Implementasi Curve25519 pada Aplikasi Chat dengan Socket

Varian Caesar - 13514041
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
13514041@std.stei.itb.ac.id

Abstrak — Pada makalah ini, akan diimplementasikan sebuah aplikasi *chatting room* sederhana dengan menggunakan bahasa pemrograman Python dan memanfaatkan *socket*. Pengguna dapat masuk dan mengirimkan pesan (*chat*). Untuk mengamankan pesan, pesan akan dienkripsi dengan algoritma BeRez menggunakan algoritma pertukaran kunci ECDH (*Elliptic Curve Diffie-Hellman*). ECDH yang digunakan akan memiliki kurva yang berbeda, yaitu kurva 25519. Kurva 25519 atau *Curve25519* adalah kurva eliptik yang memiliki kerumitan tinggi namun operasi didalamnya sangat murah. Pengujian pada makalah ini membuktikan performa *Curve25519* dibandingkan beberapa contoh kurva lainnya dari segi waktu dan memori.

Kata Kunci — *Berez; chat; Curve25519; Elliptic Curve Cryptography; Elliptic Curve Diffie-Hellman; socket.*

I. PENDAHULUAN

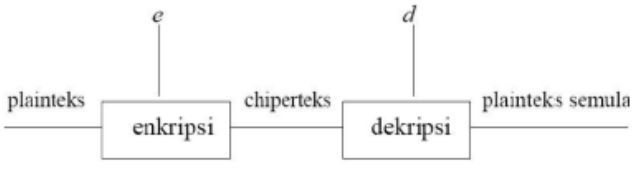
Aplikasi *chat* atau *messenger* semakin hari menjadi sesuatu yang tidak asing bagi masyarakat. Berkembangnya teknologi membuat aplikasi ini menjadi sebuah sarana yang membantu produktivitas kerja. Tidak jarang, informasi rahasia dan penting pun dikirim melalui aplikasi *chat* ini. Hal ini menjadi sebuah peluang bagi oknum yang tidak bertanggung jawab untuk melakukan kejahatan. Pesan yang dikirim sangat rawan untuk disadap dan diubah kontennya, sehingga menimbulkan kerugian bagi si pengirim pesan. Para kriptografer dan ahli sekuriti sejak jaman dahulu telah berusaha untuk mencegah hal ini dengan menciptakan beberapa algoritma kriptografi untuk mengamankan pesan ditempat pengirim maupun penerima. Kriptografi terus berkembang dari yang sifatnya klasik hingga yang modern.

Salah satu jenis algoritma kriptografi modern yang sering digunakan adalah algoritma kriptografi kunci publik atau yang bisa disebut juga dengan kriptografi kunci asimetris. Algoritma kunci publik memiliki dua buah kunci dalam proses pengamanan pesan, yaitu kunci publik dan kunci privat. Kunci privat adalah kunci yang digunakan untuk proses dekripsi dan sifatnya rahasia, sedangkan kunci publik adalah kunci yang digunakan untuk melakukan enkripsi dan sifatnya tidak rahasia. Untuk saling bertukar kunci antar pengirim dan penerima, dibutuhkan sebuah mekanisme pertukaran kunci. ECDH merupakan salah satu algoritma pertukaran kunci yang cukup aman, karena sulitnya menyelesaikan logaritma diskrit pada kurva eliptik. Pada makalah ini akan diimplementasikan sebuah aplikasi *chatroom* sederhana dengan menggunakan *socket*. Proses enkripsi dan dekripsi pesan pada aplikasi ini akan menggunakan algoritma pertukaran kunci ECDH yang menggunakan kurva 25519. Kurva 25519 atau *Curve25519* merupakan kurva eliptik yang operasinya murah tetapi tetap aman karena kompleksitasnya.

II. DASAR TEORI

A. Kriptografi Kunci Publik

Algoritma kunci publik adalah algoritma kriptografi yang menggunakan kunci berbeda untuk proses enkripsi dan dekripsi. Terdapat dua buah kunci yang digunakan, yaitu kunci privat dan kunci publik. Kunci privat adalah kunci yang sifatnya rahasia dan digunakan untuk melakukan dekripsi pesan. Kunci publik adalah kunci yang bersifat terbuka dan tidak rahasia, kunci ini digunakan untuk melakukan enkripsi pada pesan. Pemilihan kunci publik dan privat tidak boleh sembarang, biasanya akan dipilih sebuah nilai yang cukup besar agar keamanan semakin terjaga dan semakin sukar untuk menyelesaikan persamaan logaritma matematika. Secara umum, proses algoritma kunci publik dapat dilihat pada Gambar 1.



