Perbandingan Secure Socket Handshake dengan ECDH, Asymmetric Cipher dan Metode Massey-Omura

Jauhar Arifin
13515049
Teknik Informatika
Institut Teknologi Bandung
jauhararifin10@gmail.com

Abstrak

Pada paper ini dijelaskan tentang teknik handshake pada secure socket layer menggunakan metode ECDH, asymmetric cipher dan massey omura. Ketiga metode tersebut akan dibandingkan dari segi kinerja, keamanan dan kemudahan. Perbandingan kinerja dilakukan dengan mensimulasikan proses handshake menggunakan bahasa pemrograman python.

Kata kunci: ECDH, elliptic curve, diffie helman, asymmetric cipher, massey omura, secure socket layer.

1. Pendahuluan

Proses pertukaran informasi dari suatu agent ke agent lainnya menjadi sangat pending pada era berkembangnya teknologi seperti sekarang. Pertukaran informasi ini terjadi dengan sangat cepat setiap saat. Seiring dengan berkembangnya teknologipun jumlah informasi yang dipertukarkan semakin besar. Informasi yang dipertukarkan ini sering kali berisi konten yang rahasia seperti credential dari seseorang, nomor kartu kredit dan lain sebagainya. Untuk menjaga kerahasiaan informasi yang saling dipertukarkan, berbagai teknik kriptografi diterapkan sehingga informasi yang dikirim hanya dapat dibaca oleh penerima yang berwenang.

Pada jaringan internet, setiap komputer berkomunikasi satu dengan yang lainnya dengan cara mengirim *byte-byte* pesan. Byte pesan ini dikirim dengan suatu protokol tertentu seperti TCP, UDP dan lain sebagainya. Protokol pengiriman pesan tersebut umumnya mengirimkan pesan dalam bentuk *plaintext* dan dapat disadap oleh pihak lain. Untuk menghindari hal ini, tentunya dibutuhkan suatu protokol baru yang memiliki fitur penyandian pesan sehingga kerahasiaan pesan dapat terjaga.

Komputer yang menggunakan jaringan internet umumnya memiliki sistem operasi yang menyediakan *interface* untuk melakukan komunikasi dengan komputer lain. *Interface* yang paling sering digunakan adalah *socket*. *Socket* merupakan API yang disediakan oleh sistem operasi agar sebuah proses yang berjalan pada sistem operasi tersebut dapat berkomunikasi dengan proses lain. Komunikasi antar proses dapat terjadi pada satu buah komputer yang sama ataupun pada komputer yang berbeda. Socket tidak menyediakan fitur penyandian pada pesan sehingga dianggap tidak aman. Oleh karena itu, dibuat API baru yang lebih aman yaitu *secure socket.* Pada secure socket, pesan yang dikirimkan, terlebih dahulu disandikan di sisi pengirim baru dikirim melalui jaringan. Pada sisi penerima, pesan di*decrypt* terlebih dahulu sebelum dibaca oleh proses.

Untuk melakukan *encrypt* dan *decrypt* pesan, umumnya *secure socket* menggunakan algoritma enkripsi dan dekripsi dengan kunci simetris karena jauh lebih cepat dibandingkan dengan algoritma kunci asimetris. Untuk menggunakan algoritma kunci simetris, diperlukan kunci yang sama antara pengirim dan penerima pesan. Proses pertukaran kunci ini diperlukan sebelum proses pengiriman pesan dilakukan. Kunci tidak mungkin dikirimkan begitu saja dalam bentuk *plaintext* karena hal ini dapat menybabkan *attacker* dapat membaca pesan-pesan yang dikirimkan melalui jaringan karena *attacker* bisa menyadap kunci yang saling dipertukarkan. Oleh karena itu, diperlukan algoritma kriptografi untuk mengirimkan kunci ini secara aman. Proses pembentukan kunci bersama ini dinamakan proses *handshake* pada *secure socket*.

