Implementasi Tanda Tangan Digital dengan Menggunakan *Elliptic Curve* untuk Validasi *One-Time Password*

Anwar Ramadha
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
anwar.ramadha@gmail.com

***Abstrak*—Saat ini, keamanan yang menggunakan *one-time password* (OTP) menjadi sangat penting. OTP digunakan untuk melawan *replay attack* atau *eavesdropping*. Ada banyak cara untuk mengotentikasi OTP, salah satunya menggnuakan algoritma kunci publik. RSA adalah salah satu algoritma kunci publik yang populer. Namun terdapat beberapa isu yang menjadi sorotan. Penggunaan *Elliptic Curve Cryptography* (ECC) adalah salah satu solusi yang dapat menutupi kekurangan algoritma RSA. Makalah ini bertujuan untuk mengimplementasikan algoritma ECC yang digunakan untuk otentikasi OTP. Jenis ECC yang digunakan adalah *Ellitptic Curve Digital Signature Algorithm* (ECDSA).**

***Kata kunci*—One-Time Password, Elliptic Curve Cryptography, Elliptic Curve Digital Signature Algorithm.**

I. LATAR BELAKANG

Kita hidup di zaman digital yang banyak menggunakan komputer. Jaringan komputer menyediakan *platform* untuk mengerjakan berbagai macam hal seperti *e-commerce, online banking,* membagikan informasi, dan sebagainya. Untuk itu dibutuhkan suatu system keamanan yang dapat melindungi data pengguna.

Penggunaan *one-time password* (OTP) sebagai media otentikasi sudah umum saat ini. OTP adalah kata sandi yang hanya dapat digunakan untuk satu sesi atau satu kali transaksi. Algoritma yang digunakan untuk membangkitkan OTP biasanya menggunakan *pseudorandomn* atau *fully random*.

Saat ini, algoritma kunci publik sering digunakan sebagai metode otentukasi, contohnya otentikasi pada *Secure Shell* (SSH). Secara umum, cara kerja SSH adalah, *client* membangkitkan sepasang kunci (kunci publik dan kunci privat), kemudian *client* mengirim kunci publiknya ke server. Ketika *client* dan *server* terhubung, *client* akan membangkitkan tanda tangan digital dengan kunci privat Kemudian *server* akan melakukan validasi dengan terhadap tanda tangan tersebut dengan menggunakan kunci publik *client.* Contoh algoritma kunci public adalah RSA dan ECC.

Selain itu, OTP juga biasa digunakan pada aplikasi jual beli daring. Contoh penggunaannya adalah ketika pengguna ingin melakukan pengisian pulsa pada aplikasi tersebut. Sebelum pembayaran diproses, maka aplikasi tersebut akan meminta kode ke *server*. Setelah kode OTP di bangun, maka kode tersebut akan di *hash* dengan menggunakan kunci yang telah di bangun dengan menggunakan algoritma ECC untuk mendapatkan *signature*. Kemudian hasil enkripsinya dimasukkan kedalam *session* pengguna yang meminta. Sesi ini akan berakhir dengan waktu yang telah ditemtukan, misalnya tiga atau lima menit.

II. DASAR TEORI

A. One-time Password

One-time Password (OTP) dirancang untuk mengatasi *replay attack.* Identitas pengguna yang dikirimkan melalui jaringan akan terus-menerus berubah dengan menggunakan OTP. Dengan demikian, walaupun seorang berhasil mencuri identitas pengguna, penyerang tersebut tidak akan dapat menggunakan identitas yang dicuri.

Sistem yang memanfaatkan OTP harus memiliki pembangkit yang dapat memilih *password* sesuai masukan pengguna. Selain itu, sistem harus dapat melakukan validasi terhadap *password* yang diterima. Untuk membangkitkan *password*, sistem harus menggunakan *seed* yang disediakan juga oleh sistem yang dibangun.