Gambar 1. Cara kerja algoritma kriptografi kunci publik

Pesan di tempat pengirim akan dienkripsi dengan menggunakan kunci publik penerima pesan. Pesan yang telah terenkripsi ini disebut *ciphertext* dan akan dikirimkan melalui saluran komunikasi *chat* tersebut. Sampai di tempat penerima, penerima akan melakukan dekripsi pesan menggunakan kunci privat yang dimilikinya untuk mendapatkan pesan semula.

Algoritma kunci publik banyak digunakan karena kunci publik yang digunakan dapat disebar kemanapun tanpa perlu khawatir. Bahkan kunci publik dapat dikirim menggunakan saluran yang tidak aman, karena pihak penyadap tidak akan bisa berbuat apapun untuk menjebol kemanan hanya dengan kunci publik saja.

B. Berez Cipher

Berez cipher adalah algoritma block cipher yang dikembangkan oleh Bervianto Leo dan M. Reza Ramadhan pada tahun 2018. *Berez Cipher* menerima masukan blok dan ukuran kunci 256 bit. Algoritma *Berez* menggunakan jaringan feistel didalamnya, sehingga memenuhi prinsip konfusi dan difusi pada desain algoritma *block cipher* yang baik. Jumlah putaran yang ada pada *Berez Cipher* dinamis, jumlah putaran yang mungkin terjadi berada pada rentang 7 hingga 25 putaran. Jumlah putaran bergantung terhadap bilangan acak yang dibangkitkan dengan *seed* berbasis kunci.

Operasi yang dilakukan pada jaringan feistel milik algoritma ini adalah sebagai berikut: Blok kanan akan dimasukan pada fungsi f beserta kunci internal untuk putaran tersebut. Hasil dari fungsi f akan diXOR pada blok kiri. Hasil dari operasi tersebut digunakan sebagai blok kanan pada *round* berikutnya, sedangkan blok kanan menjadi blok kiri pada *round* berikutnya. Pada blok terakhir tidak perlu dilakukan pertukaran. Fungsi f yang digunakan terdiri dari:

1. XORRoundKey
   Operasi XORRoundKey adalah operasi untuk melakukan XOR antara blok pesan dengan kunci internal pada putaran tersebut.

2. ShiftByKey
   Pergeseran pada setiap baris blok plainteks berdasarkan blok kunci internal. Operasi ini dilakukan untuk mencapai prinsip konfusi pada Berez Cipher. Baris genap pada blok akan digeser ke kiri, sementara pada baris ganjil akan digeser ke kanan. Jumlah pergeseran berdasarkan nilai setiap baris pada kunci.

3. XORShifted
   XORShifted dilakukan dengan melakukan XOR dari matriks hasil operasi ShiftByKey dengan sebuah matriks yang berasal dari plainteks awal sebelum dilakukan operasi XORRoundKey yang sudah dilakukan permutasi dan pergeseran. Permutasi dilakukan untuk mengubah baris pada matriks menjadi kolom, dan begitu pula sebaliknya. Operasi ini dilakukan untuk meningkatkan prinsip difusi.

4. Substitution
   Substitusi merupakan proses terakhir yang terjadi di dalam jaringan feistel. Proses ini melakukan substitusi dari matriks hasil XORShifted dengan menggunakan S-box

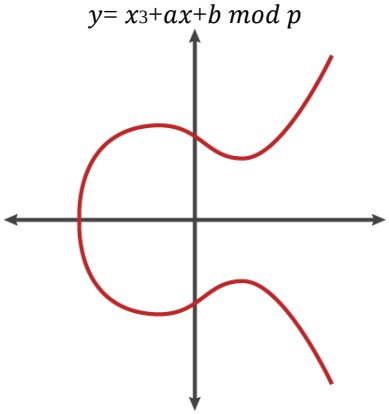
Sedangkan proses pembangkitan kunci internal diperoleh dengan tahapan sebagai berikut:

1. Membagi kunci eksternal menjadi dua buah bagian yang berukuran 128 bit berdasarkan posisi ganji genap bit tersebut.
2. K_1 yaitu kunci putaran pertama diperoleh dari hasil XOR kedua blok tersebut
3. K_2 didapatkan dengan melakukan XOR antara K_1 dan blok hasil substitusi K_1 menggunakan S-box
4. Untuk K_n ulangi langkah 3

C. ECDLP dan ECC

ECDLP (*Elliptic Curve Discrete Logarithm Problem*) adalah dasar permasalahan dan kekuatan dari ECC, yaitu sulitnya memecahkan logaritma diskrit, diberikan P dan Q adalah dua buah titik pada kurva eliptik, carilah k sehingga Q = k P. Contoh persamaan kurva eliptik sebagai berikut:

$$y = x^3 + ax + b \bmod p$$



Gambar 2. Contoh Kurva Eliptik

Berikut adalah beberapa penjelasan mengenai operasi yang terjadi pada kurva eliptik.

