Rancang Bangun Perangkat Lunak *Agievic* Berbasis *Challenge And Respond* Untuk Pengamanan Komunikasi *Peer To Peer Video Conference*

Agung Nugraha

Abstract— *Kebutuhan untuk layanan video conferencing menunjukkan peningkatan yang signifikan dalam bidang bisnis, militer dan pemerintahan. Akan tetapi, seiring dengan berkembangnya penggunaan video conference, tentunya terdapat ancaman terhadap aplikasi video conference seperti adanya penyadapan dari pihak yang tidak berkepentingan. Proses otentikasi pada sistem video conference juga dapat menjadi potensi ancaman dari pihak luar, yaitu dengan menyalin pesan stream yang dikirim antara client dan server kemudian mengirim kembali pesan tersebut ke server untuk mendapatkan akses terhadap sistem.*

Dalam makalah ini, dikembangkan sebuah perangkat lunak video conference dengan menerapkan hybrid cryptosystem yang diberi nama Agievic. Agievic menerapkan otentikasi dan key establishment antara clien dan server menggunakan sistem challenge respond berbasis public key cryptosystem, sementara komunikasi video conference diamankan dengan menerapkan teknik enkripsi untuk data video dan audio menggunakan algoritma AES 256 bit dengan mode CTR.

Berdasarkan hasil pengujian, data yang diperoleh pada saat penyadapan komunikasi video conference tidak dapat dikenali sebagai data video dan audio sehingga penyadap tidak dapat mengetahui komunikasi video conference yang sedang berlangsung. Adapun waktu rata-rata proses enkripsi dan dekripsi pada komunikasi video konferensi yaitu 6140,14 ms.

Keywords— *conference, otentikasi, hybrid, cryptosystem, Agievic*

1. PENDAHULUAN

Video conference merupakan suatu layanan komunikasi yang memiliki banyak keunggulan. Layanan ini dapat memungkinkan dua orang atau lebih melakukan komunikasi tatap muka secara *real time* di seluruh dunia melalui jaringan yang bersifat *public* maupun *private*. Selain itu, *video conference* yang mendukung jaringan IP mempunyai kelebihan dalam hal *performance*, manajemen, pemakaian biaya, dan pengembangan untuk perusahaan yang bergerak dalam bidang komunikasi *video conference* [13]

Kebutuhan akan layanan *video conference* akhir – akhir ini menunjukan peningkatan yang cukup signifikan. Hal ini dilihat dari berbagai *survey* yang telah dilakukan terhadap penggunaan aplikasi *video conference*. Seperti halnya *survey* yang telah dilakukan oleh *Research Now* dan *Global IP Solution*. Dari *survey* tersebut dikatakan bahwa dari 1200 profesional di bidang bisnis dari negara China, Jepang, Amerika dan Korea Selatan telah menggunakan aplikasi *video conference* untuk keperluan bisnis dan pribadi [21]. Selain di bidang bisnis, peningkatan juga terjadi di bidang militer dan pemerintahan [13]. Sebagai contoh dalam bidang militer, teknologi *video conference* memudahkan pimpinan di markas pusat untuk memberikan perintah atau strategi pada tentara yang sedang bertugas di lapangan [22]. Dalam pemerintahan, teknologi *video conference* digunakan oleh presiden untuk melaksanakan rapat dengan pemimpin negara lain.

Akan tetapi, teknologi *video conference* tersebut memiliki ancaman terhadap keamanan saat komunikasi berlangsung. Seseorang dapat melakukan penyadapan terhadap komunikasi *video conference* dan bisa dengan mudah merekam pembicaraan yang sedang berlangsung. Proses otentikasi sistem *video conference* juga dapat menjadi potensi ancaman dari pihak luar. Proses otentikasi dapat diserang dengan menyalin pesan *stream* yang dikirimkan diantara dua pihak dan mengirim kembali pesan tersebut kepada salah satu pihak untuk mendapatkan akses kepada suatu sistem. Oleh karena itu, diperlukannya pengamanan pada aplikasi *video conference* dengan tujuan untuk mengamankan komunikasi yang sedang berlangsung.

Aplikasi *video conference* yang aman merupakan aplikasi yang didalamnya terdapat kombinasi dari teknik enkripsi dan otentikasi. Teknik enkripsi dapat digunakan untuk mengamankan informasi yang akan dikirimkan, seperti *user id* dan *password* serta data *video* atau *audio*. Teknik otentikasi juga memungkinkan pihak yang mengirim informasi dapat memastikan bahwa informasinya terkirim pada pihak yang sah.