2. Secure Socket

Istilah *socket* berasal dari *electricity/phone socket* dimana *socket* menjadi interface yang dicolokkan satu sama lain melalui jaringan[1]. Socket dalam konteks jaringan internet dapat didefinisikan sebagai suatu interface yang menghubungkan atara dua buah proses sehingga setiap proses dapat mengirimkan pesan ke proses lain dan dapat menerima pesan dari proses lain melalui jaringan. Socket sudah menjadi interface di berbagai sistem operasi termasuk Windows, Linux dan Unix.

Operasi umum yang disediakan socket adalah:

- listen: dengan operasi ini, proses akan membuka sebuah port pada sistem operasi dan menunggu proses lain terkoneksi dengan proses ini.
- Connect: dengan operasi ini, proses akan melakukan koneksi ke proses lain yang sedang menunggu koneksi. Jika berhasil, kedua proses akan membentuk jalur penghubung yang dapat digunakan untuk melakukan pengiriman pesan.
- Send: dengan operasi ini, proses dapat mengirimkan pesan dalam bentuk kumpulan *byte* melalui jalur yang sudah dibentuk pada saat melakukan koneksi.
- Recv: dengan operasi ini, proses dapat menerima sejumlah *byte* pesan yang dikirimkan untuknya.

Pesan yang dikirimkan melalui operasi send dan recv merupakan pesan dalam bentuk plaintext. Hal ini menyebabkan pesan yang dikirim dapat dibaca oleh orang lain. Oleh karena itu, interface ini dikembangkan menjadi interface baru yang lebih aman dan dinamakan secure socket. Secure socket melakukan proses enkripsi pada pesan ketika pesan dikirimkan melalui operasi send. Penerima yang memanggil operasi recv akan melakukan dekripsi pesan sebelum dibaca oleh proses.

Secure socket umumnya menggunakan algoritma kriptografi kunci simetris karena kinerjanya yang lebih cepat dan tidak membutuhkan memori yang besar. Pertukaran pesan antar proses harus terjadi dengan cepat sehingga sebisa mungkin *overhead* dalam melakukan enkripsi dan dekripsi pesan diperkecil. Untuk melakukan enkripsi dan dekripsi antara pengirim dan penerima, dibutuhkan sebuah kunci yang sama antara pengirim dan penerima. Oleh karena itu, perlu adanya mekanisme pembentukan kunci bersama pada dua proses ini.

Pada secure socket interface, pembentukan kunci bersama antara dua buah proses pada jaringan disebut handshake. Tahap handshake ini terjadi ketika sebuah proses menjalankan operasi connect. Ketika operasi connect dijalankan dan sudah terbentuk sebuah jalur, langkah selanjutnya yang dilakukan oleh interface ini adalah melakukan handshake. Ketika tahap handshake sudah berhasil dijalankan dan kunci bersama sudah terbentuk, maka kedua buah proses baru dapat menjalankan operasi send dan recv.

3. Kriptografi Kunci Asimetris

Sistem kriptografi kunci asimetris atau terkadang disebut sistem kriptografi kunci publik pertama kali diusulkan oleh Diffie dan Hellman pada tahun 1976[2]. Kriptografi kunci asimetris melakukan enkripsi dan dekripsi dengan kunci yang berbeda. Untuk melakukan enkripsi pesan, diperlukan kunci publik. Sedangkan untuk melakukan dekripsi pesan, diperlukan kunci privat. Kunci publik dapat disebarluaskan ke semua orang, sedangkan kunci privat harus disimpan dan tidak boleh diketahui oleh pihak lain.