1. Penjumlahan
   Misal titik P + Q adalah titik R, maka:
   Koordinat titik R dapat ditentukan dengan persamaan berikut:

$$x_r = \lambda_2 - x_p - x_q \bmod p$$
$$y_r = \lambda(x_p - x_r) - y_p \bmod p$$

   Dengan gradien yang mengikuti persamaan:

$$\lambda = \frac{y_p - y_q}{x_p - x_q} \bmod p$$

2. Pengurangan
   Misal titik P – Q adalah titik R. P – Q ekuivalen dengan P + (-Q). Dalam hal ini –Q adalah (x_q, $-y_q$). Koordinat titik R dapat ditentukan dengan persamaan berikut:

$$x_r = \lambda_2 - x_p - x_q \ mod\ p$$
$$y_r = \lambda(x_p - x_r) - y_p \ mod\ p$$

   Dengan gradien yang mengikuti persamaan:

$$\lambda = \frac{y_p + y_q}{x_p - x_q} \ mod\ p$$

3. Penggandaan titik
   Misalkan terdapat titik P (x_p, y_p), penggandaan titik 2P = R mengikuti persamaan:
   Koordinat titik R:

$$x_r = \lambda^2 + 2x_p \text{ mod p}$$
$$y_r = \lambda\left(x_p - x_r\right) - y_p \text{ mod p}$$

   Gradien:

$$\lambda = \frac{3x_p - a}{2y_p} \ mod\ p$$

4. Perkalian titik
   Perkalian titik kP = Q diperoleh dengan perulangan dua operasi dasar yang telah dijelaskan sebelumnya, yaitu penjumlahan dan penggandaan.
   Contoh: 23P = 2 (2 (2 (2P) + P) + P) + P

ECC (*Elliptic Curve Cryptography*) adalah algoritma kunci publik yang menggunakan kurva eliptik sebagai sarana pembangkitan dan pengelolaan kunci. Operasi – operasi pada ECC adalah seperti yang sudah dijelaskan sebelumnya. ECC diajukan pada tahun 1985 oleh Neal Koblitz dan Victor Miller. Implementasi dari ECC meliputi ECEG (*Elliptic Curve El-Gamal*), ECDSA (*Elliptic Curve Digital Signature*) dan ECDH (*Elliptic Curve Diffie-Hellman*).

D. ECDH

ECDH (*Elliptic Curve Diffie-Hellman*) adalah versi kurva eliptik dari algoritma pertukaran kunci *Diffie-Hellman*. ECDH digunakan untuk proses pertukaran kunci pada saluran yang tidak aman. Beberapa properti dari ECDH adalah:

1. Persamaan kurva eliptik (publik)
2. Titik pada kurva eliptik (publik)
3. Bilangan bulat acak (privat)

Misalkan Alice dan Bob ingin bertukar pesan, sebelumnya mereka perlu melakukan pertukaran kunci. Alice dan Bob masing – masing akan membangkitkan sebuah bilangan acak yang bersifat rahasia dan tidak boleh diketahui, misalkan a dan b. Alice dan Bob lalu mengalikan bilangan bulat yang dimilikinya dengan sebuah titik pada kurva eliptik untuk menghasilkan kunci publik, misalkan titik ini adalah Z. Persamaanya sebagai berikut:

$$PK_a = a \ . \ Z$$
$$PK_b = b \ . \ Z$$

Alice dan Bob lalu bertukar hasil perhitungan mereka satu sama lain dengan menggunakan saluran publik. Alice dan Bob lalu akan menghitung *shared key* dengan menggunakan persamaan:

$$SK\_ALICE = a \ . \ PK_b$$
$$SK\_BOB = b \ . \ PK_a$$

Hasil perhitungan akan sama antara Bob dan Alice karena ab = ba.

E. *Curve*25519

Kurva 25519 atau *Curve25519* adalah kurva eliptik yang menawarkan keamanan yang cukup kuat dan didesain untuk digunakan bersamaan dengan skema pertukaran kunci ECDH. *Curve25519* merupakan kurva ECC yang tercepat yang sampai sekarang diketahui oleh paten.

Kurva ini pertama kali diajukan oleh Daniel J. Bernstein pada tahun 2005 dan kini implementasinya terbuka untuk publik. Kurva 25519 sendiri memiliki persamaan :

$$y^2 = x^3 + 486662x^2 + x$$

Curve25519 didesain untuk menghindari berbagai macam kemungkinan kegagalan dan serangan. Kurva ini menerima masukan 32 *byte string* kunci publik tanpa validasi.

III. Rancangan dan Implementasi

Pada bagian ini akan dijelaskan mengenai rancangan dan proses implementasi aplikasi *chat* yang dilengkapi *Curve25519* sebagai algoritma pertukaran kunci.

A. Rancangan Aplikasi

Aplikasi yang dibangun merupakan aplikasi *chatting room* sederhana dengan menggunakan socket sebagai media pertukaran data. Terdapat dua buah komponen utama pada aplikasi ini, yaitu *server* dan *client*.

