Perbandingan keamanan ECDSA, ED25519 pada openSSH

Azis Adi Kuncoro
Sekolah Teknik Elektro dan Informatika (STEI)
Institut Teknologi Bandung (ITB)
Bandung, Indonesia
13515120@std.stei.itb.ac.id

***Abstract*—Elliptic-curve cryptography (ECC) merupakan pendekatan kriptografi kunci public berdasarkan struktur dari kurva eliptik pada bidang yang terbatas. Terdapat beberapa algoritma dasar yang digabungkan dengan ECC, sehingga mewarisi juga dari sifat – sifat ECC, seperti ECDSA, dan ED25519. Algoritma – algoritma tersebut diimplementasikan pada openSSH. Pada makalah ini akan dibahas mengenai tingkat keamanan dari algoritma – algoritma tersebut.**

Keywords—ecc; ecdsa; ecdh; ed25519; curve25510; openssh;

1. Pendahuluan

Kriptografi adalah ilmu dan seni untuk menyembunyikan pesan (Schneier, 1996). Berdasarkan garis waktu, terdapat 2 jenis teknik kriptografi, yaitu kriptografi lama (*Old Cryptography*), yaitu teknik kriptografi yang muncul sebelum adanya komputer, dan kriptografi modern (*Modern Cryptography*), yaitu teknik kriptografi yang muncul setelah adanya komputer. Untuk saat ini, teknik kriptografi modern banyak digunakan, dikarenakan teknik kriptografi lama sudah dapat dipecahkan dengan komputer dalam waktu singkat, sedangkan beberapa teknik kriptografi modern masih sulit dipecahkan meski menggunakan komputer yang umum digunakan.

Dewasa ini, proses pertukaran informasi dari pengirim ke penerima sangatlah sering dilakukan. Pertukaran ini terjadi pada banyak perangkat sehari - hari kita. Konten - konten yang dikirimkan pada pesan bisa jadi mengandung informasi yang tidak boleh diketahui oleh orang lain. Untuk menjaga kerahasiaan informasi yang dipertukarkan, digunakan banyak teknik kriptografi agar kerahasiaan dapat terjamin melalui aspek - aspek keamanan.

Informasi dipertukarkan melalui medium internet, pesan - pesan yang dikirimkan direpresentasikan dengan byte - byte pesan. Terdapat aturan yang mendefinisikan bagaimana pengirimannya yang disebut protokol. Protokol untuk mengirimkan byte - byte pesan tersebut seperti TCP, UDP, dan lain sebagainya. Apabila protokol tersebut tidak melakukan enkripsi, informasi tersebut sangat mudah disadap oleh pihak lain. Untuk menjaga keamanan dari data yang akan dikirimkan, dibutuhkan protokol yang menjaga kerahasiaannya.

Salah satu implementasi dari protokol SSH adalah openSSH. Implementasi ini bersifat *open source*, dan kaya kan fitur – fitur dan pilihan algoritma yang dapat digunakan. Padamakalah ini akan dibahas mengenai tingkat keamanan dari kedua algoritma tersebut.

2. OpenSSH

SSH (Secure Shell) adalah alat untuk administrasi sistem yang aman, transfer file, dan komunikasi lainnya di internet atau jaringan tidak terpercaya lainnya. OpenSSH melakukan enkripsi identitas, kata sandi, dan data yang dikirimkan sehingga tidak dapat disadap oleh pihak lain

OpenSSH adalah implementasi open source dari protokol SSH. Hal ini didasarkan pada versi gratis oleh Tatu Ylonen dan dikembangkan lebih lanjut oleh tim OpenBSD dan komunitas pengguna.

Terdapat beberapa fitur yang disediakan dari OpenSSH, diantaranya adalah :

- *Secure remote login*

  Fitur ini memungkinkan pengguna untuk dapat login pada komputer lain melalui internet. Semua data yang dikirimkan pada *remote login* (seperti informasi *username* dan *password*) telah di enkripsi. Hal ini mendukung keamanan sehingga informasi tersebut tidak dapat disadap.

