SECURE REAL TIME PROTOCOL: SOLUSI ALTERNATIF PENGAMANAN *CHATTING*

Donny Seftyanto[1)]
[1]Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jatinangor, Sumedang, 45363
Telp : (022) 7798600, Fax: (022) 7798617
E-mail : donny.seftyanto@lemsaneg.go.id[1)]

Abstrak

Off The Record (OTR) merupakan protokol kriptografi yang digunakan untuk menjamin keamanan chatting pada banyak aplikasi, seperti Xabber. Tetapi terdapat kelemahan pada protokol ini, yaitu kegagalan otentikasi, penipuan, dan penyangkalan. Untuk memberikan solusi alternatif dalam pengamanan chatting, maka dirancang protokol bernama Secure Real Time (SRT). SRT terdiri dari tiga tahap, yaitu Trusted Public Key Distribution, Key Exchange with Digital Signature, dan Signed and Encrypted Message Transmission with Key Derivation Function. Tahapan tersebut diterapkan dengan algoritma ECDSA-384, ECDH-384, AES-256, dan SHA-384 pada aplikasi Xabber, sehingga memberikan kekuatan keamanan algoritma yang lebih tinggi dari OTR. Lalu berdasarkan hasil evaluasi yang meliputi uji keamanan komunikasi dan pembandingan performa aplikasi Xabber, diketahui bahwa protokol SRT dapat menjamin kerahasiaan, keutuhan, keotentikan, nir-penyangkalan, dan tahan replay attack terhadap data penting di ketiga tahap SRT. Sedangkan tingkat kecepatan dan kemudahan aplikasi Xabber dengan SRT relatif lebih tinggi dari aplikasi Xabber dengan OTR.

Kata kunci: *chatting, kriptografi, OTR, SRT.*

Abstract

Off The Record (OTR) is cryptographic protocol that is used to ensure the chatting safety in many applications, like Xabber. But there are weaknesses in this protocol, namely authentication failure, fraud, and repudiation. To provide alternative solution in securing chatting, then designed a protocol called Secure Real Time (SRT). SRT consists of three stages, namely The Trusted Public Key Distribution, Key Exchange with Digital Signature, and Signed and Encrypted Message Transmission with Key Derivation Function. These stages are implemented using algorithms ECDSA-384, ECDH-384, AES-256, and SHA-384 in Xabber thus providing security strength higher than OTR. Based on the results of evaluation that includes testing the security of communication and benchmarking performance of Xabber, it is known that SRT can guarantee confidentiality, integrity, authenticity, non-repudiation, and hold replay attack toward important data in the three stages of SRT. As for speed and ease relatively higher than the application Xabber with OTR.

Keywords: *chatting, kriptografi, OTR, SRT.*

1. PENDAHULUAN

Off The Record (OTR) *Messaging* merupakan protokol kriptografi pada layanan *chatting* yang terdiri dari tahap pertukaran kunci *Authenticated Key Exchange* (AKE) dan transmisi pesan *Exchanging Data*. Sedangkan algoritma yang digunakan meliputi Diffie-Hellman (DH) 1536 bit, *Advanced Encryption Standards* (AES) 128 bit, dan *Secure Hash Algorithm* (SHA) [9]. Dengan kemampuan yang diberikan, OTR Development Team menyatakan bahwa OTR telah diterapkan di berbagai aplikasi. Beberapa aplikasi yang menerapkan OTR *Messaging*, diantaranya Xabber, Adium, IM+, climm, mcabber, CenterIM, Kopete, Jitsi, ChatSecure, Jackline, Profanity, Kadu, Cryptocat, dsb [10]. Namun terdapat kelemahan pada OTR yang terbawa sejak dicetuskannya versi pertama hingga versi ketiga saat ini. Kelemahan OTR v.3 tersebut yaitu:

a. Kegagalan otentikasi [1] dan penipuan [3] karena *Man-In-The-Middle* (MITM) *attack* pada tahap pertukaran kunci AKE; dan

b. Penyangkalan terhadap pesan *chatting* yang ditransmisikan karena tidak ada penanda yang mengikat pesan dan mampu membuktikan bahwa pesan tersebut hanya dapat dikirimkan oleh sang pengirim [3].

