One-Time Password Berbasis Waktu dan Algoritma RSA sebagai Metode Autentikasi

Adin Baskoro Pratomo - 13513058
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
adinbaskoro.p@gmail.com

***Abstrak*—Makalah ini membahas metode baru untuk autentikasi pada saluran yang tidak aman, dengan menggabungkan *one-time password* dan algoritma kriptografi kunci publik RSA. Metode ini memungkinkan server untuk mengenali pengguna yang masuk hanya dengan *password yang* dikirimkan, namun tidak rentan terhadap *replay* attack.**

Kata kunci—one-time password; kunci publik; RSA

I. PENDAHULUAN

Penggunaan *one-time password* (OTP) sudah cukup umum saat ini. OTP adalah *password* yang hanya dapat digunakan untuk satu sesi atau sekali transaksi. Metode yang digunakan untuk membangkitkan OTP dapat berupa algoritma yang menggunakan *password* yang telah dibangkitkan sebelumnya, atau dengan menggunakan waktu. Apabila menggunakan waktu, *password* yang dibangkitkan hanya akan dapat digunakan untuk jangka waktu tertentu saja. Biasanya, OTP digunakan sebagai metode autentikasi pengguna melalui HTTP, dengan menggunakan *Authorization header*. Dengan metode ini, pengguna perlu memasukkan username, serta OTP yang sudah dibangkitkan. OTP sering digunakan untuk menghindari *replay attack*, yaitu jenis serangan ketika suatu transaksi yang valid diulangi dengan niat jahat.

Algoritma kriptografi kunci publik saat ini sering digunakan sebagai metode autentikasi, salah satu contohnya adalah autentikasi pada SSH (*Secure Shell)*. Cara kerjanya secara umum yaitu, *client* membangkitkan pasangan kunci, kemudian mengirimkan kunci publiknya ke *server*. Ketika *client* akan terhubung ke *server*, *client* akan membangkitkan sebuah *signature* dengan kunci privat yang dimiliki. Kemudian *server* akan melakukan validasi terhadap signature tersebut dengan kunci publik yang telah diberikan oleh *client.*Salah satu algoritma kriptografi kunci publik yang sering digunakan yaitu RSA.

Makalah ini akan berfokus pada penggabungan kedua metode tersebut, untuk membuat sebuah metode baru untuk autentikasi secara aman pada saluran yang tidak aman, tanpa perlu ada *identifier* seperti username. Pertama, *client* membangkitkan pasangan kunci publik, kemudian mengirimkan kunci publiknya ke server. Kemudian, OTP dibangkitkan berdasarkan *timestamp* saat pembangkitan dimulai. OTP berupa hasil hash dari *timestamp* tersebut. Kemudian, OTP tersebut dijadikan *signature* dengan menggunakan algoritma RSA. *Signature* tersebut kemudian dikirimkan ke *server* untuk divalidasi. Dari *signature* tersebut, *server* dapat memastikan yang terhubung adalah *client* yang berhak, sekaligus menghindari *replay attack*, karena OTP yang dibangkitkan selalu berubah berdasarkan waktu.

II. DASAR TEORI

Pada bagian ini dibahas beberapa teori yang menjadi dasar untuk implementasi OTP berbasis waktu dan RSA untuk autentikasi.

A. Replay Attack

Salah satu jenis serangan yang lazim ditemui pada jaringan komputer adalah *man in the middle attack* (MITM). Salah satu serangan yang dilakukan dengan metode MITM adalah pencurian identitas autentikasi seperti ID dan kata sandi pengguna. MITM biasanya dilakukan pada saluran yang tidak terenkripsi, karena paket data yang dikirimkan dapat langsung dipahami.

Ketika penyerang sudah berhasil mencuri identitas pengguna, penyerang dapat menggunakan ulang (*replay*) identitas tersebut untuk masuk ke sebuah sistem. Hal ini membuat penyerang hanya perlu mencuri identitas sekali saja untuk mendapat hak akses. Jenis serangan ini disebut *replay attack*.