- *Secure file transfer*

  Fitur ini memungkinkan pengguna untuk mengirimkan *file*pada dua komputer yang sudah terhubung. *File* yang dikirim dijaga kerahasiannya dengan cara dienkripsi.

- *Secure remote command execution*

  Fitur ini memungkinkan pengguna untuk mengeksekusi perintah – perintah secara *remote*. Perintah dieksekusi seolah – olah dilakukan pada komputer target.

- *Access control*

  Fitur ini memungkinkan pemberian otoritas kepada pengguna lain dan juga dapat membatasi hal- hal yang dapat dilakuakn oleh pengguna tersebut dalam suatu sistem.

- *Port forwarding*

  Fitur ini memungkinkan peningkatan keamanan aplikasi berbasis TCP / IP lainnya seperti telnet, ftp, dan Sistem X Window. Teknik ini disebut juga

dengan *tunneling*. Cara kerja teknik ini adalah dengan mengubah rute koneksi TCP / IP untuk melewati koneksi SSH, mengenkripsi juga secara transparan dari ujung ke ujung.

3. ECDSA

ECDSA (*Elliptic Curve Digital Signature Algorithm*) dikembangkan oleh *American National Standards Institute* pada tahun 2005. ECDSA merupakan variasi dari DSA (*Digital Signature Algorithm*) yang menggunakan teknik kriptografi kurva eliptik.

a. Perbandingan kunci dan ukuran tanda tangan dengan DSA

Seperti teknik kriptografi kurva eliptik secara umum, ukuran bit kunci publik yang diperlukan untuk ECDSA adalah sekitar dua kali ukuran tingkat keamanan dalam bit. Sebagai contoh, pada tingkat keamanan 80 bit (artinya penyerang memerlukan maksimum sekitar 2^{80} operasi untuk menemukan kunci privat) ukuran kunci publik ECDSA akan menjadi 160 bit, sedangkan ukuran kunci publik DSA setidaknya adalah 1024 bit. Ukuran tanda tangan sama untuk DSA dan ECDSA, yaitu sekitar 4 t bit, dimana t adalah tingkat keamanan yang diukur dalam bit, yaitu sekitar

b. Pembangkitan tanda tangan

Untuk dapat membangkitkan tanda tangan, berikut merupakan langkah – langkah yang dilakukan pada algoritma ECDSA yang dapat dilihat pada Tabel 1.

Tabel 1. Algoritma pembangkitan tanda tangan

Algoritma pembangkitan tanda tangan
1. Hitung e = HASH(m) 2. z = L_n bit paling kiri dari e, dimana L_n merupakan panjang bit dari orde grup n. 3. Memilih bilangan acak k dari 1 hingga n - 1. 4. Hitung titik kurva $(x_1, y_1) = k \times G$ 5. Hitung $r = x_1 \bmod n$. Jika r = 0, kembali ke langkah 3. 6. $s = k^{-1}(z + r\, d_A) \bmod n$. Jika s = 0 kembali ke langkah 3. 7. Pasangan titik yang valid (r, s) atau (r, -s mod n)

c. Verifikasi tanda tangan digital

Berikut merupakan skenario verifikasi dari tanda tangan digital dengan menggunakan ECDSA.

Bob akan melakukan verifikasi dari tanda tangan digital yang diberikan oleh alice, pertama – tama ia harus memiliki salinan kunci publik dari alice yang disebut dengan Q_A. Bob dapat melakukan verifikasi apakah Q_A valid melalui langkah – langkah berikut :

1) Periksa apakah Q_A tidak sama dengan elemen identitas O, koordinatnya pun harus valid.
2) Periksa apakah Q_A berada pada kurva eliptik.
3) Periksa apakah $n \times Q_A = O$.

Setelah itu, bob akan melakukan verifikasi dengan mengikuti langkah – langkah berikut:

1. Memastikan r dan s merupakan bilangan bulan diantara 1 hingga n-1. Jika tidak, maka tidak valid
2. Menghitung e = HASH(m), dimana HASH merupakan fungsi yang sama digunakan pada pembuatan tanda tangan.
3. z = L_n bit paling kiri dari e.
4. Hitung $u_1 = z\, s^{-1} \bmod n$ dan $u_2 = r\, s^{-1} \bmod n$.
5. Hitung $(x_1, y_1) = u_1 \times G + u_2 Q_A$. Jika $(x_1, y_1) = O$ maka tanda tangan tidak valid.
6. Tanda tangan valid apabila $r = x_1 \pmod n$, invalid jika tidak sama.