Berdasarkan kelemahan tersebut, diperlukan solusi alternatif dalam pengamanan *chatting*. Langkah penulis adalah dengan merancang protokol *Secure Real Time* (SRT) yang tersusun dari tiga tahap. Tahap dari SRT meliputi *Trusted Public Key Distribution* untuk mengamankan distribusi kunci publik pengguna, *Key Exchange with Digital Signature* untuk mengamankan pertukaran kunci sesi, dan *Signed and Encrypted Message Transmission with Key Derivation Function* untuk mengamankan transmisi pesan *chatting*. Ketiga tahap tersebut diterapkan dengan algoritma *Elliptic Curve Diffie-Hellman* (ECDH) 384 bit, *Elliptic Curve Digita Signature Algorithm* (ECDSA) 384 bit, AES 256 bit, dan SHA 384 bit sehingga diharapkan memberi kekuatan keamanan algoritma yang lebih tinggi dari OTR.Untuk mengamati layanan keamanan dan performa yang diberikan, maka diterapkan SRT pada aplikasi Xabber yang sebelumnya telah menerapkan OTR. Layanan keamanan diamati melalui pemberian serangan penyadapan, modifikasi data, pesan palsu, penyangkalan, dan *replay attack* pada tahap pertukaran kunci dan transmisi pesan. Sedangkan performa aplikasi diamati melalui pembandingan kecepatan dan kemudahan penggunaan aplikasi.

Berdasarkan hal tersebut, diharapkan protokol SRT mampu meningkatkan keamanan dan performa layanan dibandingkan aplikasi sebelumnya. Keamanan yang dimaksud yaitu dengan menjamin kerahasiaan, keutuhan, keotentikan data, nir-penyangkalan, dan tahan *replay attack* pada setiap tahapnya, serta didukung dengan kekuatan keamanan algoritma yang lebih tinggi. Sedangkan performa yang diberikan memiliki kecepatan yang lebih tinggi dan penggunaan aplikasi yang lebih mudah dibandingkan aplikasi sebelumnya.

2. TINJAUAN PUSTAKA

Bagian ini berisi beberapa penjelasan dari berbagai sumber untuk mendukung penelitian ini.

2.1 *Algoritma Kriptografi*

Dalam setiap protokol kriptografi selalu terdapat algoritma kriptografi yang digunakan untuk menjalankan rangkaian langkah terurutnya. Pada protokol OTR terdapat algoritma DH 1536 bit, AES 128 bit, dan SHA, sedangkan protokol SRT menggunakan algoritma ECDH 384, ECDSA 384, AES 256, dan SHA 384. Algoritma DH, ECDH, dan ECDSA adalah algoritma kunci publik/asimetris, sedangkan AES adalah algoritma kunci simetris, dan SHA adalah fungsi hash.

Algoritma kunci publik dibagi menjadi 3 (tiga) jenis, yaitu *Integer Factorization Cryptography* (IFC) seperti RSA, *Finite Field Cryptography* (FFC) seperti DH, dan *Elliptic Curve Cryptography* (ECC) seperti ECDH dan ECDSA. Pada protokol SRT hanya menggunakan algoritma berbasis ECC saja, karena memungkinkan untuk memperoleh tingkat kekuatan keamanan yang relatif lebih tinggi namun menggunakan ukuran parameter kunci yang relatif lebih kecil. Kelebihan tersebut tergambar dari hasil perbandingan kekuatan keamanan algoritma pada Tabel 1 [8].

Tabel 1. Perbandingan Kekuatan Algortima (Sumber: NIST, 2012)

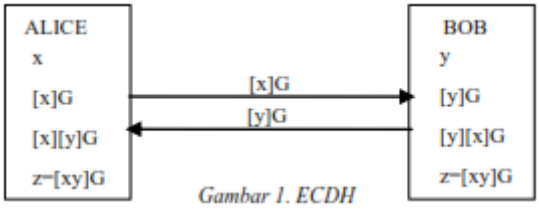
Kekuatan Keamanan	Algoritma Kriptografi (bit)			
	Algoritma Simetrik	FFC (cth: DH)	IFC (cth:RSA)	ECC (cth: ECDSA)
80	2TDEA	1024	1024	160
112	3TDEA	2048	2048	224
128	AES-128	3072	3072	256
192	AES-192	7680	7680	384
256	AES-256	15360	15360	512