B. One-time Password

One-time password (OTP) dirancang untuk mengatasi *replay attack*. Dengan OTP, identitas pengguna yang dikirimkan melalui jaringan komputer akan terus menerus berubah, dan hanya dapat digunakan satu kali saja. Dengan demikian, meskipun penyerang dapat mencuri identitas pengguna, penyerang tersebut tidak akan dapat menggunakan identitas yang dicuri.

Sistem yang memanfaatkan *One-time password* harus memiliki pembangkit yang mampu membangkitkan *password* berdasarkan masukan pengguna. Selain itu, *server* harus dapat melakukan validasi terhadap *password* yang diterima. *Server*

juga harus mampu memberikan *seed* yang ikut digunakan untuk membangkitkan *password*.

Pembangkitan OTP dapat dilakukan dengan berbagai cara, namun yang paling umum adalah dengan melakukan *hash* pada identitas pengguna dan seed dari server. Salah satu metode yang memanfaatkan fungsi *hash* adalah metode yang diajukan oleh Leslie Lamport. Metode ini bekerja dengan cara melakukan hash pada identitas pengguna dan seed sebanyak n iterasi untuk menghasilkan *password*. kemudian, ketika *password* tersebut digunakan, *password* tersebut disimpan di *server*, kemudian jumlah iterasi dikurangi 1. Berikutnya, ketika pengguna ingin melakukan autentikasi kembali, jumlah iterasi *hash* yang dilakukan adalah n-1. Untuk melakukan verifikasi, sever cukup menghitung *hash* dari *password*, dan membandingkannya dengan nilai yang sudah tersimpan.

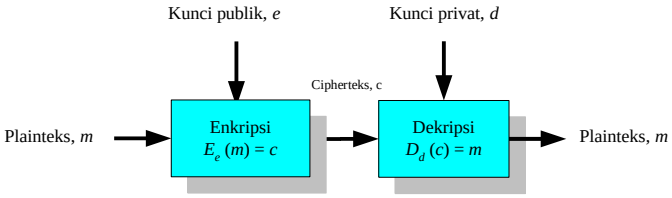
C. Algoritma Kriptografi Kunci Publik

Algoritma kriptografi kunci publik adalah salah satu bentuk kriptografi kunci nirsimetri. Kelebihan yang dimiliki oleh algoritma kriptografi kunci publik yaitu kunci yang dgunakan untuk enkripsi dapat disebarluaskan tanpa membuat kunci untuk dekripsi diketahui oleh publik. Hal ini memberikan beberapa dampak, yaitu:

1) Penyampaian kunci dapat melalui saluran yang aman. Hal ini karena kunci yang diperlukan untuk enkripsi hanya kunci yang bersifat publik, sehingga dapat disebarluaskan tanpa takut pesan rahasia yang terenkripsi dapat didekripsi oleh orang lain.

2) Sebuah pesan bisa ditandatangani dengan menggunakan kunci yang dimiliki secara privat. Tanda tangan tersebut dapat diverifikasi dengan menggunakan kunci yang dapat disebar secara publik. Dengan demikian, tanda tangan yang diberikan tidak dapat dipalsukan, dan penandatangan tidak dapat berbohong mengenai tanda tangan tersebut.

Secara umum, skema algoritma kriptografi kunci publik dapat dilihat pada diagram berikut:



Gambar 1. Skema Algoritma Kriptografi Kunci Publik

Pada diagram tersebut, dapat terlihat bahwa enkripsi dilakukan dengan kunci publik. Masukan berupa plainteks dienkripsi menghasilkan cipherteks. Untuk mendekripsi cipherteks tersebut, kunci yang digunakan adalah kunci privat. Hasil dekripsi adalah plainteks semula.

Salah satu algoritma kriptografi kunci publik yang umum digunakan saat ini adalah algoritma RSA. Algoritma ini sering digunakan karena sangat aman, meskipun waktu yang dibutuhkan lebih banyak.