Dari permasalahan yang telah dijelaskan, maka akan dilakukan perancangan dan pembangunan perangkat lunak *Agievic* yang dapat digunakan untuk mengamankan komunikasi *video conference*. Pernah

dilakukan sebelumnya oleh peneliti lain yg relevan dengan penelitian yang dilakukan.

2. LANDASAN TEORI

2.1. *Kriptografi*

Kriptografi secara umum merupakan ilmu dan seni untuk menjaga kerahasian berita [10] Kriptografi juga dapat diartikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data [5].

Ada empat tujuan mendasar dari kriptografi yang juga merupakan aspek keamanan informasi, yaitu sebagai berikut :

a. Kerahasiaan, adalah layanan yang ditujukan untuk menjaga isi informasi dari siapapun, kecuali pihak yang memiliki kunci rahasia untuk membuka informasi yang telah disandikan.
b. Integritas data, berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk dapat menjaga integritas data, suatu sistem harus memiliki kemampuan untuk mendeteksi manipulasi data yang dilakukan pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pendistribusian data lain ke dalam data yang asli.
c. Otentikasi, berhubungan dengan identifikasi, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan harus diotentikasi keasliannya, isi datanya, waktu pengiriman, dan lain sebagainya.
d. Non-repudiasi, merupakan usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/ membuat.

2.2. *Protokol Kriptografi*

Protokol merupakan prosedur untuk mencapai suatu maksud yang dilakukan oleh minimal dua pihak, dilaksanakan terurut dari awal hingga akhir yang tiap langkahnya harus dilakukan satu per satu dan berurutan [10].

Suatu protokol memiliki beberapa karakteristik lain yaitu :

1. Tiap pihak yang terlibat dalam protokol harus mengetahui protokol dan keseluruhan langkah yang akan dilaksanakan;
2. Tiap pihak yang terlibat dalam protokol harus setuju untuk mengikutinya;
3. Protokol harus tidak ambigu, tiap langkah harus terdefinisi dengan jelas dan tidak ada peluang untuk ketidakpahaman;
4. Protokol harus lengkap, harus ada tinda khusus untuk tiap kemungkinan situasi.

Protokol kriptografi adalah suatu protokol y menggunakan kriptografi. Protokol kriptog melibatkan penggunaan algoritma kriptografi dan tuj protokol kriptografi lebih dari sekedar kerahasiaan [1

Tujuan utama penggunaan kriptografi dalam seb protokol adalah untuk mencegah atau pun mendet adanya penyadapan dan kecurangan yang dilakukan pihak yang terlibat dalam protokol maupun pihak luar protokol.

Terdapat tiga tipe protokol kriptografi, yaitu :

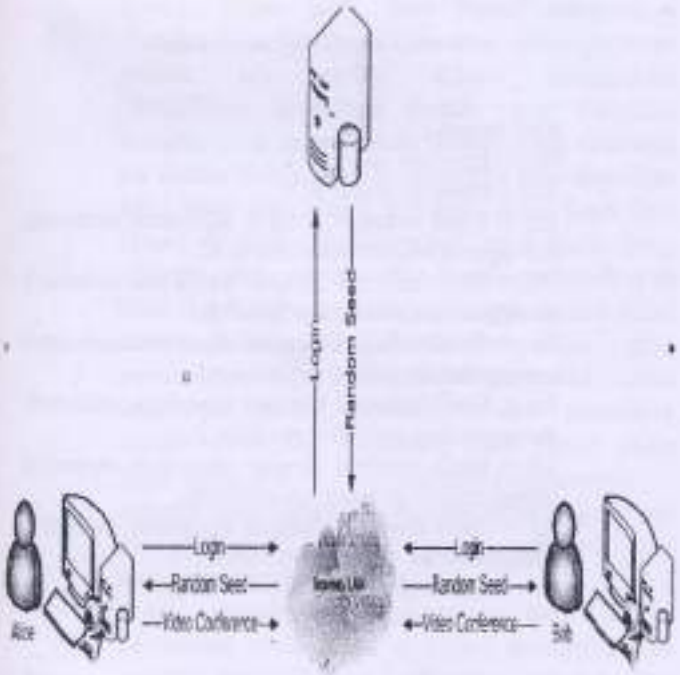
a. *Arbitrated Protocol*
   Protokol ini menggunakan *arbitrator* da komunikasinya. *Arbitrator* adalah pihak ke terpercaya yang tidak berkepentingan melakukan tugas untuk menyelesaikan proto Protokol ini menjamin tidak terjadi kecurangan saat komunikasi terjadi antara enti Tetapi, protokol ini mempunyai kekuran dalam sulitnya mencari pihak ketiga yang da jujur dan dapat dipercaya. Contoh dari proto yang menerapkan tipe *Arbitrated Protocol* ada *Secure Electronic Transaction*
b. *Adjudicated Protocol*
   Adjudicator adalah pihak ketiga yang da menilai suatu transaksi terlihat jujur atau ti dalam suatu perselisihan. Artinya adalah pi ketiga yang dapat menilai apakah transaksi y sedang terjadi dapat dilakukan atau tid Protokol ini tidak mempunyai kemampuan da mencegah terjadinya kecurangan saat komunik Contoh dari protokol yang menerapkan *Adudicated Protocol* adalah *Karberos Protoc*
c. *Self-enforcing Protocol*
   Protokol ini memberikan jaminan keadilan da berhubungan melalui protokol tanpa melibat pihak ketiga. Tetapi, protokol ini tidak da diimplementasikan pada setiap situasi kondisi. Contoh dari protokol yang menerap tipe *Self-enforcing Protocol* antara lain ad *Diffie Hellman Key Exchange* dan *Authentica Key Exchange* Protokol 2.