Berdasarkan Tabel 1 didapati bahwa algoritma ECDH-384, ECDSA-384, dan AES-256 pada SRT memiliki kekuatan keamanan pada tingkat 256. Sedangkan algoritma DH-1536 dan AES-128 pada protokol OTR berada pada tingkat 128 ke bawah. Jelas bahwa seluruh algoritma pada SRT mampu memberikan kekuatan keamanan dengan tingkat yang lebih tinggi dari OTR. Dengan demikian penyerang semakin sulit untuk memecahkan sistem kriptografi pada layanan *chatting*. Berikut penjelasan singkat algoritma yang digunakan pada SRT.

2.1.1 *Elliptic Curve Cryptography* (ECC)

ECC merupakan metode kriptografi kunci publik yang menggunakan kurva ellips dengan semua variabel dan koefisiennya terbatas pada elemen dari suatu Galois Field. Kurva elips yang digunakan mengacu pada standar efisiensi kriptografi kurva elips dengan kode secp384r1 dari Certicom [2]. Kurva tersebut diterapkan baik pada algoritma ECDSA maupun ECDH di protokol SRT, berikut penjelasan kedua algoritma berbasis ECC tersebut. ECDSA adalah algoritma yang digunakan untuk memberikan tanda tangan digital yang bersifat probabilistik kepada pesan. Dengan demikian memberikan jaminan keutuhan, keotentikan data, dan nir-penyangkalan terhadap pesan terkait [4]. Kesesuaian penerapannya di penelitian ini, diuji terhadap contoh proses ECDSA kurva secp384r1 pada "*Suite B Implementer's Guide to FIPS 186-3 (ECDSA)*" [7]. Selain itu fungsi hash SHA 384 bit selalu digunakan sebelum pesan ditandatangani agar panjang pesan menjadi 384 bit.

ECDH adalah algoritma yang digunakan untuk pertukaran kunci sesi antara dua pihak secara aman, di mana masing-masing pihak memiliki dan menggunakan pasangan kunci publik kurva elips. ECDH berbeda dengan DH karena ECDH bekerja berdasarkan *Elliptic Curve Discrete Logarithm Problem* (ECDLP) bukan *Discrete Logarithm Problem* (DLP), namun memiliki langkah umum yang sama dengan DH (lihat Gambar 1). Alice membangkitkan kunci acak x dan mengalikannya dengan generator kurva elips G, lalu mengirimnya [x]G ke Bob. Setelah menerima, Bob membangkitkan kunci acak y dan mengalikannya hingga mengirimkan [y]G kepada Alice. Dengan demikian, kedua pihak dapat memperoleh nilai yang sama yaitu z atas hasil operasi [xy]G, yang mana z akan digunakan sebagai kunci sesi komunikasi terenkripsi [4].