4. Ed25519

Ed25519 diperkenalkan pertama kali pada openSSH versi 6.5. Algoritma ini merupakan implementasi EdDSA menggunakan *twisted edwards curve* yang dijamin memiliki tingkat keamanan yang lebih baik dan kinerja yang lebih cepat dibandungkan dengan DSA ataupun ECDSA.

Ed25519 mendukung tingkat keamanan 128 bit, yang berarti dibutuhkan 2^{128} operasi untuk menemukan kunci. Algoritma ini memiliki beberapa karakteristik untuk menjadikannya salah astu tanda tangan digital yang baik, diantaranya adalah :

- Verifikasi satu tanda tangan cepat

Perangkat lunak seperti (https://ed25510.cr.yp.to/software.html) hanya membutuhkan 273.364 siklus untuk melakukan verifikasi tanda tangan pada CPU *Nehalem / Westmere* yang digunakan secara luas oleh intel. Pengukuran kinerja ini untuk pesan yang pendek, untuk pesan yang sangat panjang, waktu verifikasi didominasi oleh waktu *hashing*.

- Kunci yang kecil

Panjang kunci yang digunakan pada Ed25510 sebesar 256 bit atau 32 bytes.

- Tanda tangan yang kecil

Pada Ed25519 panjang tanda tangannya adalah 512 bit atau 64 bytes, hal ini merupakan angka yang cukup kecil diantara lainnya.

- Deterministik

Tidak seperti ECDSA, Ed225519 tidak bergantung pada sumber entropi ketika menandatangani pesan. Sumber entropi dapat menjadi potensi penyerangan apabila tidak membangkitkan bilangan acak yang baik. Ed25519 mengatasi masalah ini dengan selalu membangkitkan tanda tangan yang sama untuk suatu data.

- *Collision resistant*

Adanya kolisi pada fungsi hash tidak dapat merusak tanda tangan yang dihasilkan dari algoritma ini.

- *Fool proof session keys*

Tanda tangan dihasilkan secara deterministik, tetapi pembangkitan kunci mengalami pembangkitan yang non deterministik. Hal ini bukan hanya fitur dari segi kecepatan,

tetapi juga dari segi keamanan, yang secara langsung relevan dengan runtuhnya sistem keamanan *Playstation 3*.

- Tingkatk keamanan yang tinggi

Sistem ini memiliki target keamanan 2^{128}, percobaan untuk memecahkannya memiliki tingkat kesulitan yang sama dengan NIST P-256, RSA dengan ~3000 bit kunci, blok cipher 128 bit yang kuat, dll. Serangan terbaik yang diketahui rata – rata menghabiskan lebih dari 2^{140} bit rata – rata operasinya.

Skema pembangkitan kunci dan juga proses tanda tangan serupa dengan EdDSA, algoritma tersebut dapat dilihat pada Gambar 1.

Algorithm 1 EdDSA key setup and signature generation

Key setup.
1: Hash k such that $H(k) = (h_0, h_1, \ldots, h_{2b-1}) = (a, b)$
2: $a = (h_0, \ldots, h_{b-1})$, interpret as integer in little-endian notation
3: $b = (h_b, \ldots, h_{2b-1})$
4: Compute public key: $A = aB$.
Signature generation.
5: Compute ephemeral private key: $r = H(b, M)$.
6: Compute ephemeral public key: $R = rB$.
7: Compute $h = H(R, A, M)$ and convert to integer.
8: Compute: $S = (r + ha) \mod l$.
9: Signature pair: (R, S).