3. DESAIN

Dalam perancangan aplikasi *Agievic*, akan diguna protokol pada [6] yang terdiri dari proses otentikasi, *establishment* dan komunikasi. Dari tiga proses proto tersebut, diimplementasikan proses otentikasi dan *establishment* pada aplikasi Agievic sedangkan p proses komunikasi dilakukan modifikasi den mengimplementasikan protokol RTP yang diaman dengan algoritma AES 256 bit.

Sistem *Agievic* terdiri dari dua entitas yaitu *client* dan *server*. *Server* merupakan entitas yang berperan untuk menampung *client* yang ingin berkomunikasi. Ketika *client* ingin berkomunikasi dengan *client* lainnya, maka keduanya harus *login* terlebih dahulu ke *server*. Selain itu, *server* juga berperan sebagai *Trusted-Third Party* (TTP) yang akan mendistribusikan *random seed* untuk menggenerate *session key* pada setiap *client*. *Client* merupakan entitas yang melakukan komunikasi *video conference* secara aman dengan menggunakan aplikasi Agievic yang berkomunikasi secara *point to point*.

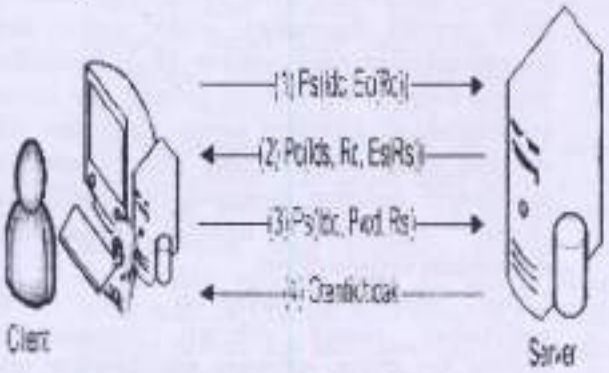
Aplikasi *Agievic* merupakan aplikasi yang dapat melakukan komunikasi *video conference* secara aman dengan mengimplementasikan teknik enkripsi dan otentikasi. Aplikasi ini juga memiliki *random generator* yang digunakan untuk pembangkitan nilai *random* pada proses otentikasi dan pembangkitan *session key* untuk proses komunikasi antara *client*. Keamanan pada aplikasi *Agievic* yaitu terletak pada proses penyandian informasi baik pada proses otentikasi, pertukaran kunci maupun pada proses komunikasi antara *client*. Berikut ini merupakan gambaran dari sistem *Agievic*.



Gambar 1. Gambaran umum sistem Agievic

3.1. *Proses Otentikasi*

Proses otentikasi dilakukan oleh *client* dan *server* (*mutual authentication*) dengan menggunakan sistem *challenge respond*. Tahap pertama pada proses ini yaitu diawali dengan *client* yang mengirimkan *request* kepada *server* untuk dapat masuk ke dalam sistem. Kemudian *server* akan memberikan respon dari *request client*. *Server* akan membandingkan data yang dikirim oleh *client* dengan data yang terdapat pada *database server*, jika sesuai maka *client* dapat masuk kedalam sistem. Gambar di bawah ini merupakan proses otentikasi pada aplikasi *Agievic*.