1. *Server*
   Seperti *socket* pada umumnya, *server* adalah komponen aplikasi yang akan terus berjalan untuk menerima koneksi dari klien. Server akan membuat koneksi *socket* dan akan melakukan *bind* pada *port* tertentu untuk mendengarkan koneksi. Klien yang

melakukan koneksi ke *domain* dan *port* yang sama dengan *server* akan masuk ke dalam aplikasi *chatting*.

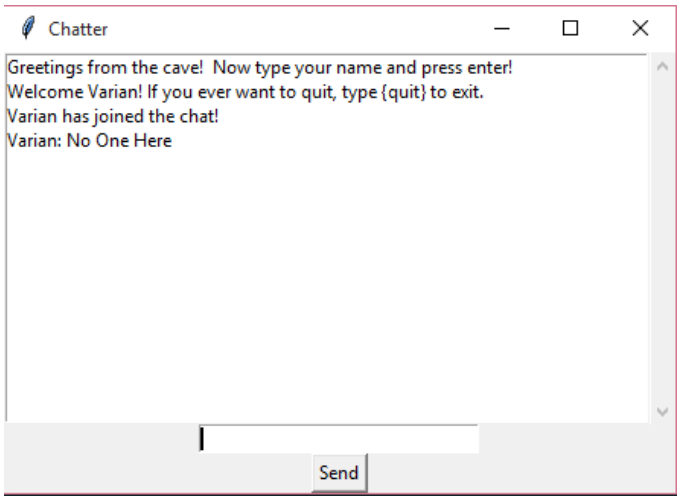
Klien yang terhubung lalu akan didaftarkan pada struktur data klien yang terdapat pada *server*. Karena bersifat *chatting room*, *server* juga bertugas untuk melakukan *broadcasting* pesan yang ditulis oleh klien. Jika ada klien yang keluar dari aplikasi, maka *server* akan mengumumkan ke *chatting room* bahwa klien yang bersangkutan sudah tidak berada pada *chatting room*.

2. *Client*

Client adalah pengguna aplikasi yang akan melakukan koneksi ke *server* untuk dapat terhubung dan saling mengirimkan pesan. Pada saat pertama kali dijalankan, *client* akan menerima masukan berupa *domain* dan *port* mana yang akan dituju. Setelah itu, *socket* akan mencoba membuka koneksi ke *domain* dan *port* yang bersangkutan. Apabila berhasil, maka kini *client* telah terhubung ke *server*.

Server lalu akan bertanya mengenai *username* pengguna. *Client* dapat memasukan *username* yang ingin ia gunakan pada saat menggunakan aplikasi. *Client* juga dapat keluar dari aplikasi dengan cara mengetikkan perintah '{quit}' ke *server*.

Gambar 3 merupakan antarmuka aplikasi *chatting room* yang dibangun.



Gambar 3. Antarmuka aplikasi

B. Rancangan Sistem Pertukaran Kunci

Pertukaran kunci yang dilakukan menggunakan ECDH dan kurva 25519. Implementasi memanfaatkan *library cryptography* pada Python untuk mempersingkat waktu dan juga memastikan bahwa implementasi yang dibuat tidaklah *buggy*. Untuk memudahkan pembahasan, proses pertukaran kunci akan dijelaskan dalam kelompok berikut:

1. Pembangkitan Kunci

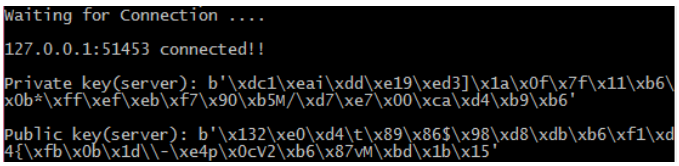
Pembangkitan kunci terdiri dari pembangkitan kunci privat dan kunci publik. Masing – masing *server* dan *client* akan melakukan proses pembangkitan bilangan acak berukuran 32 bytes. Dari bilangan acak ini, akan dibangkitkan kunci publik yang berukuran 32 bytes pula. Gambar 4 merupakan contoh pembangkitan kunci privat dan publik menggunakan *Curve25519*.