Pembangkitan OTP dapat dilakukan dengan berbagai metode. Cara yang paling umum digunakan adalah melakukan *hash* pada indentitas pengguna dan *seed* dari *server*. Salah satu metode yang digunakan 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 *password* tersebut disimpan *server,* kemudian jumlah iterasi dikurangi satu. Untuk melakukan otentikasi, *server* cukup menghitung *hash* dari *password* dengan iterasi n-1 dan membandingkannya dengan nilai yang tersimpan pada *server.*

B. Replay Attack

Serangan yang umum ditemukan pada jaringan komputer adalah *man in the middle attack* (MITM). Salah satu jenis serangan yang dilakukan dengan metode MITM adalah

pencurian identitas untuk otentikasi seperti ID dan kata sandi. MITM biasa dilakukan pada saluran yang tidak terenkripsi, karena data yang dikirim dapat langsung dibaca dan dipahami.

Ketika penyerang berhasil mencuri identitas, penyerang dapat menggunakan ulang identitas tersebut untuk masuk ke sebuah sistem. Hal ini dapat membuat penyerang hanya perlu mencuri identitas satu kali untuk mendapatkan hak akses.

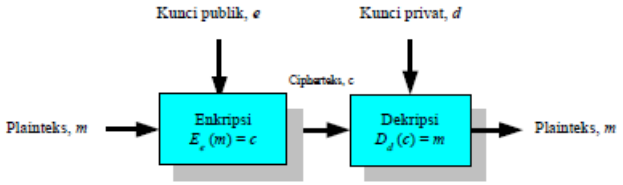
C. *Algoritma Kunci Publik*

Algoritma kriptografi kunci publik adalah salah satu bentuk kriptografi kunci asimetris. Kelebihan yang dimilki oleh algoritma jenis ini adalah kunci yang digunakan untuk enkripsi dapat sebarkan tanpa diketahui oleh umum. Hal ini memberi beberapa dampak, antara lain:

a. Penyampaian kunci dapat melalui saluran yang aman. Hal ini karena kunci yang diperlukan untuk enkripsi hanya kunci yang bersifat publik. Sehingga dapat disebarkan tanpa khawatir pesan yang disampaikan dapat didekripsi orang lain.

b. Sebuah pesan bias 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 menghindar dari tanda tangan tersebut.

Secara umum, alur algoritma kriptografi kunci publik dapat dilihat pada Gambar 1.



Gambar 1 Alur Algoritma Kunci Publik

Pada Gambar 1, enkripsi plainteks menggunakan kunci publik menjadi cipherteks. Sedangkan untuk menderipsi cipherteks tersebut, kunci yang digunakan adalah kunci privat.

Salah satu algoritma kriptografi kunci publik adalah RSA. Algoritma ini sering digunakan karena aman, meskipun waktu yang dibutuhkan lebih banyak. Namun algortima ini mempunyai masalah pada penyimpanan kunci yang cukup besar. Oleh karena itu, diusulkan algoritma *Elliptic Curve Cryptography* untuk menutup kekurangan tersebut.

D. *Elliptic Curve Cryptography*

Sebagian besar algortima kriptografi kunci publik seperti RSA, ElGamal, dan Diffle-Hellman menggunakan *integer* dengan bilangan yang sangat besar. Sehingga memiliki masalah pada penyimpanan dan pemrosesan kunci dan pesan Untuk mengatasi masalah tersebut, maka alternatif yang dapat digunakan adalah menggunakan kurva eliptik (*Elliptic Curve*). Komputasi dengan menggunakan kurva eliptik menawarkan keamanan yang sama dengan ukuran kunci yang lebih kecil. Kriptografi yang menggunakan kurva eliptik disebut *Elliptic Curve Cryptography* (ECC).