Gambar 2. Proses Otentikasi

Keterangan :
Ec = Hasil enkripsi dengan menggunakan *private key* milik *client*
Es = Hasil enkripsi dengan menggunakan *private key* milik *server*
Idc = Identitas *client*
Ids = Identitas *server*
Pc = Hasil enkripsi dengan algoritma asimetrik menggunakan *public key client*
Ps = Hasil enkripsi dengan algoritma asimetrik menggunakan *public key server*
Rc = Bilangan acak yang dibangkitkan oleh *client*
Rs = Bilangan acak yang dibangkitkan oleh *server*
pwd = *Password*

Penjelasan protokol proses otentikasi adalah sebagai berikut :

1) *Client* mengirimkan *request* kepada *server* untuk dapat masuk ke dalam sistem *Agievic* dengan cara membangkitkan nilai *random* (Rc), nilai random Rc tersebut kemudian dienkripsi menggunakan *private key client*. Hasil enkripsi nilai random kemudian ditambahkan dengan identitas *client* (Idc) dan dienkripsi menggunakan *public key server* untuk kemudian dikirimkan kepada *server*.
2) *Ciphertext* yang diterima dari *client* didekripsi menggunakan *private key server* untuk mendapatkan Idc dan *ciphertext* Rc. *Ciphertext* Rc tersebut didekripsi menggunakan *public key client* sehingga mendapatkan nilai *random* yang dikirimkan oleh *client* (Rc). Setelah itu *server* akan membandingkan Rc dengan nilai *random* pada *database*, jika nilai Rc tersebut sudah ada dalam *database*, maka *server* tidak melanjutkan langkah protokol selanjutnya. Namun jika nilai Rc tersebut tidak ada dalam *database*, maka Rc

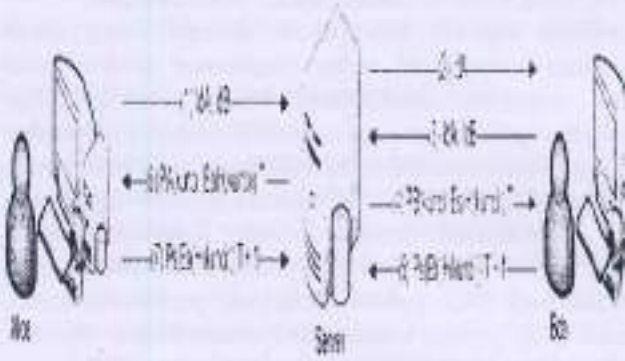
akan disimpan di *database*. Setelah dapat memastikan bahwa nilai random sebelumnya tidak pernah digunakan, maka *server* akan membangkitkan nilai *random* (Rs), melakukan proses enkripsi Rs dengan *private key server*, menambahkan identitas *server* (Ids) dan nilai *random* yang diterima dari *client;* kemudian melakukan enkripsi menggunakan *public key client*. *Ciphertext* tersebut selanjutnya dikirimkan kepada *client*.

3) *Client* menerima *ciphertext* dari *server*, melakukan proses dekripsi menggunakan *private key client*, sehingga mendapatkan nilai *random* yang dikirimkan oleh *server* (Rs) dengan mendekripsi menggunakan *public key server*, mendapatkan identitas *server* serta mendapatkan nilai *random* yang sebelumnya dikirimkan kepada *server*. *Client* akan membandingkan nilai *random* yang sebelumnya telah dibangkitkan dengan nilai yang diberikan oleh *server*, jika nilai *random* tersebut berbeda, maka entitas yang dihubungi/berkomunikasi adalah *server* yang tidak dikehendaki; sebaliknya jika nilai yang diterima adalah sama, maka *client* meyakini bahwa entitas yang dihubungi/berkomunikasi adalah *server* yang berhak (terotentikasi). Jika *client* telah memastikan bahwa entitas yang dihubungi adalah *server* yang dikehendaki, maka *client* melakukan proses enkripsi identitas, *password*, dan nilai *random* yang telah diterima dengan menggunakan *public key server*. *Ciphhertext* tersebut kemudian dikirimkan kepada *server*.

4) *Ciphertext* yang diterima dari *client* didekripsi menggunakan *private key server* sehingga mendapatkan identitas dan *password client* serta nilai *random* yang sebelumnya dikirimkan kepada *client*. Jika nilai *random* yang diterima dari *client* sama dengan nilai yang dibangkitkan oleh *server*, maka *server* meyakini bahwa entitas yang dihubungi/berkomunikasi adalah *client* yang berhak (terotentikasi). Jika *server* telah memastikan bahwa entitas yang dihubungi adalah *client* yang dikehendaki, maka *server* akan membandingkan identitas dan *password* yang telah diterima dengan data yang tersimpan dalam *database server*. Jika identitas dan *password* tersebut sesuai dengan *database server*, maka *client* dapat tergabung dalam sistem. Sebaliknya, jika identitas dan *password* tidak sesuai dengan *database server*, maka *client* tersebut tidak dapat masuk dalam sistem.