Kunci privat dan kunci publik dapat dibangkitkan dengan berbagai cara. Berbagai algoritma enkripsi telah ditemukan untuk melakukan pembangkitan kunci ini. Untuk menciptakan pasangan kunci yang aman, sebuah kunci privat harus sulit untuk ditemukan jika seorang *attacker* hanya memiliki informasi kunci publik. Beberapa algoritma memanfaatkan persoalan matematika yang tergolong NP problem seperti integer factorization, discrete logarithm, dan subset sum. Sulitnya persoalan tersebut mengakibatkan attacker tidak dapat menemukan kunci privat seseorang jika hanya memiliki informasi kunci publiknya.

Dalam melakukan pengiriman pesan, seorang pengirim pesan terlebih dahuli melakukan enkripsi terhadap pesan yang ingin dikirim menggunakan kunci publik penerima pesan. Seorang attacker yang dapat menyadap pesan tersebut tidak dapat mengetahui isinya karena untuk membuka isi pesan tersebut dibutuhkan kunci privat dari penerima pesan yang hanya dimiliki oleh penerima pesan. Penerima pesan akan menerima pesan ini dengan cara melakukan proses dekripsi pada pesan tersebut menggunakan kunci privat yang telah dibangkitkannya.

Beberapa algoritma kunci asimetris telah ditemukan. Beberapa algoritma yang cukup populer adalah RSA dan Elgamal. Algoritma RSA mmemanfaatkan persoalan matematika yang cukup sulit yaitu prime factorization. Persoalan prime factorization akan menjadi sangat sulit ketika bingan yang difaktorkan menjadi sangat besar. Algoritma Elgamal memanfaatkan persoalan matematika discrete logarithm yang juga sulit untuk dipercahkan ketika

bilangan menjadi sangat besar. Umumnya algoritma kriptografi kunci asimetris memerlukan bilangan yang besar untuk melakukan proses enkripsi dan dekripsi untuk menjaga keamanannya. Semakin besar bilangannya, semakin sulit attacker untuk memecahkan pesan yang telah dienkripsi. Oleh karena itu, algoritma ini jarang digunakan untuk melakukan pengiriman pesan yang berukuran besar karena membutuhkan waktu yang lama dan memori yang banyak.

Dengan menggunakan elliptic curve cryptography proses enkripsi dan dekripsi dapat dipercepat dengan mengurangi panjang bit kunci. Hal ini menyebabkan bilangan yang digunakan sebagai kunci publik dan privat menjadi lebih kecil. Dengan menggunakan bilangan yang lebih kecil, penggunaan CPU dapat lebih ditekan, begitu juga dengan memorinya. Meskipun panjang kuncinya menjadi lebih kecil, elliptic curve cryptography tetap menjaga keamanan dari pesan yang dikirim.

4. Metode Diffie Helman

Untuk melakukan komunikasi antara dua buah agent dengan menggunakan secure socket, diperlukan adanya kunci yang sama untuk melakukan enkripsi dan dekripsi pesan. Kunci ini harus dimiliki oleh kedua belah pihak. Diperlukan suatu mekanisme sehingga kedua agent dapat saling menyetujui satu buah nilai kunci tertentu. Proses menentukan kunci ini disebut key exchange dan berlangsung pada tahap handshake pada secure socket.

Metode Diffie-Helman merupakan algoritma yang dapat digunakan untuk melakukan pertukaran kunci antara dua buah agent sehingga kunci yang dipertukarkan tidak perlu dikirimkan melalui jaringan komunikasi sehingga attacker tidak dapat mengetahui kunci yang digunakan kedua agent meskipun dapat mengetahui pesan yang dikirimkan melalui jaringan. Metode ini menggunakan konsep yang mirip seperti algoritma Elgamal yaitu menggunakan perpangkatan modular pada bilangan bulat.