2. Pertukaran Kunci

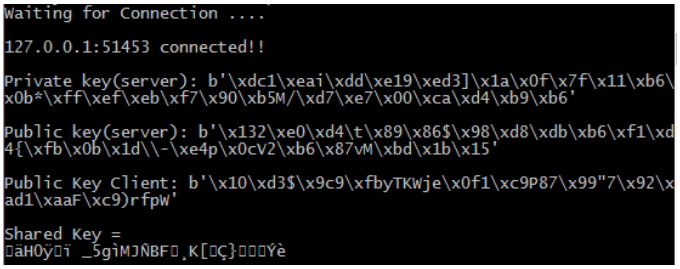
Setelah *server* dan *client* membangkitkan kunci privat dan kunci publik nya masing – masing, *client* akan melakukan koneksi ke-*server* dan akan terjadi pertukaran kunci untuk membangkitkan *shared key* yang sama. Secara detil, urutannya adalah sebagai berikut:

1. *Client* dan *Server* membangkitkan pasangan kunci privat dan publik.
2. *Client* melakukan koneksi ke *server*
3. *Server* menerima koneksi *client* dan mengirimkan kunci publik ke *client*
4. *Client* menghitung *shared key*, dan mengirimkan kunci publik ke *server*
5. *Server* menghitung *shared key* dan menyimpannya di struktur data lokal

Hal yang sama juga akan dilakukan apabila ada banyak klien yang terkoneksi dengan *server*. Setelah pertukaran kunci antara klien dan *server* selesai, proses pengiriman dan *broadcasting* dapat berjalan dengan sewajarnya. Gambar 5 merupakan contoh proses pertukaran kunci antara *server* dan *client*.



Gambar 4. Proses pembangkitan kunci



Gambar 5. Proses Pertukaran Kunci

C. Rancangan Alur Kerja Program

Setelah membahas proses pertukaran kunci, pada subbab ini akan dijelaskan mengenai proses pengiriman dan *broadcasting* pesan dari klien ke *server* dan dari *server* ke klien. Proses dimulai ketika klien terhubung ke *server* dan melakukan

pertukaran kunci. *Server* kini memiliki *list* yang berisi informasi berupa *client* dan *shared key*. Ketika seorang klien mengetik pesan dan mengirimkannya ke *server*, algoritma enkripsi *Berez Cipher* akan berjalan di klien dan melakukan enkripsi terhadap pesan tersebut menggunakan *shared key*. Pesan yang terenkripsi tersebut lalu akan dikirim dan diterima oleh *server*.

Server ketika menerima pesan terenkripsi, pertama akan mencoba untuk melakukan dekripsi terlebih dahulu. Tahap pertama yang dilakukan adalah dengan mencari *shared key* klien yang bersangkutan pada *list*. Setelah ditemukan, *server* akan melakukan dekripsi pesan tersebut. *Server* sekarang harus meneruskan pesan tersebut ke klien yang lain, oleh karena itu *server* akan melakukan traversal pada *list* klien, melakukan enkripsi berdasarkan *shared key* klien tersebut dan mengirimkannya. Gambar 6 merupakan contoh proses pengiriman pesan.

```
send_encrypted Point(7147,7719) bytearray(b'\xd6<o\x81?\xe5d\xd2Y\xOc\xf5\x9c\xb
c\xe5\xfe\xc5;a\xd50qqv5\x17\xOb\xff\xb1\x04!\xd6f\xdd\xfaW\xceO\xb2J\xa4\x85\xc
a\xfb\x01\xc9\xc1%\xa53\xa4\xc2-\xa8<:ZVe\xd6\x92\xc7Y}\xO3')
recv_encrypted Point(7147,7719) b'\xfa\xd1\x1d\xda4B\xd4\xOe\x8fk\xc4ao\x90\x97\
x03PU%\xbd1\x12\x02k!QV\x1f\xe2v\x87\x95'
send_encrypted Point(7147,7719) bytearray(b'\xeO\n\xbe\xdd\xa1\xaa\x84V\xf1\xa1q
\xf8\xc9\\T\xeeK\xdO\xc8=O)\xb2w\xa1\xad\xd4\x19\x81\xa2H\xfb\xe6\xeb\xcf\x8c\x1
fzx\xO8.E\xa6\xf5?ª\xbfWs\xbe\xc6\x1cr\xeb\xd3\r\xd1\x1f\xec\xfe\x8c\xadWP')
send_encrypted Point(7147,7719) bytearray(b'\x99t^`\xc1\xd2\xO7\t\x1e\xff\xec&\x
ae\x8b\x82\xb9\xcfn\x86\x8f\x8e\xb3\xO3\x93m\x85.\x1f\xa3\xec\x8b\xbc')
recv_encrypted Point(7147,7719) b'6\xa4\xf9Xb\xfbC\xc8\x93\xc7\xc6\xf8\xde\xb15\
xbcQ\x80&/\xea[4\xeba!\xeO\xeb_\xf9\xe9Q'
send_encrypted Point(7147,7719) bytearray(b'\xc7\xad\xb3\x98\xaf4ª]8>\xdd\xO6l\x
9d\xOf\xcdQX\x9c\x87\r\xb7\xb3^N&\xb7\x98\xfdnjV')
```