3.2. *Key Establishment*

Proses *key establishment* hanya dapat terjadi sete *client* melakukan *login* kepada *server*. Saat *cl* melakukan request kepada *server* untuk berkomunik dengan *client* lainnya dan *client* yang dituju meneri request tersebut, maka *server* akan mengirim *random seed* yang sama kepada kedua *client* y nantinya akan digunakan untuk pembangkitan *sess key*. Gambar di bawah ini merupakan proses *establishment* pada aplikasi *Agievic*.



Gambar 3. Proses *Key Establishment*

Keterangan :
IdA = Identitas *client* A.
IdB = Identitas *client* B.
PA = Hasil enkripsi dengan algoritma asime menggunakan *publickey client* A.
PB = Hasil enkripsi dengan algoritma asime menggunakan *public key client* B.
Ps = Hasil enkripsi dengan algoritma asime menggunakan *public key server.*
Ea = Hasil enkripsi dengan algoritma asime menggunakan *private key client* A.
Eb = Hasil enkripsi dengan algoritma asime menggunakan *private key client* B.
Es = Hasil enkripsi dengan algoritma asime menggunakan *private key server.*
H = fungsi hash.
T = *timestamp.*

Penjelasan protokol yang digunakan un penyediaan kunci Agievic adalah sebagai berikut :

1) *Client* melakukan *request* komunikasi kep *server* dengan mengirimkan identitas diri dan identitas *client* lain yang akan dihubung
2) *Server* menerima permintaan panggi komunikasi dan melihat status *client* peneri jika status *client* penerima tidak aktif, m *server* akan mengirimkan pesan kepada *cl* pemanggil bahwa *client* yang dituju sed tidak aktif . Namun jika status *client* peneri aktif, maka *server* akan meneruskan permintaan tersebut dengan mengirim

identitas *client* yang meminta panggilan komunikasi kepada *client* penerima.