Misalkan dua buah agent bernama Alica dan Bob ingin melakukan handshake menggunakan algoritma Diffie-Helman. Langkah-langkah yang dilakukan adalah sebagai berikut[4]:

1. Pertama-tama Alice dan Bob sudah sepakat dengan dua buah nilai yaitu p dan g. P merupakan sebuah bilangan prima yang besar, sedangkan g merupakan bilangan bulat positif yang kurang dari p. Bilangan p akan menjadi modulus pada semua operasi yang dilakukan pada algoritma Diffie-Helman. Bilangan g disebut basis dan akan menjadi basis dalam perpangkatan yang akan dilakukan di tahap berikutnya. Bilangan p dan g ini tidak bersifat rahasia dan dapat diketahui oleh semua orang termasuk attacker.
2. Alice membangun sebuah bilangan bulat positif a secara acak. Bilangan ini bersifat rahasia dan hanya dapat diketahui oleh Alice. Bilangan bulat ini sebaiknya merupakan bilangan yang besar agar sulit ditebak oleh attacker.
3. Sama seperti Alice, Bob juga membangun bilangan bulat b secara acak yang bersifat rahasia dan hanya diketahui oleh Bob.
4. Alice menghitung nilai $x=g^a \bmod p$. X merupakan nilai yang akan dikirimkan melalui jaringan dan bersifat publik. Perlu diketahui meskipun nilai x dapat diketahui oleh attacker, attacker tidak dapat menemukan nilai a. Hal ini disebabkan karena untuk menemukan nilai a, attacker perlu memecahkan persoalan discrete logarithm yang sangat sulit untuk dipecahkan.
5. Alice kemudian mengirimkan nilai x kepada bob melalui jaringan yang insecure.
6. Bob kemudian menerima nilai x dan menghitung nilai $k=x^b \bmod p$. Nilai k ini yang akan menjadi kunci bersama Alice dan Bob.
7. Bob kemudian menghitung nilai $y=g^b \bmod p$. Nilai ini mirip seperti nilai x yang dihitung oleh Alice.
8. Bob mengirimkan nilai y kepada Alice.
9. Alice menerima nilai y dan menghitung nilai $k=y^a \bmod p$. K merupakan kunci yang akan digunakan bersama-sama oleh Alice dan Bob. Perlu diketahui bahwa nilai k yang dimiliki Alice sama dengan k yang dimiliki Bob. $k= y^a \bmod p = (g^b)^a \bmod p = (g^a)^b \bmod p = x^b \bmod p$.

Nilai k yang telah disetujui oleh Alice dan Bob akan digunakan untuk melakukan enkripsi dan dekripsi pada pesan-pesan yang akan dikirimkan oleh Alice dan Bob selanjutnya. Enkripsi dan dekripsi pesan dapat dilakukan menggunakan algoritma kriptografi kunci simetrik seperti AES dan DES.

5. Elliptic Curve Diffie Hellman

Untuk melalukan pertukaran kunci menggunakan algoritma Diffie-Hellman secara aman diperlukan nilai kunci yang besar. Nilai kunci yang besar akan menyulitkan attacker melakukan bruteforce attack.

Penggunaan kunci yang besar tentunya memerlukan komputasi dan memori yang besar pula. Komputasi ini dapat menjadi overhead yang besar dalam pengiriman pesan.

Dengan menggunakan elliptic curve cryptography, nilai kunci yang dipertukarkan menggunakan algoritma Diffie-Hellman dapat diperkecil. Kunci yang lebih pendek akan lebih mudah dan cepat dihitung. Dengan teknik elliptic curve cryptography, keamanan dan kerahasiaan dari kunci akan tetap terjaga meskipun panjang kuncinya menjadi lebih pendek. Hal ini disebabkan karena operasi-operasi pada elliptic curve cryptography lebih sukar dilakukan dibanding dengan operasi biasa.