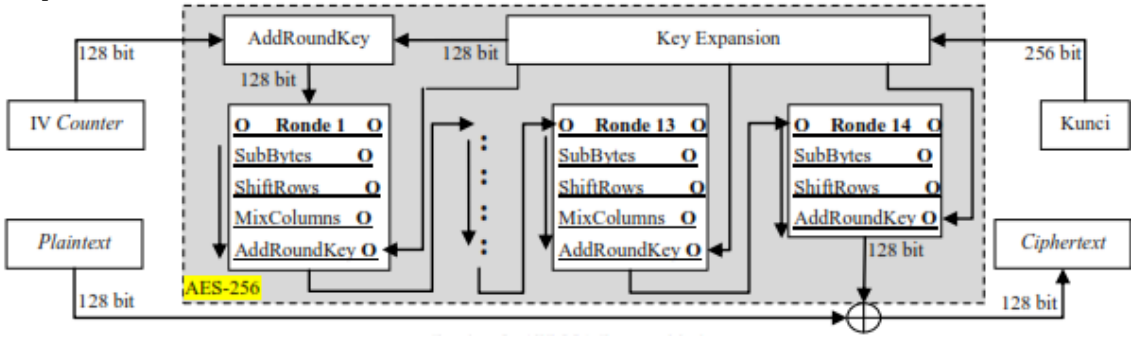


Gambar 1. ECDH

2.1.2 Advanced Encryption Standard (AES)

AES adalah algoritma kunci simetrik blok cipher yang memproses 128 bit blok data menggunakan kunci dengan panjang 128 bit, 192 bit, atau 256 bit untuk setiap iterasinya. Pada penerapannya di protokol SRT, AES menggunakan panjang kunci 256 bit (AES-256). Kunci tersebut akan diperpanjang dengan Key Expansion sehingga output-nya (128 bit) dapat digunakan di setiap ronde AES. Proses AES-256 diawali dengan Add Round Key lalu dilanjutkan dengan paket proses sebanyak 14 ronde. Paket proses tersebut terdiri dari 4 (empat) macam proses, yaitu SubBytes, Shiftrows, Mixcolumns, dan Add Round Key. Keempat proses berlaku pada setiap ronde AES, kecuali pada ronde terakhir yang hanya berlaku proses SubBytes, Shiftrows, dan Add Round Key saja [13].

Dalam penerapannya di SRT, AES-256 menggunakan mode operasi block cipher counter, sehingga input-nya adalah Kunci (256 bit) yang bersifat tetap dan IV Counter (128 bit) yang akan bertambah setiap iterasi AES (Lihat Gambar 2). Sedangkan output dari AES (128 bit) akan di -XOR-kan dengan blok Plaintext (128 bit) untuk setiap iterasinya sehingga didapatkan Ciphertext (128 bit). Proses tersebut akan beriterasi hingga seluruh blok Plaintext telah di-XOR dan menghasilkan Ciphertext. Melalui penggunaan mode operasi counter, proses penyandian AES relatif lebih cepat dibandingkan mode operasi lainnya, tidak ada error propagation, dan proses enkripsi/dekripsi dilakukan dengan satu cara [11]. Kesesuaian penerapan AES dengan mode counter akan diuji terhadap contoh proses pada "Recommendation for Block Cipher Modes of Operation" [6].



Gambar 2. AES 256 Counter Mode

2.2 Key Derivation Function (KDF)

KDF adalah fungsi untuk medapatkan kunci acak secara berkelanjutan menggunakan pembangkit bilangan acak, yang pada penelitian ini menggunakan fungsi hash SHA-384. Kunci acak yang diperoleh dari KDF digunakan sebagai kunci enkripsi simetrik (AES-256) beserta inisial vektornya (IV Counter-128 bit). Pada penerapan KDF, setidaknya membutuhkan input berupa shared secret value v dan counter c. Lalu menghitung nilai hash H(v,c) dari gabungan v dan c setiap kali melakukan enkripsi pesan. Sehingga diperoleh nilai v baru, nilai v=H(v,c) inilah yang digunakan untuk enkripsi pesan. Proses tersebut berulang secara terus menerus dengan nilai c yang selalu bertambah setiap kali proses dilakukan [5].

3. PEMBAHASAN

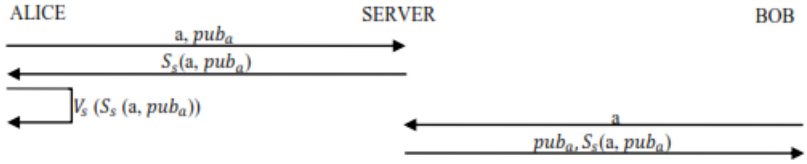
Berikut dijelaskan rancangan protokol SRT, implementasi, dan pengamatannya pada aplikasi Xabber.

3.1 Rancangan Protokol Secure Real Time (SRT)

Dalam menjamin keamanan komunikasi, SRT tersusun dari tiga tahap, yaitu

3.1.1 Trusted Public Key Distribution

Tahap ini bekerja untuk mendistribusikan kunci publik pengguna secara aman dan terpercaya karena menggunakan peran penyedia layanan chatting sebagai pihak ketiga terpercaya. Pihak ketiga bertugas menandatangani kunci publik dan mendistribusikannya kepada pengguna yang membutuhkan (lihat Gambar 3). Penandatangan kunci publik terjadi ketika pengguna (Alice) melakukan instalasi aplikasi Xabber pada pertama kali. Alice membangkitkan kunci privat $priv_a$ dan kunci public pub_a yang berpasangan. Kemudian mengirimkan pesan registrasi berupa identitas a dan kunci publik kepada pihak ketiga terpercaya (Server), sedangkan kunci privat tetap disimpan Alice. Lalu Server menandatangani keduanya S_s (a, pub_a) dan mengirimnya kepada Alice. Alice mengonfirmasinya V_s (S_s (a, pub_a)) jika valid maka proses berhasil, tetapi jika tidak valid maka proses gagal atau terdapat indikasi serangan.