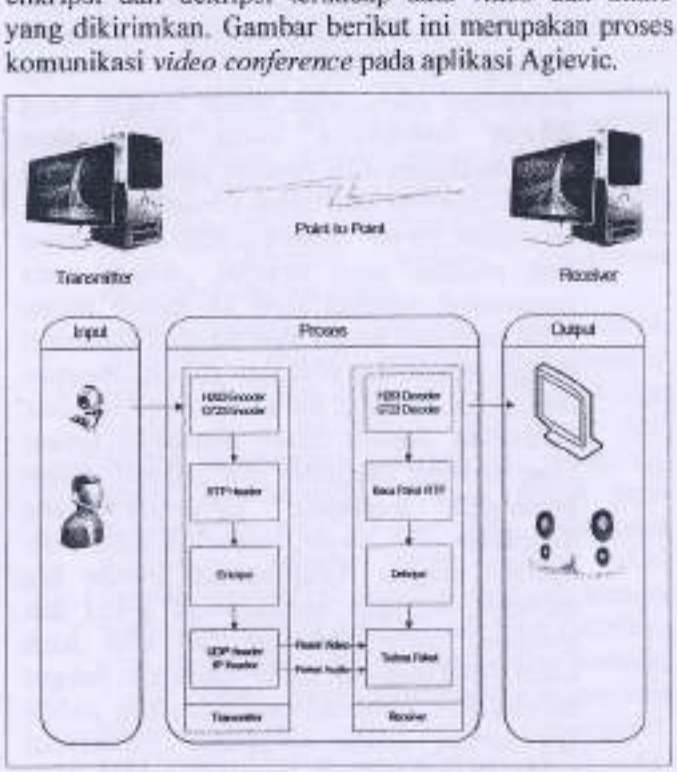
3) *Client* menerima permintaan panggilan komunikasi dan mengirimkan identitas dirinya dan identitas *client* lain yang menghubunginya.
4) *Server* menerima *respond* persetujuan, membangkitkan nilai *random* yang digunakan sebagai *random seed* untuk komunikasi *client*. *Server* membuat *digital signature* dari *random seed* tersebut dengan cara memasukan *random seed* ke dalam fungsi *hash*, hasil dari fungsi *hash* tersebut dienkripsi dengan *private key* miliknya dan digabungkan dengan *random seed* dan *timestamp*, kemudian dikirimkan kepada *client* penerima (*client* yang menerima panggilan komunikasi).
5) *Client* penerima menerima *ciphertext* yang dikirimkan oleh *server*, kemudian melakukan dekripsi dengan menggunakan *private key* miliknya, sehingga mendapatkan kunci dan *timestamp* serta *ciphertext* dari nilai *hash* kunci. Nilai *hash* dari kunci didapatkan dengan mendekripsi *ciphertext* menggunakan *public key server*. *Client* melakukan identifikasi keutuhan kunci yang diterima dengan cara memasukan kunci yang diterima ke dalam fungsi *hash*, sehingga mendapatkan nilai hash dari kunci tersebut. Nilai *hash* dari kunci dibandingkan dengan nilai hash yang dikirim oleh *server*. Jika hasil perbandingan nilai hash tersebut adalah sama, maka nilai kunci tidak mengalami kerusakan atau perubahan. Jika *client* telah yakin akan integritas kunci, maka *client* penerima membuat *digital signature* dari kunci yang telah diperoleh, menggabungkannya dengan *timestamp* + 1 dan melakukan enkripsi menggunakan *public key server* lalu mengirimkannya kepada *server*.
6) *Server* menerima *ciphertext* dari *client* kemudian melakukan dekripsi menggunakan *private key* miliknya, sehingga mendapatkan *digital signature* kunci dan *timestamp* + 1 yang dikirimkan *client*. Kemudian *server* melakukan identifikasi mengenai keutuhan kunci yang dikirim sebelumnya dengan cara membandingkan *digital signature* kunci dan *timestamp* yang diterima dengan yang dikirim sebelumnya. Jika *digital signature* kunci yang dikirimkan *client* tidak sesuai dengan yang dikirim sebelumnya, maka protokol akan kembali pada langkah empat (4) atau jika *timestamp* yang diterima sudah terdapat dalam *database*, berarti teridentifikasi langkah pada protokol ini sebagai *replay attack*, sehingga protokol akan terhenti. Namun jika *digital signature* kunci atau *timestamp* + 1 yang dikirimkan *client* telah sesuai dengan yang dikirim sebelumnya, maka *Server* akan membangkitkan nilai *random* yang digunakan sebagai *random seed* untuk komunikasi *client*. Kemudian *server* membuat *digital signature* dari *random seed* tersebut, dengan cara memasukan *random seed* ke dalam fungsi *hash*, hasil dari fungsi *hash* tersebut dienkripsi dengan *private key* miliknya dan digabungkan dengan *random seed* dan *timestamp*, kemudian dikirimkan kepada *client* pemanggil (*client* yang meminta panggilan komunikasi). *Client* pemanggil menerima *ciphertext* yang dikirimkan oleh *server*, kemudian melakukan dekripsi dengan menggunakan *private key* miliknya, sehingga mendapatkan kunci dan *timestamp* serta *ciphertext* dari nilai *hash* kunci. Nilai hash dari kunci didapatkan dengan mendekripsi *ciphertext* menggunakan *public key server*. *Client* melakukan identifikasi keutuhan kunci yang diterima dengan cara memasukan kunci yang diterima ke dalam fungsi *hash*, sehingga mendapatkan nilai *hash* dari kunci tersebut. Kemudian nilai *hash* dari kunci yang diterima tersebut dibandingkan dengan nilai *hash* yang dikirim oleh *server*. Jika hasil perbandingan nilai *hash* tersebut adalah sama, maka nilai kunci tidak mengalami kerusakan atau perubahan. Jika *client* telah yakin akan integritas kunci, maka *client* pemanggil membuat *digital signature* dari kunci yang telah diperoleh, kemudian menggabungkannya dengan *timestamp* + 1 dan melakukan enkripsi menggunakan *public key server* dan mengirimkannya kepada *server*.

3.3. *Proses Komunikasi*

Komunikasi *video conference* dilakukan oleh *client* secara *point to point* tanpa melibatkan *server*. Untuk melakukan proses komunikasi secara aman, *client* akan menggunakan *session key* hasil dari pembangkitan *random* generator dengan input *random seed* dari *server*. *Session key* tersebut digunakan sebagai input kunci pada penyandian data *video* dan *audio* menggunakan algoritma AES.