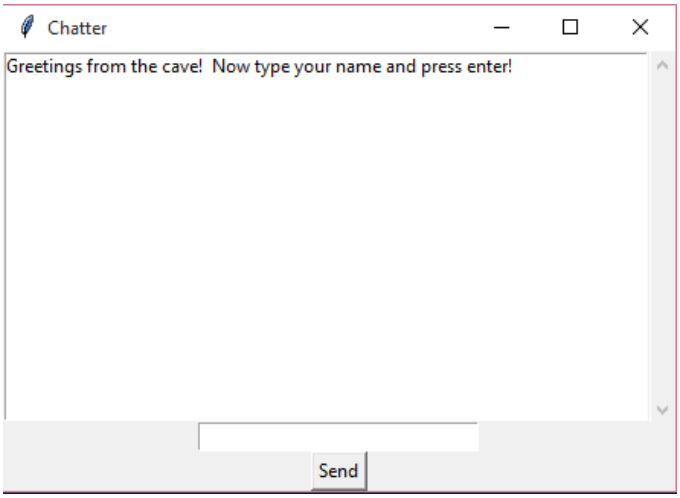
Gambar 6. Proses Pengiriman Pesan

IV. HASIL DAN ANALISIS

Pada bab ini akan ditunjukkan hasil yang diperoleh dan dilakukan analisis terhadap solusi yang dibangun. Hasil implementasi berisikan tampilan dan detil yang terjadi berkaitan dengan aplikasi yang dibuat. Sedangkan analisis berfokus pada performa *Curve25519* dibandingkan dengan berbagai jenis kurva lainnya yang sering digunakan.

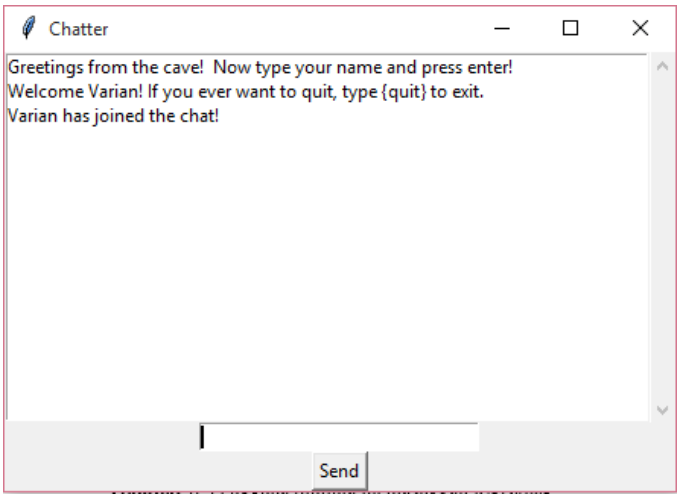
A. Hasil Implementasi

Implementasi menghasilkan sebuah aplikasi *chatting room* menggunakan *socket* yang diamankan dengan menggunakan algoritma Berez Cipher dan algoritma pertukaran kunci menggunakan *Curve25519*. Implementasi dilakukan pada bahasa Python dan menggunakan bantuan *library* bernama *cryptography* dan TkInter (GUI). Pesan yang dapat dikirim hanya berupa teks. Aplikasi yang dibuat merupakan aplikasi *chatting room* yang bebas, artinya tidak perlu mendaftar dan membuat akun. Pada saat pertama kali terhubung, pengguna akan ditanya tentang *username* mereka. Gambar 7 merupakan tampilan yang dimaksud.



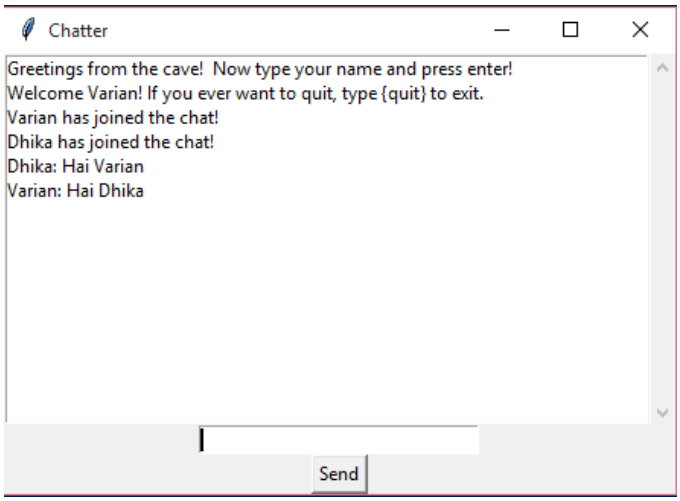
Gambar 7. Pengguna diminta memasukkan *username*

Setelah memasukkan *username*, selanjutnya, *server* akan mengirimkan sambutan dan klien dapat segera melakukan aktivitas *chatting*. Gambar 8 menunjukkan proses yang dimaksud.