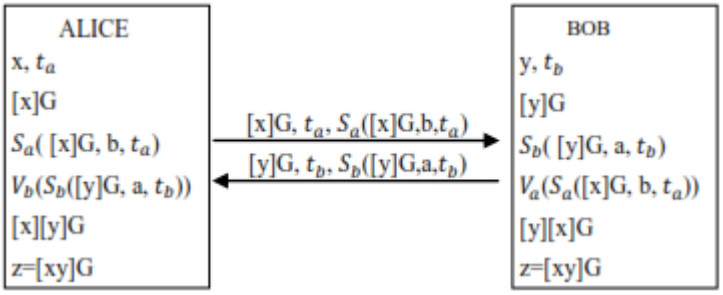


Gambar 3. Trusted Public Key Distribution

Untuk pendistribusian kunci publik, terjadi ketika ada pengguna lain (Bob) memulai chatting kepada seseorang (Alice) untuk pertama kalinya. Maka Bob mengirim a sebagai tanda permintaan kunci publik milik Alice kepada Server. Server akan memberikan kunci publik milik Alice yang tertandatangani S_s (a, pub_a) kepada Bob. Lalu Bob dapat memastikan kebenaran kunci publik milik Alice melalui verifikasi tanda tangan Server.

3.1.2 *Key Exchange with Digital Signature*

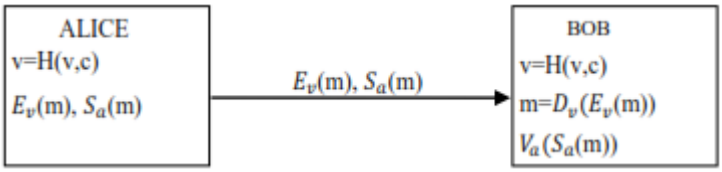
Key Exchange with Digital Signature bekerja dengan menerapkan teknik tanda tangan digital selama pertukaran kunci sesi, sehingga terhindar dari serangan MITM [12]. Dalam protokol SRT, tahap ini tersusun dari gabungan algoritma ECDSA-384 dan ECDH-384 (lihat Gambar 4).Ketika seorang pengguna (Alice) ingin memulai *chatting* dengan pengguna lain (Bob), Alice membangkitkan kunci acak x sebagai parameter awal pertukaran kunci dan Alice mencatat waktunya sebagai penanda waktu kunci. Kunci x dikalikan dengan generator kurva elips G, lalu ditandatangani bersama identitas Bob dan waktu. Pesan pertukaran kunci berupa hasil tanda tangan ([x]G,b,) bersama [x]G dan dikirim ke Bob. Setelah Bob menerimanya, ia melakukan hal yang sama seperti Alice hingga mengirimkan ([y]G,a,), [y]G, dan ke Alice. Dengan demikian, Alice dan Bob telah saling menerima pesan pertukaran kunci. Lalu mereka memverifikasi pesan pertukaran terhadap tanda tangan terkait, (([y]G,a,)) untuk Alice dan (([x]G,b,)) untuk Bob. Jika terbukti valid, maka kedua pihak dapat memperoleh nilai yang sama yaitu z atas hasil operasi [xy]G. Nilai z akan digunakan Alice dan Bob sebagai kunci sesi komunikasi terenkripsi padatahap selanjutnya.



Gambar 4. Key Exchange With Digital Signature

3.1.3 *Signed and Encrypted Message Transmission with Key Derivation Function*

Sesuai dengan namanya, tahap ini adalah gabungan dari penerapan teknik tanda tangan digital, teknik enkripsi, dan KDF untuk mengamankan transmisi pesan *chatting*. Algoritma yang digunakan meliputi ECDSA-384 untuk tanda tangan, AES 256 untuk enkripsi pesan, dan SHA-384 sebagai fungsi hash pada KDF (lihat Gambar 5). Tahap ini dapat dimulai ketika kedua pengguna (Alice dan Bob) telah memiliki kunci sesi bersama z, yang mana hasil dari tahap sebelumnya. Jika Alice ingin mengirim pesan m kepada Bob, maka nilai z akan dimasukan ke dalam KDF. Dalam KDF, z dipecah menjadi v dan c lalu dihitung nilai v baru dengan operasi hash H(v,c), di mana c selalu bertambah setiap kali pesan dikirmkan. Disamping itu, pesan ditandatangani oleh Alice (m) lalu dienkripsi (m) dengan v sebagai kuncinya. Pesan tertandatangani (m) dan pesan terenkripsi (m) dikirimkan kepada Bob.