D. Algoritma RSA

Algoritma RSA merupakan salah satu algoritma kriptografi kunci publik yang paling umum digunakan saat ini. Algoritma ini ditemukan oleh tiga peneliti MIT pada tahun 1976, yaitu Ron Rivest, Adi Shamir, dan Len Adleman. RSA mengandalkan sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima.

Algoritma RSA memiliki properti utama sebagai berikut:

1) Bilangan prima p dan q (rahasia)
2) $n = p \,.\, q$ (tidak rahasia)
3) $\phi(n) = (p-1)(q-1)$ (rahasia)
4) Kunci enkripsi e (tidak rahasia)
   Syarat: PBB(e, $\phi(n)$) = 1
5) Kunci dekripsi d (rahasia)
   d dihitung dari $d \equiv e^{-1} \bmod (\phi(n))$
6) Plainteks m (rahasia)
7) Cipherteks c (tidak rahasia)

Untuk membangkitkan pasangan kunci, diperlukan dua bilangan prima p dan q. kemudian, hitung nilai n dengan persamaan berikut:

$$n = p \,.\, q.$$

Hitung $\phi(n)$ dengan persamaan

$$\phi(n) = (p-1)(q-1).$$

Untuk menentukan kunci publik, pilih sebuah bilangan bulat e, dengan e relatif prima terhadap $\phi(n)$. Kunci privat dapat ditentukan dengan persamaan

$$ed \equiv 1 \ (\mathbf{mod}\ \phi(n)) \text{ atau } d \equiv e^{-1} \bmod (\phi(n))$$

Hasil dari perhitungan-perhitungan diatas adalah pasangan (e, n) yang merupakan kunci publik dan (d,n) yang merupakan kunci privat.

Proses enkripsi dengan algoritma RSA terdiri dari beberapa tahap:

1) Nyatakan pesan menjadi blok plainteks m_i, dengan syarat $0 < m_i < n-1$.
2) Hitung blok cipherteks c_i untuk blok plainteks m_i dengan persamaan $c_i = m_i^e \ \mathbf{mod}\ n$.

Dekripsi pada cipherteks dapat dilakukan dengan persamaan

$$m_i = c_i^d \ \mathbf{mod}\ n$$

Kekuatan utama algoritma RSA terletak pada sulitnya melakukan pemfaktoran pada bilangan menjadi faktor-faktor prima. Perlu waktu yang sangat lama untuk melakukan hal tersebut, bahkan dengan *supercomputer* yang ada saat ini.

Apabila faktor prima dari suatu bilangan sudah bisa didapat dengan cepat, maka algoritma ini akan dapat ditembus dengan mudah.

Kelemahan dari RSA adalah waktu yang dibutuhkan untuk melakukan enkripsi lebih lama dibanding algoritma kunci simetri seperti AES dan DES. Oleh karena itu, pada umumnya algoritma RSA tidak digunakan untuk mengenkripsi pesan secara langsung, melainkan digunakan untuk mengenkripsi kunci algoritma enkripsi simetri. Kunci algoritma enkripsi simetri tersebutlah yang digunakan untuk mengenkripsi pesan.

III. Rancangan Sistem

Pada bagian ini dijelaskan rancangan dari sistem yang akan dibuat. Rancangan meliputi algoritma autentikasi, serta rancangan perangkat lunak yang akan digunakan untuk pengujian.

A. Rancangan Algoritma Autentikasi

Algoritma autentikasi yang akan dibuat terdiri dari dua bagian utama, yaitu sistem pembangkitan *one-time password*, dan sistem penandatanganan dengan kunci publik menggunakan algoritma RSA. *One-time password* digunakan untuk menghindari *replay attack*, sedangkan algoritma kriptografi kunci publik digunakan sebagai sarana autentikasi. Kedua elemen tersebut memungkinkan autentikasi yang sederhana dan aman.