ECC adalah algoritma kunci publik yang lebih baru namun belum dianalisis dengan baik. Algoritma ini dikembangkan oleh Neal Koblitz dan Victor S. Miller pada tahun 1985. ECC diklaim dapat memberikan keamanan yang sama dibanding RSA dengan panjang kunci yang lebih pendek. Sebagai contoh, kunci sepanjang 160-bit pada ECC sama dengan 1024-bit pada RSA. Hal ini menyebabkan penggunaan memori dan komputasi lebih sedikit dibanding RSA. Algortima ini cocok untuk diterapkan pada perangkat nirkabel yang memiliki prosesor, memori, dan umur baterai yang terbatas.

ECC adalah sistem kriptografi kunci publik yang sejenis dengan RSA, Rabin, ElGamal, D-H, dll. Setiap pengguna memiliki kunci publik dan kunci privat. Beberapa sistem kritografi kunci publik yang memanfaatkan kurva eliptik adalah sebagai berikut.

a. *Elliptic Curve ElGamal* (ECEG)
b. *Elliptic Curve Digital Signature* (ECDSA)
c. *Elliptic Curve Diffie-Hellman* (ECDH)

Inti dari sistem kriptografi kunci publik yang menggunakan kurva eliptik adalah grup eliptik (himpunan titik pada kurva eliptik dan sebuah operasi biner +). Pada RSA, operasi dasar yang digunakan adalah perpangkatan, sedangkan pada ECC memiliki operasi perkalian titik (penjumlahan berulang dua buah titik).

Dua pihak yang berkomunikasi menyepakati paramerter data sebagai berikut:

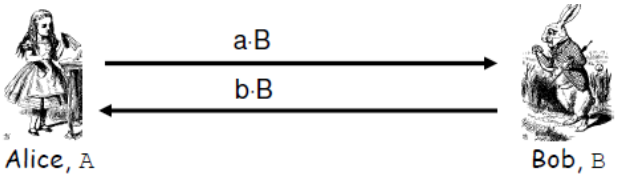
a. Persamaan kurva eliptik $y^2 = x^3 + ax + b \bmod p$, dengan p adalah bilangan prima.
b. Grup eliptik yang dihitung dari persamaan kurva eliptik.
c. Titik basis $B(X_B, Y_B)$ yang dipilih dari grup eliptik untuk operasi kriptografi.

Setiap pihak membangkitkan pasanagan kunci publik dan kunci privat. Kunci privat x dibangun dengan membangkitkan bilangan acak dari selang [1, p – 1]. Sedangkan kunci publik Q dibangkitkan antara hasil kali antara kunci privat dan titik basis B, sehingga Q = x . B.

E. *Elliptic Curve Cryptography Diffie-Hellman*

Algoritma ini adalah salah satu variasi dari ECC. *Elliptic Curve Cryptography Diffie-Hellman* (ECDH) mempunyai tiga buah kunci yang harus dibangkitkan yaitu kunci publik, kunci privat, dan kunci rahasia. Kunci publik didapatkan dari kurva eliptik dan titik B(x, y) pada kurva. Kunci privat dibangkitkan

dengan cara menentukan nilai acak dari rentang [1, p-1]. Sedangkan kunci rahasia dihitung dari perkalian kunci publik dank unci privat. Skema pertukaran atau pehitungan kunci rahasia dapat dilihat pada Gambar 2.



Gambar 2 Skema pertukaran kunci pada ECDH

Algoritma Diffie-Hellman memiliki langkah-langkah sebagai berikut:

a. Alice dan Bob menghitung kunci publik dank unci privat masing-masing.
   Alice :
   Kunci privat = a
   Kunci publik = P_A = a.B
   Bob :
   Kunci privat = b
   Kunci publik = P_B = b.B

b. Alice dan Bob saling mengirim kunci publik masing-masing.

c. Keduanya melakukan perkalian kunci privat dengan kunci publik mitranyauntuk mendapatkan kunci rahasia yang mereka bagi.