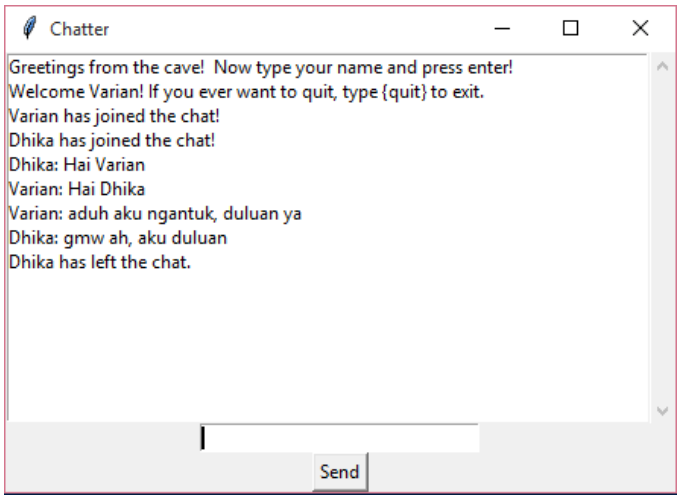
Gambar 8. Respon dari *server* ketika pengguna memasukkan *username*

Gambar 9 menunjukkan aktivitas *chatting* yang terjadi pada lebih dari 1 klien.



Gambar 9. Lebih dari 1 klien terhubung ke *server*

Apabila pengguna ingin keluar, maka harus mengetikkan perintah "{quit}". *Server* lalu akan memutus hubungan dan akan mengumumkan ke *chat room*. Gambar 10 merupakan contoh tampilan yang dimaksud.



Gambar 10. Tampilan ketika ada *client* yang keluar

B. Pengujian dan Analisis

Analisis dan pengujian dilakukan untuk membandingkan kinerja *Curve25519* dibandingkan dengan berbagai kurva lainnya yang sering digunakan pada skema pertukaran kunci. Aspek yang dibandingkan adalah waktu eksekusi total (pembangkitan sampai dengan proses pertukaran kunci) dan total memori yang digunakan. Pengujian performa dilakukan pada mesin dengan spesifikasi sebagai berikut:

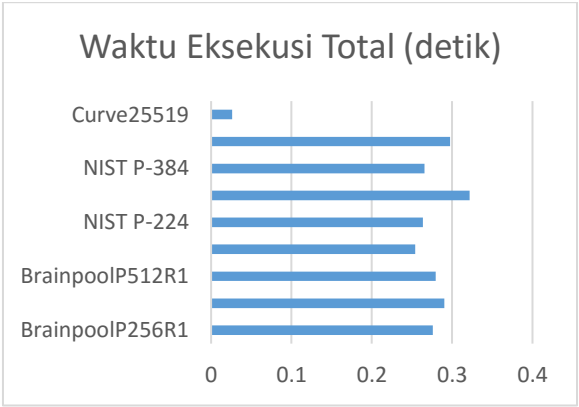
Processor	AMD APU A8-5550M Processor
Operating System	Windows 10 Enterprise
Chipset	AMD A76 FCH
Memory	DDR3L 1600 MHz SDRAM, 8 GB
Storage	2.5'' 9.5mm SATA 1 TB 5400 RPM

1. Analisis Waktu
   Pengujian dilakukan untuk membandingkan kinerja *Curve25519* dalam hal waktu eksekusi dibandingkan dengan jenis kurva lain.

Tabel 1. Perbandingan waktu eksekusi

Jenis Kurva	Waktu Eksekusi Total (detik)
BrainpoolP256R1	0.27595
BrainpoolP384R1	0.29019
BrainpoolP512R1	0.27971
NIST P-192	0.25406
NIST P-224	0.26364
NIST P-256	0.32188
NIST P-384	0.26577
NIST P-521	0.29733
Curve25519	0.02604

Tabel 1 merupakan hasil perbandingan *Curve25519* terhadap berbagai kurva lainnya yang cukup terkenal digunakan. Waktu eksekusi yang dimaksud adalah waktu yang dibutuhkan dari proses pembangkitan kunci privat dank unci publik sampai dengan proses pertukaran dan penurunan *shared key*. Terlihat bahwa performa *Curve25519* mengalahkan kurva lainnya. Selisih waktu *Curve25519* pun cukup signifikan bila dibandingkan dengan kurva lainnya. Gambar 11 merangkum Tabel 1 menjadi bentuk grafik yang mudah dilihat.



Gambar 11. Grafik waktu eksekusi

Curve25519, sesuai namanya adalah kurva yang berada pada GF(2^{255}-19). Kita dapat memprediksi jumlah waktu yang diperlukan oleh sebuah proses perhitungan dengan menghitung jumlah operasi yang dibutuhkan. Eksponen pada kurva 25519 adalah 255 bit, setiap pelelaran akan membutuhkan 3.6 operasi

dan setiap perkalian dengan faktor k akan membutuhkan 4.6 operasi. Sehingga maksimal akan terjadi 255*(3.6 + 4.6) = 2091 operasi saja. Sedangkan bila kita bandingkan dengan kurva lainnya, rata – rata memiliki eksponen 256 bit dan berada pada GF yang lebih rumit (misal: $2^{256} - 2^{224} + 2^{192} + 2^{96} – 1$).