Pembangkitan *one-time password* dilakukan berdasarkan waktu, dengan menghitung selang waktu saat ini (T_i) dengan waktu *epoch* yang telah disetujui sebelumnya (T_0). Selang waktu tersebut kemudian dibagi dengan sebuah selang waktu lain, yang menandakan waktu berlaku dari *password*. Nilai tersebut lalu digabung dengan *seed* yang sudah ditentukan. Hasil penggabungan kemudian di-*hash* untuk mendapatkan *one-time password.*

Agar proses autentikasi memungkinkan, perlu ada sistem untuk mengenali pengguna. Hal ini dicapai dengan algoritma RSA sebagai algoritma kriptografi kunci publik. *One-time password* yang sudah dibangkitkan dibuat tanda tangan digitalnya dengan menggunakan kunci yang didapat dengan algoritma RSA. Tanda tangan digital tersebut merupakan paket yang dapat digunakan sebagai metode autentikasi.

Cara sebuah sistem melakukan autentikasi pengguna berdasarkan paket yang diterima yaitu dengan mendekripsi paket dengan kunci publik yang sesuai dari paket yang diterima. Untuk mengetahui kunci publik yang sesuai, perlu ditambahkan *signature* seperti *username* saat autentikasi. Setelah didekripsi, sistem membangkitkan *one-time password* dengan metode yang sama dengan sebelumnya. *One-time password* tersebut kemudian di-*hash*. Kemudian, tanda tangan yang telah diterima digunakan sebagai alat verifikasi dari hasil *hash* tersebut. Apabila cocok, autentikasi berhasil. Sebaliknya apabila gagal, autentikasi gagal.

B. Rancangan Server

Server yang akan dibuat untuk pengujian metode ini hanya berupa server sederhana, yang akan melakukan *listen* pada *port* tertentu. Komunikasi akan dilakukan dengan protokol sederhana. Jika server menerima *string*

```
auth:<username>:<public_key>
```

paket berikutnya yang diterima akan dianggap sebagai paket autentikasi, dan akan divalidasi.

C. Rancangan Client

Aplikasi *client* cukup sederhana, hanya akan melakukan koneksi *socket* ke *server*, kemudian mengirim *username* dan *one-time password* yang sudah ditandatangani dengan kunci privat. Proses pengiriman mengikuti protokol sederhana yang sudah dijelaskan pada bagian rancangan *server*.

IV. Implementasi Sistem

Penulis mengimplementasikan sistem dengan menggunakan bahasa Python 3, serta *library* Crypto dan hashlib yang sudah tersedia. Pembangkitan *one-time password* dilakukan dengan menentukan waktu *epoch*, kemudian mengambil waktu sistem. Selisih waktu tersebut kemudian dibagi dengan interval waktu yang ditentukan. Kemudian hasil pembagian digabung dengan sebuah nilai *seed*. Nilai seed yang digunakan sudah ditentukan sebelumnya. Hasil penggabungan kemudian di-*hash* dengan algoritma SHA-256 yang terdapat pada *library* hashlib.

Pasangan kunci privat dan kunci publik dibangkitkan dengan menggunakan library Crypto, dengan implementasi algoritma RSA yang tersedia. Panjang kunci yang digunakan adalah 4096 bit. Hasil hash lalu dibuat tanda tangan digitalnya untuk dikirim. Server kemudian dapat melakukan verifikasi tanda tangan digital berdasarkan perhitungan hasil hash OTP yang dilakukan di server.