Tahapan-tahapan untuk melakukan pertukaran kunci dengan Elliptic Curve Diffie-Hellman sebenarnya mirip dengan algoritma Diffie-Hellman biasa. Akan tetapi seluruh operasinya dilakukan di atas kurva eliptik. Berikut langkah-langkah yang dilakukan Alice dan Bob untuk melakukan pertukaran kunci dengan menggunakan algoritma Elliptic Curve Diffie-Hellman:

1. Alice dan Bob setuju terhadap suatu kurva eliptik, nilai p sebagai modulo dan nilai G sebagai basis.
2. Alice membangun kunci privat a secara acak.
3. Bob membangun kunci privat b secara acak.
4. Alice menghitung nilai x=aG mod p. Nilai a dikali G adalah perkalian titik dengan skalar pada kurva eliptik yang telah disepakati.
5. Alice mengirimkan nilai x kepada Bob.
6. Bob menghitung nilai k=bx mod p. Dengan bx dihitung dengan melakukan perkalian titik dengan skalar pada kurva eliptik yang telah disepakati.
7. Bob menghitung nilai y=bG mod p.
8. Bob mengirim y kepada Alice.
9. Alice menghitung nilai k=ay mod p.

6. Metode Massey-Omura

Massey-Omura merupakan teknik kriptografi pada pesan secara tanpa menggunakan kunci publik. Dengan metode ini, pesan perlu dienkripsi dengan menggunakan dua buah kunci, yaitu kunci pengirim dan kunci penerima. Pengirim dan penerima tidak perlu membangkitkan kunci publik karena pesan dienkripsi dengan menggunakan kunci privat masing-masing. Secara singkat, metode massey-omura adalah sebagai berikut:

1. Alice yang ingin mengirimkan pesan kepada Bob membangkitkan kunci privat untuk enkripsi dan dekripsi pesan.
2. Bob juga melakukan pembangkitan kunci privat untuk enkripsi dan dekripsi pesan.
3. Alice mengenkirpsi pesan yang ingin dikirim menggunakan kunci privat-nya. Pesan terenkripsi tersebut kemudian dikirimkan kepada Bob.
4. Bob menerima pesan terenkripsi dari Alice. Pada tahap ini, Bob tidak dapat membuka pesan yang dienkripsi oleh Alice dikarenakan hanya Alice yang memiliki kunci untuk membuka pesan tersebut. Bob kemudian mengenkripsi pesan yang didapatkannya. Pada tahap ini, pesan tersebut sudah dienkripsi dua kali, yaitu menggunakan kunci privat Alice dan kunci privat Bob. Bob mengirimkan pesan tersebut ke Alice.
5. Alice menerima pesan yang sudah dienkripsi dengan kunci privat Alice dan kunci privat Bob. Pada tahap ini, Alice mendekripsi pesan dengan menggunakan kunci privatnya. Pesan kemudian berubah menjadi pesan yang hanya terenkripsi dengan kunci privat Bob. Alice kemudian mengirim pesan ini ke Bob.
6. Bob menerima pesan dari Alice yang terenkripsi dengan kunci privat Bob. Bob kemudian membuka pesan tersebut dengan menggunakan kunci privatnya. Hasilnya adalah plaintext pesan.

Proses enkripsi dan dekripsi pesan pada metode Massey-Omura tidak dapat dilakukan dengan sembarang algoritma kriptografi. Untuk melakukan pendekripsian pesan yang terkunci dengan dua buah kunci, perlu adanya mekanisme sehingga ketika pesan dibuka dengan salah satu kunci, pesan masih terenkripsi dengan kunci yang satunya. Untuk memeroleh hal itu, diperlukan algoritma kriptografi yang independen terhadap urutan penguncian dan pembukaan kunci.

Proses enkripsi pada algoritma Massey-Omura dilakukan dengan menggunakan perpangkatan modular. Sebuah bilangan bulat prima p perlu disepakati terlebih dahulu oleh kedua belah pihak. Enkripsi terhadap pesan m dilakukan dengan menghitung $c=m^a \bmod p$ dimana a merupakan kunci privat pengirim atau penerima. Dekripsi pesan c dilakukan dengan menghitung nilai $m=c^b \bmod p$ dimana $1 = a*b \bmod (p-1)$.