Gambar 5. Signed and Encrypted Message Transmission with Key Derivation Function

Ketika Bob menerima pesan dari Alice, z dimasukan ke dalam KDF hingga mendapat nila v yang sama dengan Alice. Dengan demikian, Bob dapat mendekripsi pesan (m)) dengan kunci v, hingga didapatkan m dari proses tersebut. Lalu memverifikasi pesan m berdasarkan tanda tangan Alice, (m)). Jika valid, maka tahap ini berjalan sukses. Sebaliknya jika tidak, maka tahap ini gagal dan harus dimulai kembali dari awal.

3.2 Implmentasi

Implementasi protokol SRT dalam mengamankan layanan *chatting* dilakukan dengan tiga macam aplikasi, yaitu aplikasi Xabber di *smartphone* Android, aplikasi web XAMPP di *server* dan aplikasi *chatting* Ejabberd di *server*. Aplikasi Xabber dihadirkan dengan menerapkan SRT ke dalamnya, di mana sebelumnya Xabber menerapkan protokol OTR untuk pengamanan *chatting*. Aplikasi web XAMPP dibangun untuk memberikan layanan distribusi kunci seperti yang dijelaskan pada bagian 3.1.1 tentang *Trusted Public Key Distribution*. Sedangkan aplikasi *chatting* Ejabberd diinstalasi untuk memberikan layanan *chatting* dengan protokol komunikasi XMPP/jabberd. Selanjutnya di penelitian ini, aplikasi Xabber diinstalasi ke dalam *smartphone* Samsung Galaxy-W 1,4 GHz dan Sony Xperia-P 1 GHz. Sedangkan aplikasi web XAMPP dan aplikasi *chatting* Ejabberd diinstalasi ke dalam *virtual private server* (VPS) 2,4 GHz.

3.3 Pengamatan

Pengamatan protokol SRT pada aplikasi Xabber dilakukan dalam 2 bagian, yaitu:

3.3.1 Pengujian Keamanan Komunikasi

Pengujian dilakukan dengan memberikan serangan pada pesan yang ditransmisikan dalam 3 (tiga) tahap protokol SRT. Serangan tersebut, meliputi: penyadapan untuk memastikan data penting terjamin akan kerahasiaannya, modifikasi data untuk memastikan pesan terjamin keutuhannya, pengiriman pesan palsu untuk memastikan pesan terjamin keotentikannya, penyangkalan pesan untuk memastikan pesan terjamin nir- penyangkalan, dan *replay attack* untuk memastikan pesan penting yang dikirimkan tidak

dapat dikirimkan kembali [13]. Oleh karena itu, penyerang melakukan *capture* paket jaringan pengguna Xabber dengan *tool* Wireshark untuk melakukan penyadapan. Selain itu dilakukan skenario serangan, bahwa penyerang telah berkompromi dengan penyedia layanan *chatting* sehingga ia mendapat *username* dan hash *password* pengguna.

Dengan diketahuinya itu, penyerang dapat berpura-pura menjadi seorang pengguna untuk melakukan serangan terhadap pesan *chatting*. Tetapi penyerang tidak bisa menggunakan Xabber, karena ia hanya memiliki hash *password* akun *chatting* dan tidak memliki *password* untuk mengakses pasangan kunci publik pengguna. Oleh karena itu, penyerang hanya bisa mengakses akun *chatting* pengguna menggunakan aplikasi Pidgin untuk melakukan serangan berupa penyadapan, modifikasi, pengiriman pesan palsu, dan *replay attack* terhadap pesan *chatting*. Berikut hasil serangan yang diberikan pada tiga tahap SRT.