Gambar 1 Algoritma tanda tangan dengan Ed25519

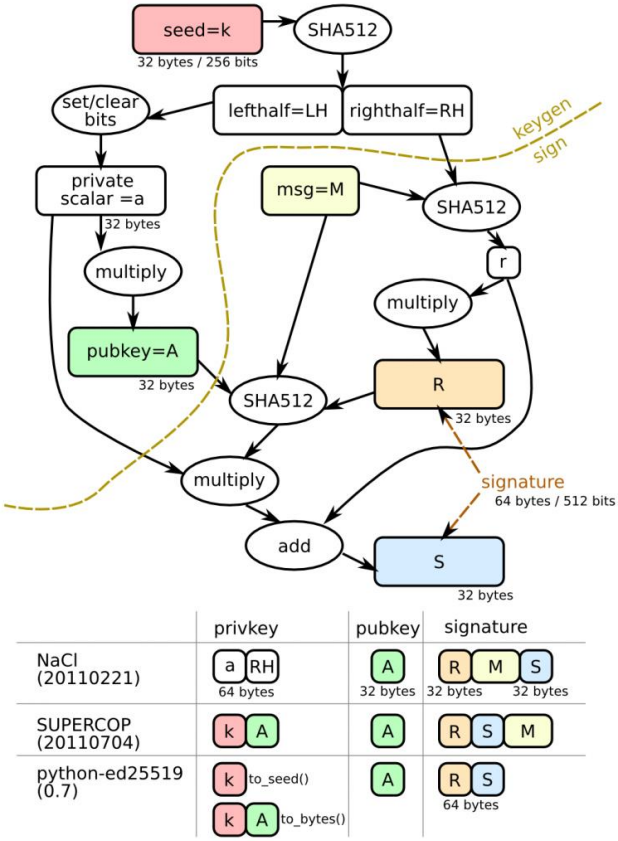
Sedangkan mekanisme verifikasi pada EdDAS pun juga tidak berbeda dengan EdDSA, algoritma tersebut dapat dilihat pada Gambar 2.

Algorithm 2 EdDSA signature verification

1: Compute $h = H(R, A, M)$ and convert to integer.
2: Check if group equation $8SB = 8R + 8hA$ in E holds.
3: If group the equation holds, the signature is correct.

Gambar 2 Algoritma verifikasi pada Ed25519

Secara umum, terdapat skema yang menjelaskan secara visual melalui graf mengenai pembangkitan kunci dan proses tanda tangan digital, hal ini dapat dilihat pada Gambar 3.



Gambar 3 Skema pembangkitan kunci dan tanda tangan digital pada Ed25519

5. Percobaan serangan pada ECDSA

Terdapat percobaan yang telah dilakuakn oleh Joachin Breitner tentang *Biased Nonce Sense: Lattice Attacks against weak ECDSA signatures in cryptocurrencies.* Pada makalah tersebut dibangkitkan ratusan kunci privat bitcoin dan puluhan etherium, ripple, SSH, dan HTTPS. Dengan melakukan penyerangan kriptanalitik terhadap tanda tangan digital yang terkandung pada blockchain publik. Karena tanda tangan digital yang dihasilkan dari ECDSA membutuhkan pembangkitan *nonce* untuk setiap pesan. Jika *nonce* ini tidak dibangkitkan secara *uniform random*, penyerang dapat memiliki potensi untuk mengeksploitasi bias ini dengan menghitung signing key / private key. Makalah tersebut menggunakan *lattice-based algorithm* untuk menyelesaikan permasalahan bilangan yang tersembunyi untuk menghitung secara efisien kunci privat ECDSA.

Sebagai contoh, berikut merupakan hasil dan analisis dari percobaan pada Ethereum. Data yang diperoleh dari tanda tangan digital DSA dan ECDSA berasal dari melakukan query pada *interface* RPC. Didapatkan sekitar 311 juta tanda tangan dengan 34 juta tanda tangan digital yang unik. Kriptanalisis dilakukan dengan melakukan klustering tanda tangan digital, dan memeriksa kunci yang dibangkitkan lebih dari satu kali. Hasil nya dapat dilihat pada Tabel 2.