7. Handshake Dengan ECDH dan Kriptografi Kunci Asimetris

Algoritma pertukaran kunci ECDH dan algoritma kriptografi kunci asimetris dapat digunakan untuk melakukan handshake pada secure socket. Setiap agen yang akan melakukan komunikasi harus terlebih dahulu mengetahui kunci publik dari masing-masing agen. Dalam praktiknya, kunci publik dapat dikirimkan pada saat handshake juga, akan tetapi hal ini memerlukan pihak ketiga yang bertindak sebagai certificate authority yang berperan dalam memastikan kebenaran kunci publik yang dikirimkan kedua belah pihak. Hal tersebut diluar cakupan paper ini. Untuk kemudahan, diasumsikan setiap agen sudah mengetahui kunci publik dari agen lainnya.

Untuk melakukan pertukaran kunci dengan aman, kedua agen harus memastikan terlebih dahulu bahwa mereka melakukan komunikasi dengan agen yang benar. Hal ini dapat dicapai dengan mengirimkan challenge ke agen yang bersangkutan. Misalkan Alice ingin melakukan koneksi ke Bob. Ketika Alice menginisiasi handshake, Ia mengirimkan sebuah bilangan acak yang dienkripsi menggunakan algoritma kunci asimetris dengan kunci publik Bob. Jika Ia benar-benar melakukan koneksi dengan Bob, Bob kemudian akan melakukan dekripsi pada bilangan acak tersebut. Bilangan acak tersebut kemudian dienkripsi lagi menggunakan kunci publik Alice. Jika Alice menerima bilangan yang sama seperti yang Ia kirimkan di awal handshake, maka Alice yakin bahwa ia benar-benar sedang berbicara dengan Bob. Seorang attacker tidak mungkin dapat berpura-pura menjadi Bob karena hanya Bob yang dapat mendekripsi challenge yang diberikan oleh Alice.

Setelah yakin dengan agen yang diajak berkomunikasi, kedua agen kemudian dapat melakukan pertukaran kunci dengan algoritma Elliptic Curve Diffie-Hellman. Untuk meningkatkan integritas pesan, komunikasi dengan algoritma Diffie-Hellman dapat dilakukan dengan menggunakan algoritma kriptografi kunci asimetris. Proses handshake akan selesai setelah algoritma Diffie-Hellman berhasil dijalankan dan kunci bersama telah terbentuk.

8. Handshake Dengan Massey-Omura

Pertukaran kunci dengan menggunakan metode Massey-Omura sebenarnya cukup sederhana. Kunci bersama dapat dibangkitkan oleh salah satu agen, kemudian dikirimkan ke agen lainnya dengan menggunakan metode Massey-Omura. Dengan menggunakan metode ini, tidak dapat dilakukan challenge karena tidak ada kunci publik yang diketahui oleh masing-masing agen.

9. Analisis

Meskipun metode Massey-Omura terlihat lebih sederhana dan dapat selalu menjaga kerahasiaan dan integritas pesan tanpa memerlukan kunci publik, akan tetapi metode ini dapat menjadi sangat tidak aman. Metode Massey-Omura tidak memiliki mekanisme autentikasi sehingga agen tidak dapat memastikan apakah Ia sedang berkomunikasi dengan agen yang benar. Hal ini disebabkan, seluruh kunci dari agen bersifat privat sehingga tidak ada mekanisme untuk mengirimkan pesan ke agen tertentu saja. Hal ini dapat memberikan celah pada attacker untuk melakukan man in the middle attack. Berbeda dengan metode ECDH dan kriptografi kunci asimetris. Pada ECDH dan kirptografi kunci asimetris, terdapat tahap challenge menggunakan kriptografi kunci asimetris yang dapat memastikah bahwa agen yang diajak handshake adalah benar agen yang dimaksud.