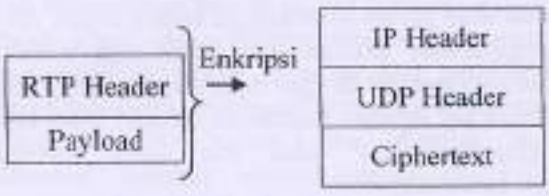
Komunikasi *video conference* pada aplikasi Agievic menggunakan protokol RTP. Oleh karena itu, akan digunakan suatu proses yang dapat mengirimkan paket data *video* dan *audio* melalui protokol RTP. Selain itu, digunakan juga suatu proses yang dapat melakukan

enkripsi dan dekripsi terhadap data *video* dan *audio* yang dikirimkan. Gambar berikut ini merupakan proses komunikasi *video conference* pada aplikasi Agievic.



Gambar 4. Komunikasi *video conference* aplikasi Agievic

Pada komunikasi *video conference*, protokol RTP menggunakan dua buah port yang masing – masing digunakan untuk melakukan transmisi data *video* dan *audio*. Proses transmisi dimulai ketika aplikasi melakukan capture *video* dan *audio* dari *device* yang digunakan. Setelah melakukan *capture*, data *video* dan *audio* tersebut dikompresi menggunakan *codec* yang telah ditentukan yaitu H263 untuk *video* dan G723 untuk *audio*. Data yang telah dikompresi dibentuk menjadi paket RTP dengan diberikan *header* RTP dan dienkripsi menggunakan algoritma AES 256 dengan mode CTR. *Ciphertext* yang telah dihasilkan tersebut kemudian dibentuk menjadi paket UDP yang siap untuk ditransmisikan. Gambar berikut ini merupakan susunan paket data yang dienkripsi dan siap ditransmisikan.



Gambar 5. Susunan paket yang siap ditransmisikan

Proses penerimaan paket data pada aplikasi *Agievic* dimulai ketika data diterima oleh aplikasi. Saat data diterima, data kemudian didekripsi menggunakan algoritma dan kunci yang sesuai. Oleh karena itu, kunci yang digunakan untuk mendekripsi tidak ses maka paket data tidak dapat dikenali sebagai paket RTP sehingga aplikasi tidak dapat menjalankan *vi* dan *audio* yang dikirim. Jika kunci yang diguna sesuai maka data *video* dan *audio* yang didapatkan a didecode menggunakan *codec* yang sesuai u selanjutnya dijalankan oleh aplikasi.

4. IMPLEMENTASI DAN PENGUJIAN

Implementasi aplikasi dilakukan pada OS berb Windows yang terkoneksi dengan jaringan internet. S user mengaktifkan aplikasi untuk pertama kali m akan muncul tampilan *login*. Pada tampilan *lo* terdapat *field username* dan *password* yang harus oleh user untuk dapat terkoneksi dengan server. Gam 4.1 menunjukkan tampilan ketika user melakukan *lo* kepada server.

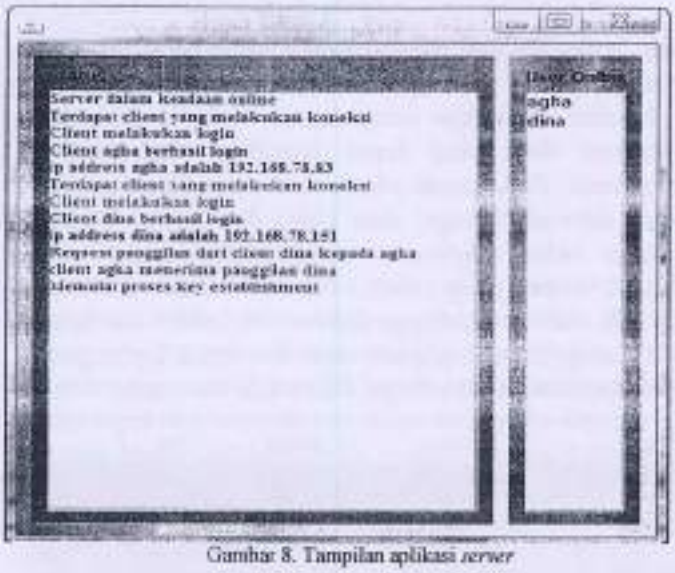
Gambar 6. Tampilan *Login* aplikasi Agievic

Pada proses koneksi ke *server*, terjadi pr otentikasi antara *client* dan *server* yang dilakukan se *mutual authentication*, sehingga jika kedua entitas d memastikan bahwa pihak yang dihubungi adalah ben pihak yang sah, maka *client* dapat masuk ke da sistem Agievic. Untuk melakukan komunikasi v *conference* dengan user lain, maka user terlebih dal memilih user tujuan yang *online* pada *list user on* lalu menekan tombol *call*. Permintaan panggilan a disampaikan ke *server* dan *server* akan meneru panggilan kepada *user* tujuan. Jika user yang dipan sedang tidak melakukan komunikasi *video confere* dan panggilan tersebut diterima, maka aplikasi se otomatis akan memberitahukan *server* bahwa pangg diterima. Selanjutnya user dapat melakukan komuni *video conference* secara *point to point* tanpa melibat *server*. Gambar dibawah ini menunjukkan tamp utama aplikasi *Agievic* ketika user berhasil *online* terdapat user lain yang ingin berkomunikasi.