a. *Trusted Public Key Distribution*
   Penyadapan yang dilakukan pada tahap ini tidak mampu memperoleh data rahasia seperti kunci privat pengguna. Sedangkan kunci publik pengguna terjamin keutuhan, keotentikan, dan nir-penyangkalannya karena telah ditandatangani (a, pub_a) oleh pihak ketiga. Lalu *replay attack* yang dilakukan tidak berpengaruh apapun karena tahap ini hanya dilakukan satu kali ketika instalasi aplikasi.
b. *Key Exchange with Digital Signature*
   Penyadapan yang dilakukan pada tahap ini tidak mampu memperoleh data rahasia seperti kunci sesi z. Sedangkan pesan pertukaran kunci $[x]G, t_a$ terjamin keutuhan, keotentikan, dan nir-penyangkalannya karena telah ditandatangani $S_a([x]G,b,t_a)$ oleh pengguna. Lalu *replay attack* yang dilakukan tidak berpengaruh apapun karena terdapat catatan waktu sebagai penentu waktu berlakunya pesan pertukaran kunci.
c. *Signed and Encrypted Message Transmission with Key Derivation Function*
   Penyadapan yang dilakukan pada tahap ini tidak mampu memperoleh pesan *chatting* m, karena terenkripsi(m). Sedangkan pesan *chatting* m terjamin keutuhan, keotentikan, dan nir-penyangkalannya karena telah ditandatangani (m) oleh pengguna. Lalu *replay attack* yang dilakukan tidak berpengaruh apapun Karena telah menerapkan KDF $v = H(v,c)$ sehingga kunci v selalu berubah setiap kali mengenkripsi pesan.

Selanjutnya dapat disajikan perbandingan layanan keamanan antara aplikasi Xabber dengan OTR dan Xabber dengan SRT berdasarkan hasil pengujian keamanan dan tinjauan pustaka [1][3]. Pembandingan difokuskan pada layanan keamanan di tahap pertukaran kunci dan transmisi pesan *chatting* kedua protokol. Tabel 2 menunjukan bahwa Xabber dengan OTR mengalami kegagalan untuk menjamin keotentikan dan nir-peyangkalan pada pertukaran kunci dan tidak adanya jaminan nir-penyangkalan pada transmisi pesan. Sedangkan Xabber dengan SRT mampu menjamin kerahasiaan, keutuhan data, keotentikan, nir-penyangkalan, dan tahan *replay attack* di kedua tahap.

Tabel 2. Perbandingan Layanan Keamanan Protokol Kriptografi

No	Layanan Keamanan	Xabber dengan OTR		Xabber dengan SRT	
		Pertukaran Kunci	Transmisi Pesan	Pertukaran Kunci	Transmisi Pesan
1	Kerahasiaan	Ada	Ada	Ada	Ada
2	Keutuhan data	Ada	Ada	Ada	Ada
3	Keotentikan	Gagal	Ada	Ada	Ada
4	Nir-penyangkalan	Gagal	Tidak ada	Ada	Ada
5	Tahan *replay attack*	Ada	Ada	Ada	Ada

3.3.2 Pembandingan Performa

Untuk mengamati performa Xabber yang menerapkan SRT dan OTR, dilakukan pembandingan terhadap:

a. Kecepatan Proses

Pencatatan waktu untuk protokol SRT dan OTR di setiap tahapnya dilakukan pada 2 (dua) *smartphone* (lihat penjelasan 3.2) dengan tiga macam panjang pesan. Berdasarkan pencatatan didapatkan bahwa waktu rata- rata SRT untuk pertukaran kunci adalah 7,49 detik dan untuk transmisi pesan adalah 0,552 detik. Sedangkan waktu rata-rata OTR untuk pertukaran kunci adalah 13,280 detik dan untuk transmisi pesan adalah 0,01 detik. Terlihat bahwaSRT lebih cepat 6,045 detik untuk pertukaran kuncinya, namun sedikit lebih lambat 0,542 detik untuk transmisi pesannya terhadap OTR. Jika diakumulasikan selisih kedua tahap tersebut, maka protokol SRT lebih cepat 5,503 detik atau 41,5% lebih cepat dari protokol OTR.

b. Kemudahan Penggunaan

Untuk mengetahui tingkat kemudahan penggunaan aplikasi Xabber, maka dilakukan pengajuan kuesioner kualitatif kepada 7 (tujuh) orang responden. Pada pelaksanaannya, responden terlebih dahulu diminta

untuk menggunakan aplikasi Xabber yang menerapkan OTR dan Xabber yang menerapkan SRT. Hal ini dimaksudkan agar pengguna dapat membandingkan performa kedua aplikasi secara objektif.