Koneksi antara *server* dan *client* dilakukan dengan menggunakan *socket.* Alamat dan *port* yang digunakan dijadikan parameter dari aplikasi. *Server* akan melakukan *listen* pada *port* yang telah dimasukkan

Ketika server mendapat permintaan untuk autentikasi, server akan melakukan autentikasi terhadap paket yang diterima. Apabila autentikasi valid, *server* akan mengirimkan respon kepada *client* bahwa autentikasi berhasil. Begitu pula sebaliknya, apabila autentikasi gagal *server* akan memberikan respon bahwa autentikasi gagal kepada *client.*

V. Pengujian

Hasil pengujian autentikasi *client* kepada *server* adalah sebagai berikut:

Gambar 2. Tampilan *client*

Saat pengujian pada *client*, terlihat *one-time password* yang dibangkitkan adalah **49403919KRIPTO**. *Password* ini dihasilkkan dari pembagian waktu ketika program dijalankan dengan interval yang sudah ditentukan, yang pada pengujian ini diatur menjadi 30 detik. Hasil pembagian tersebut kemudian ditempel dengan *seed* yang juga sudah ditentukan sebelumnya. *Seed* yang digunakan pada pengujian ini adalah KRIPTO.

Username, *hash* OTP yang sudah ditandatangani, dan kunci publik kemudian dikirim ke *server*. Dari sisi server, tampilannya adalah sebagai berikut:

Gambar 3. Tampilan *server*

Pada sisi *server*, OTP juga akan dibangkitkan. Dari hasil pengujian, terlihat bahwa OTP yang dibangkitkan sama. Hal ini karena selisih waktu antara pembangkitan OTP pada *client* dan pembangkitan OTP pada *server* masih dibawah interval yang ditentukan. Karena OTPnya sama, maka hasil *hash*nya pun sama.

Setelah menghitung *hash*, tanda tangan digital yang sudah diterima kemudian diverifikasi, berdasarkan hasil perhitungan *hash* yang dilakukan pada *server*. Apabila tanda tangan valid, maka autentikasi diterima.

Perhitungan OTP akan selalu berubah seiring waktu berjalan. Hal ini terlihat ketika menjalankan *client* ulang, setelah beberapa lama. Hasil eksekusi ulang *client* adalah sebagai berikut:

Gambar 4. Hasil eksekusi ulang *client*

Pada hasil eksekusi tersebut, dapat dilihat bahwa OTP yang dibangkitkan berbeda. Sebelumnya OTP yang dibangkitkan adalah **49403919KRIPTO**. Setelah eksekusi ulang, nilai OTP yang dibangkitkan adalah **49403926KRIPTO**. Hal ini membuat identitas lama yang dicuri menjadi tidak valid, meskipun ditandatangani dengan kunci privat yang sama.

VI. Kesimpulan Dan Saran

Penggabungan *one-time password* dan algoritma kriptografi kunci publik memungkinkan autentikasi yang lebih praktis karena pengguna cukup memasukkan *username*. Pasangan kunci milik pengguna akan dibangkitkan secara otomatis, dan langsung dapat digunakan sebagai metode autentikasi yang cukup aman. Dengan demikian, meskipun terjadi serangan *man in the middle attack*, identitas yang dicuri tidak akan dapat digunakan ulang.

Metode ini masih dapat dikembangkan lebih lanjut, dengan lebih memperbaiki struktur paket yang dikirimkan, agar penggunaan *bandwidth* jaringan lebih rendah. Selain itu, dapat pula diberikan mekanisme untuk menonaktifkan *password* apabila telah digunakan lebih dari beberapa kali, untuk menambah keamanan.

Referensi

[1] N. Haller dkk, “A One-Time Password System”, RFC 2289, Februari 1998. [Online]. Diakses pada: https://tools.ietf.org/rfc/rfc2289.txt

[2] Leslie Lamport, "Password Authentication with Insecure Communication", Communications of the ACM 24.11 (November 1981), 770-772

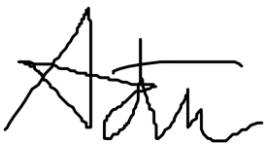
[3] R. L. Rivest , A. Shamir , L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, v.21 n.2, p.120-126, Feb. 1978 [doi>10.1145/359340.359342]

[4] Munir, Rinaldi. 2016. Bahan Kuliah Algoritma RSA

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, 19 Desember 2016



Adin Baskoro Pratomo 13513058