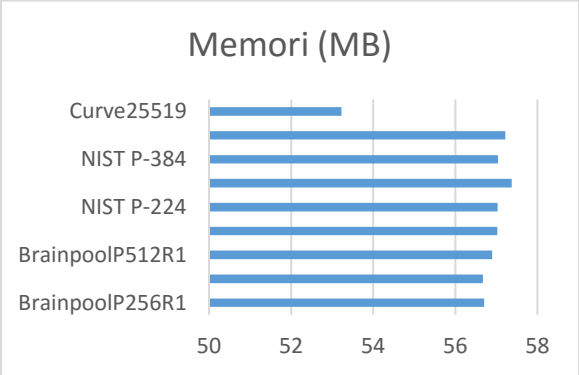
2. Analisis Memori

Pengujian pertama akan dilakukan terlebih dahulu untuk mengetahui jumlah memori yang digunakan oleh masing – masing kurva. Untuk proses pengukuran, digunakan bantuan *library* bernama memory_profiler.

Tabel 2. Perbandingan memori yang digunakan

Jenis Kurva	Memori (MB)
BrainpoolP256R1	56.700
BrainpoolP384R1	56.670
BrainpoolP512R1	56.898
NIST P-192	57.022
NIST P-224	57.027
NIST P-256	57.375
NIST P-384	57.039
NIST P-521	57.215
Curve25519	53.223

Dari Tabel 2, terlihat pula bahwa *Curve25519* juga hemat dalam sisi pemakaian memori. Namun perbedaan ini tidaklah signifikan. Penyebab hematnya memori yang digunakan, kemungkinan juga disebabkan karena sederhananya persamaan kurva dan jumlah eksponen. Gambar 12 merupakan rangkuman isi Tabel 2 dalam bentuk grafik yang mudah dilihat.



Gambar 12. Grafik perbandingan memori

KESIMPULAN

Pada makalah ini penulis telah berhasil membuat aplikasi *chat room* sederhana dengan menggunakan *socket* pada bahasa pemrograman Python. Aplikasi ini diamankan dengan menggunakan algoritma enkripsi *BeRez Cipher*. Skema pertukaran kunci dilakukan dengan menggunakan skema ECDH. Kurva yang digunakan pada ECDH adalah Curve25519 yang merupakan kurva dengan kinerja yang cukup baik diantara kurva lainnya. Pengujian membuktikan, *Curve25519* lebih cepat dalam hal waktu eksekusi dan lebih irit memori.

UCAPAN TERIMA KASIH

Penulis ingin menyampaikan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T selaku dosen pengampu mata kuliah IF4020 Kriptografi Semester II / 2017 – 2018. Peran beliau dalam makalah ini sangat besar dan beliau yang memberikan kritik, saran dan usulan kepada penulis untuk mencoba melakukan eksplorasi topik ini.

Penulis juga berterima kasih kepada rekan-rekan sesama peserta IF4020 Kriptografi. Terima kasih karena telah memberikan semangat dan bantuan dalam proses pengerjaan makalah ini.

REFERENSI

[1] AndreaCorbellini, ‘Elliptic Curve Cryptography: ECDH and ECDSA’. [Online]. Available: http://andrea.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/. [Accessed May 18, 2018].

[2] Cryptography Documentation, 2018. [Online]. Available : https://cryptography.io/en/. [Accessed May 15, 2018].

[3] CryptoStackExchange, ‘EC Curve Selection’. [Online]. Available: https://crypto.stackexchange.com/questions/52883/ec-curve-selection. [Accessed May 16, 2018].

[4] Github, ‘eccsnacks’. [Online]. Available: https://github.com/nnathan/eccsnacks. [Accessed May 17, 2018].

[5] GlobalSign, ‘Elliptic Curve Cryptography’. [Online]. Available: https://www.globalsign.com/en/blog/elliptic-curve-cryptography/. [Accessed May 16, 2018].

[6] Lecture Slides IF4020 Kriptografi

[7] SafeCurves, ‘Introduction’. [Online]. Available: https://safecurves.cr.yp.to/. [Accessed May 16, 2018].

PERNYATAAN

Saya yang bertandatangan dibawah menyatakan bahwa makalah ini adalah hasil pekerjaan saya sendiri. Bukan hasil terjemahan dari pekerjaan orang lain dan bukan bentuk plagiarisme.

Bandung, 17 Mei 2018

Varian Caesar | 13514041