Operasi pada Massey-Omura sama seperti pada metode ECDH dan Kriptografi Kunci Asimetris. Sehingga kinerja dari kedua algoritma ini hampir mirip dalam aspek kecepatan operasi tiap tahapnya. Secara lengkap, metode Massey-Omura memerlukan tiga kali operasi dan tiga kali pengiriman pesan, sedangkan pada metode ECDH dan kriptografi kunci asimetrik memerlukan empat kali. Hal ini dikarenakan pada kriptografi kunci asimetrik diperlukan mekanisme autentikasi sehingga memerlukan tambahan dua kali pengiriman data. Meskipun begitu, hal ini dapat dicegah dengan melakukan autentukasi bersamaan dengan pertukaran kunci. Jika hal ini dilakukan proses pengiriman pesan yang perlu dilakukan hanyalah dua kali. Umumnya operasi IO seperti mengirim pesan melalui jaringan sangat lambat dan jauh lebih lambat dibandingkan operasi-operasi kirptografi pada dua metode ini. Oleh karena itu, kecepatan dari kedua metode ini dibandingkan dari jumlah IO yang dibutuhkan. Sehingga pada aspek keceptan, metode ECDH dan kriptografi kunci asimetrik lebih cepat dibandingkan dengan metode Massey-Omura.

Dari segi kemudahan, metode Massey-Omura lebih mudah diimplementasikan dibanding dengan metode ECDH dan kriptografi kunci asimetrik. Hal ini disebabkan pada metode Masset-Omura operasi yang diperlukan hanyalah dua jenis yaitu enkripsi dan dekripsi pesan. Pada metode ECDH diperlukan dua

buah operasi yaitu menghitung nilai kunci yang dikirim melalui publik network dan menghitung nilai kunci bersama. Pada tahap autentikasi juga diperlukan algoritma kunci asimetris untuk mengenkripsi dan mendekripsi pesan. Sehingga pada metode ECDH dan kriptografi kunci asimetrik diperlukan empat buah operasi yang harus diimplementasikan.

10. Kesimpulan

Metode ECDH dengan kriptografi kunci asimetrik jauh lebih aman dari berbagai jenis serangan dibanding dengan metode Massey-Omura. Hal in disebabkan karena adanya mekanisme autentikasi pada metode ECDH dan kriptografi kunci asimetrik. Dari segi kecepatan, metode ECDH dan kriptografi kunci asimetrik juga lebih cepat meskipun tidak jauh perbedaannya. Selain itu metode ECDH dan kriptografi kunci asimetrik yang dioptimisasi memerlukan bandwidth yang lebih sedikit dibandingkan dengan metode Masset-Omura. Meskipun begitu, metode Massey-Omura lebih mudah diimpementasikan karena jumlah operasinya yang tidak terlalu beragam.

Dalam secure socket, sebaiknya digunakan metode ECDH dan kriptografi kunci asimetrik. Hal ini dikarenakan keamanan dan kinerja dari algoritma ini jauh lebih baik dibandingkan dengan metode Massey-Omura. Meskipun metode Massey-Omura lebih mudah diimplementasikan, akan tetapi kemudahan ini tidak sebanding dengan attack yang mungkin terjadi pada metode tersebut.

11. References

[1] Limi Kalita. “Socker Programming.” International Journal of Computer Science and Information Technologies, 2014, pp. 1–6.

[2] Rifki Sadikin. Kriptografi Untuk Keamanan Jaringan. Yogyakarta: Andi, 2012.

[3] Whitfield, and Martin E. “New Directions in Cryptography.” IEEE TRANSACTIONS ON INFORMATION THEORY, IT-22, 6 Nov. 1976, pp. 1–11.

[4] “A Review of the Diffie-Hellman Algorithm and Its Use in Secure Internet Protocols.” SANS Institute InfoSec Reading Room, pp. 1–9.

12. 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. 18 Mei 2018

Jauhar Arifin
(13515049)