Berdasarkan 7 (tujuh) kuesioner yang terkumpul, seluruh reponden memberikan keterangan yang searah pada setiap pertanyaanya. Dari seluruh pertanyaan dapat ditarik keterangan bahwa seluruh responden merasa kesulitan dalam pengorasian Xabber dengan OTR karena proses pertukaran kuncinya yang rumit. Pengguna dituntut untuk mengotentikasi kunci publik pengguna lain secara manual setelah OTR berjalan (*lazy authentication*). Sedangkan menurut responden, Xabber dengan SRT tidak membebaninya dengan tahapan yang rumit karena proses pengamanan berjalan secara otomatis termasuk otentikasi kunci publik pengguna lain. Dengan demikian, Xabber yang menerapkan SRT memiliki tingkat kemudahan penggunaan yang relative lebih tinggi dari Xabber dengan OTR.

4. SIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, dapat ditarik kesimpulan bahwa protokol SRT mampu menjamin kerahasiaan, keutuhan, keotentikan, nir-penyangkalan, dan tahan *replay attack* terhadap data penting di seluruh tahapnya dengan kekuatan algoritma yang relatif lebih tinggi. Sedangkan kecepatan proses dan kemudahan penggunaan yang diberikan oleh protokol SRT relatif lebih tinggi dari protokol OTR di aplikasi Xabber. Dengan demikian protokol SRT dapat menjadi solusi alternatif dalam pengamanan *chatting* yang sebelumnya gagal dijaga oleh protokol OTR. Pada pekerjaan selanjutnya perlu dilakukan penelitian untuk menghadirkan manajemen kunci yang lengkap seperti proses pembaruan kunci publik dan penelitian untuk menghadirkan protokol pertukaran kunci sesi pada *group chatting*.

5. DAFTAR RUJUKAN

[1] Bonneau, Joseph & Morrison, Andrew. 2006. *Finite-State Security Analysis of OTR Version 2* [online] (updated 2006)Available at: http://www.jbonneau.com/doc/BM06-OTR_v2_analysis.pdf. [Accessed 29 Juli 2016].

[2] Certicom. 2010. *SEC2*: *Recommended Elliptic Curve Domain Parameters.* Canada: Certicom Corp.

[3] Green, Matthew. 2014. *Noodling About IM Protocol* [online] (updated 26 Juli 2014)Available at: http://blog.cryptographyengineering.com/2014_07_01_archive.html. [Accessed 29 Juli 2016].

[4] Hankerson, D., Menezes,A. & Vanstone, S. 2004. *Guide to Elliptic Curve Cryptography.* America: Springer.

[5] ISO. 2006. *ISO/IEC 18033-2 - Information Technology - Security Techniques – Encryption Algorithms*. Switzerland: ISO.

[6] NIST. 2001. *Recommendation for Block Cipher Modes of Operation-Methods and Techniques.* America: U.S. Department of Commerce.

[7] NIST. 2010. *Suite B Implementer's Guide to FIPS 186-3 (ECDSA).* America: U.S.Department of Commerce.

[8] NIST. 2012. *Recommendation for Key Management-Part1: General (Revision 3).* America: U.S.Department of Commerce.

[9] OTR Development Team. 2012. *Off The Record Messaging Protocol version 3* [online] (updated 2012) Available at: https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html [Accessed 29 Juli 2016].

[10] OTR Development Team. 2015. *OTR-Enabled Software* [online] (updated 2015) Available at: https://otr.cypherpunks.ca/software.php [Accessed 29 Juli 2016].

[11] Rogaway, Phillip. 2011. *Evaluation of Some Blockcipher Modes of Operation.* Japan: CRYPTREC.

[12] Schneier, Bruce. 1996. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons.

[13] Stalling, William. 2014. *Cryptography and Network Security 6th Edition*. America: Prentice Hall.

Halaman ini sengaja dikosongkan