pesan tersebut diterima oleh server, server akan mendekripsi pesan tersebut dengan private key yang dimilikinya, kemudian melakukan parsing terhadap hasilnya sehingga didapatkanlah atribut-atribut beserta nilainya yang diinginkan.


HTTP GET Request Encryption Scheme
Initial Client Request
URI :
http://localho
st/index.php?
id=12345
&login=yes
PUBLIC
KEY
Encrypt
Encrypted Client Request
URI :
http://localhos
t/index.php?
action=.....
PRIVATE
KEY
Decrypt
Decrypted Attributes in Server
id=12345
&login=yes

VI. Hasil Implementasi

Misalkan saja URI yang hendak kita tuju adalah :
http://localhost/index.php?id=12345&login=yes

Public key dan private key yang digunakan digenerate dengan menggunakan algoritma RSA 1024-bit. Private key yang terdapat pada server adalah :

mGHkpAkwhps7K6Udi9A8gBNZg4KSICkA4OLRr24sW1k7FF52xxSQZIhZDTm4RAiJ7rnXeswb4jbjcq9xx2ardZQ66HgcJLDgK98QonE19b64Cu5MSYZnGq6EM8Wpomyk8S6qeNWIBDIdEnWwYvmVN2AEfFgrjmpTUBAD3pwflhdCARfXs59eJV7fkN6lyVpBtKXvY9JL9YzQBoOpzhloW3nuyxxhG9K19Zf5ATZSxaxbdtXBu7zU6h0B74Ab9Y0D3nPvw4fe7ktxxTv6xbkMeBbr6xxhhxbEN7vpiJKkIu8ZWUuoJVzgxxtbkTxadTLxaVKSXrJLkgzsvI8Esi7vDQBwgCQH0Bxxf6PGlxbkFaqcsuRGMKVYhXLCcqfZ9qZtfZaNVPRcUWtxxCNG7sW6d42cTuXxaoVSl7TbMQ

Sedangkan public key yang terdapat pada page adalah :

SwXJdmPDAC2YB8C2nHxbxbfPjcLSXxxyRTC8jOHB50ZTVJXdeV3B9Q9rL9iNvjoxxMl6kW7DN6cTmLBODHeKzOvAMsrERizcXgmR5Wxajqjy7RST5EMALK6yE11SDqGDalQxajFj1ydt0QodCY3Yz0KPZfdGs2iffQKZxb9GnRxa5m0JxxBHnZ3MReFxaoJQB3Iu27FI6pBQlofJyFo5ocyXrOlEVItfPS3Fb9LIHs3vVfhxxpfCAE

Pesan yang dienkripsi (dari URI di atas), adalah sebagai berikut : id=12345&login=yes. Setelah dienkripsi, pesan tersebut menjadi :

C21uf9HRxxHwlnBBHt6b3trqRL7568DXwjrWka4n6knABkHHDuHRxbb7lUxx5JFIF8MdhUIepp67J5w5GBxbUEWF3BHtHeTkdqwmd3Sxag5LoGlF1w9htujVrwFrGo6lhMESd8LpvKWhJAy1f41MU7wpxaI7SGJIft0eVivHE4SHUbdSI

Maka, URI yang dikirimkan kepada server menjadi :

http://localhost/index.php?action=C21uf9HRxxHwlnBBHt6b3trqRL7568DXwjrWka4n6knABkHHDuHRxbb7lUxx5JFIF8MdhUIepp67J5w5GBxbUEWF3BHtHeTkdqwmd3Sxag5LoGlF1w9htujVrwFrGo6lhMESd8LpvKWhJAy1f41MU7wpxaI7SGJIft0eVivHE4SHUbdSI

Ketika pesan ini sampai di server, server mendekripsi kembali pesan di atas menjadi plaintext id=12345&login=yes. Proses yang dilakukan selanjutnya oleh server adalah :

- String split dengan regex("&")
- String split dengan regex ("=") untuk masing-masing hasil split di atas
- Nilai pertama adalah nama resource, sedangkan nilai kedua hasil split adalah nilai dari atribut tersebut.

Penulis juga melakukan percobaan ketika penulis mengubah sedikit data hasil enkripsi di atas. Hasilnya, pesan yang didapatkan ketika dekripsi menjadi error dan server mengeluarkan pesan bahwa request failed.

Pembangkitan kunci dan pengenkripsian pesan memakan waktu 207 ms, yang menunjukkan bahwa computational cost dari proses enkripsi ini masih dapat ditoleransi.

VII. Kemungkinan Pengembangan

Penulis baru mengimplementasikan skema ini pada HTTP Request GET. Namun, implementasi dari skema pengamanan ini sesungguhnya dapat pula dilakukan pada request-request lain seperti POST dsb. Sebagai contoh, pada request POST, dapat dilakukan langkah-langkah berikut :

1. Menyiapkan data yang hendak di-POST
2. Mengambil setiap atribut dan nilainya, kemudian mengenkripsinya
3. Mengirimkannya hanya lewat satu atribut action.

Namun, cara ini juga masih memiliki kelemahan. Kelemahan-kelemahan yang masih dimiliki oleh cara ini adalah diantaranya : tidak dapat menerapkan prinsip authentication secara sepenuhnya. Sebagai contoh, apabila URI yang terbentuk setelah enkripsi tadi digunakan kembali secara berulang-ulang, kita tidak dapat memastikan bahwa data tersebut dikirim oleh pengguna yang sah.

Untuk menanggulangi kelemahan di atas, dapat digunakan suatu skema diantaranya :

- Penggunaan timestamp pada masing-masing request, yang diimplementasi bersamaan dengan logging request pada server
- Penggunaan URI page requester yang ikut dienkripsi pada URI di atas.

Namun, hasil testing awal pada skema baru ini telah menunjukkan suatu hasil yang menjanjikan.

VIII. Kesimpulan

Kesimpulan yang dapat ditarik dari penelitian ini adalah sebagai berikut :

- HTTP Request dapat diamankan dengan menggunakan skema kunci asimetrik.
- Computational cost dari penggunaan kunci asimetrik masih dapat ditoleransi.
- Masih terdapat kelemahan pada implementasi penulis yang dapat dikembangkan lebih jauh.

References

[1] http://www.w3.org/Protocols/HTTP/HTRESP.html
[2] http://www.w3.org/Protocols/rfc2616/rfc2616.html
[3] http://www.cs.cornell.edu/courses/CS5430/2013sp/TL04.asymmetric.html

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, 20 Mei 2013

Nicholas Rio (13510024)