   Alice : K_{AB} = a.(bB)
   Bob : K_{AB} = b.(aB)
   Sehingga kunci rahasia = K_{AB} = abB

III. RANCANGAN SISTEM

Pada bagian ini, akan dijelaskan rancangan sistem yang akan dibuat. Rancangan meliputi algoritma otentikasi dan perangkat lunak simulasi untuk pengujian yang akan dijalankan pada terminal.

A. Rancangan Algoritma Otentikasi

Algoritma autentikasi yang akan dibuat terdiri dari dua bagian utama, yaitu sistem pembangkitan *one-time password*, dan sistem penandatanganan dengan kunci publik menggunakan algoritma ECC. *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 OTP dilakukan berdasar waktu, dengan cara menghitung selang waktu saat ini (T_1) 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 kemudian digabung dengan *seed* yang sudah ditentukan, Hasil penggabungan kemudian di-*hash* untuk mendapatkan *one-time password*.

Setelah OTP dibangkitkan, *password* tersebut dikirimkan melalui nomor telepon. Sedangkan kunci publik pengguna dikirim ke *server*. Setelah kunci publik didapatkan, nomor telepon pengguna digabungkan dengan kode OTP kemudian di *hash* dengan kunci rahasia sebanyak n kali.

Untuk melakukan otentikasi, pengguna cukup mengirimkan OTP dan kunci publik milik pengguna. Setelah itu, *server* melakukan perhitungn kunci rahasia kembali dari kunci publik yang telah diterima bersama dengan OTP. Hal ini untuk membuktikan bahwa pengguna yang sama yang mengirimkan OTP yang sama. Kemudian *server* melakukan *hash* pada OTP tersebut sebanyak n kali. Jika *hash* yang dihasilkan sama dengan sebelumnya, maka pengguna terotentikasi, jika tidak tolak OTP.

B. Rancangan Perangkat Lunak

Perangkat lunak yang akan dibuat untuk pengujian ini hanya berupa program yang dapat dijalankan pada terminal. Program tersebut dapat membangkitkan OTP dari *seed* yang telah dibangun oleh sistem, membangkitkan kunci dengan algoritma ECC, dan mengotentikasi OTP yang diberikan.

Untuk mensimulasikan *session* pengguna, maka program cukup menyimpan *hash* dari OTP yang telah dibangkitkan. Dengan begitu, program hanya menyimpan OTP untuk satu kali *run*.

IV. IMPLEMENTASI SISTEM

Sistem yang akan dibuat diimplementasikan dengan menggunakan Bahasa Python 2.7, serta *library* untuk hash yang sudah tersedia. Pembangkitan OTP menggunakan identitas pengguna berupa nomor telepon yang dihash sebanyak n kali dengan menggunakan kunci yang telah dibangun dengan menggunakan SHA-256.

Pasangan kunci privat dan kunci publik dibangkitkan dengan menggunakan algoritma ECC yang dbangun sendiri. Panjang kunci yang digunakan adalah 160-bit. Hasil *hash* lalu dibuat tanda tangan digitalnya untuk dikirim oleh. *Server* kemudian dapat melakukan otentikasi pada tanda tangan digital berdasarkan perhitungan dari hasil *hash* OTP.

V. KESIMPULAN DAN SARAN PENGEMBANGAN

Penggabungan antara *one-time password* dan algoritma kriptografi kunci publik memungkinkan otentikasi yang cukup aman. Dengan demikian, meskipun terjadi serangan *man in the middle attack*, identitas yang dicuri tidak akan dapat digunakan.

Metode ini masuh dapat dikembangkan lebih lanjut dengan mekanisme untuk membuat OTP yang dihasilkan menjadi kadaluarsa ketika dicoba beberapa kali tetapi tidak berhasil.

REFERENCES

[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] Munir, Rinaldi. 2017. Bahan Kuliah Algoritma ECC

PERNYATAAN

Dengan ini kami menyatakan bahwa makalah yang kami tulis ini adalah tulisan kami berdua, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2018

Anwar Ramadha