Tabel 2 Nonce tanda tangan digital yang berulang

Nonce	Total Signature	Repeated Nonce Signature	Distinct Keys
(n-1)/2	4.275.639	2.456.870	918
Lainnya	19.052	2.214	378

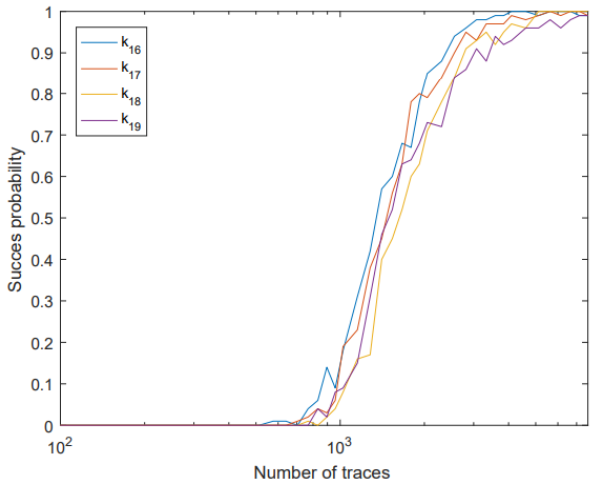
6. Percobaan serangan pada Ed25519

Terdapat percobaan yang telah dilakukan oleh Niels Samwel yang berjudul *Breaking Ed25519 in WolfSSL*. Pada makalah tersebut ditunjukkan kasus dimana kebocoran elektromagnetik dapat dieksploitasi, serupa dengan mekanisme yang membuat EdDSA deterministik yang mempersulit pelaksanaan keamanannya. Dijelaskan bahwa telah berhasil dipatahkan Ed25519 pada implementasi di WolfSSL yang mana digunakan pada aplikasi – aplikasi IoT. Dilakukan *differential power analysis* (DPA) pada fungsi hash SHA-512, yang membutuhkan sekitar 4.000 jejak. Ditemukan bahwa terdapat cara untuk melakukan *tweaking* pada protokol EdDSA yang murah dan efektif terhadap penyerangan yang telah disebutkan sebelumnya.

7. Kesimpulan

Berdasarkan hasil percobaan serangan pada kedua algoritma tersebut didapatkan bahwa Ed25519 mampu menangani serangan lebih baik daripada ECDSA. Hal ini dikarenakan Ed25519 menggunakan kurva edward twist yang memiliki tingkat keamanan lebih baik. Kunci publik yang digunakan pada Ed25519 juga lebih ringkas karena mengandung sebanyak 68 karakter saja.

Pada gambar 4 dapat dilihat probablitias dari serangan pada tipe data *word* yang tidak diketahui k_{16}, ... , k_{19}. Untuk setiap data titik pada gambar, dilakukan penyerangan sebanyak 100 kali dengan beberapa jejak. Terlihat penyerangan berhasil jika seluruh 64 bit dari tipe *word* yang dipulihkan dengan benar dengan menggunakan serangan pada sebuah *byte* sebanyak 8 kali. Gambar tersebut menujukkan bahwa probabilitas kesuksesan dari serangan secara cepat meningkat ketika lebih dari 1000 jejak digunakan.



Gambar 4 Probabilitas keberhasilan percobaan serangan

UCAPAN TERIMAKASIH

Penulis mengucapkan terimakasih kepada Tuhan Y.M.E., karena atas berkat, rahmat dan karunia-Nya makalah ini dapat diselesaikan pada waktunya. Tak lupa terimakasih kepada dosen pengampu matakuliah ini yaitu beliau bapak Dr. Ir. Rinaldi Munir yang memberikan tugas ini sehingga penulis mendapatkan pengalaman baik dalam membuat makalah. Terakhir, penulis juga mengucapkan terimakasih kepada teman – teman yang secara langsung maupun tidak langsung membantu penyelesaian makalah ini.

DAFTAR PUSTAKA

[1] Samwel, Niels. 2017. Breaking Ed25519 in WolfSSL. IACR.

[2] Breitner, Joachim. 2019. Biased Nonce Sense: Lattice Attacks against Weak ECDSA Signatures in Cryptocurrencies. IACR.

[3] Munir, Rinaldi. 2015. Slide Kuliah IF4020 Kriptografi: Tanda – tangan digital

[4] Munir, Rinaldi. 2015. Slide Kuliah IF4020 Kriptografi: Digital Signature Algorithm

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, 10 Mei 2019

Azis Adi Kuncoro (13515120)