Gambar 7. Tampilan Utama aplikasi Agievic

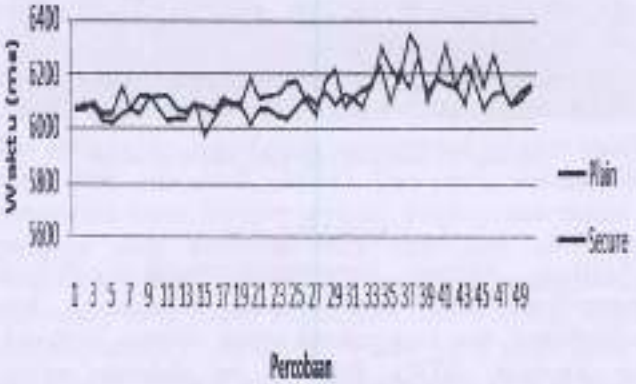
Mode komunikasi pada aplikasi *Agievic* terdiri dari 2 yaitu mode *plain* dan *secure*. Mode komunikasi ini ditujukan ketika komunikasi *video conference* sedang dilaksanakan. Ketika sesi komunikasi *video conference* dimulai antara 2 *client*, aplikasi menjalankan mode *plain*, user dapat mengganti mode tersebut dengan menekan tombol *secure*. Sebaliknya, user juga dapat mengubah mode *secure* menjadi *plain* dengan menekan tombol *plain*.

Server mempunyai layanan untuk menerima proses *login* dari *client* dan permintaan komunikasi *video conference*. Ketika *server* diaktifkan, maka secara otomatis *server* membuka koneksi TCP pada *port* 56565. Dengan menggunakan *port* tersebut, *client* mengirimkan permintaan ke *server* untuk melakukan proses *login* ataupun komunikasi *video conference*. Setiap permintaan yang dilakukan oleh *client* akan ditampilkan pada aplikasi *server*. Aplikasi *server* juga menampilkan *client* yang telah melakukan *login* kepada sistem. Gambar berikut ini merupakan tampilan pada aplikasi *server* ketika terdapat *client* yang melakukan permintaan kepada *server*.



Gambar 8. Tampilan aplikasi *server*

Pengujian dilakukan dengan melakukan perhitungan waktu yang dibutuhkan oleh aplikasi ketika melakukan komunikasi *video conference* dalam mode *secure* dan mode *plain*. Perhitungan waktu dimulai ketika aplikasi melakukan *capture video* dan *audio* yang akan ditransmisikan dari sisi pengirim sampai dengan paket data *video* dan *audio* dijalankan di sisi penerima. Pengujian dilakukan sebanyak 50 kali dan selanjutnya data yang didapatkan akan dibandingkan. Berikut ini merupakan grafik hasil pengujian kecepatan aplikasi *Agievic*.

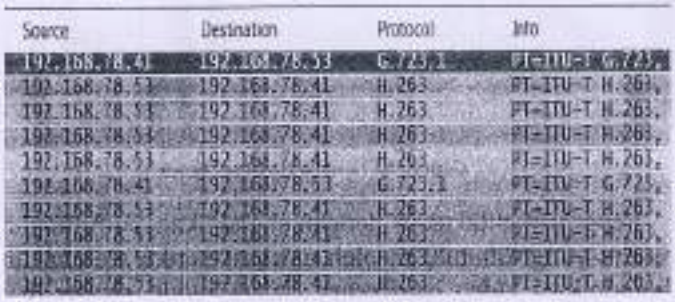


Gambar 9. Grafik perbandingan waktu mode *plain* dan *secure*

Berdasarkan data hasil pengujian (data terlampir), waktu rata – rata aplikasi melakukan komunikasi *video conference* pada mode *plain* dan *secure* masing – masing adalah 6101,74 ms dan 6140,14 ms. Sehingga selisih waktu rata – rata yang didapatkan adalah 38,4 ms Pengujian keamanan dilakukan dengan tujuan untuk mengetahui proses pengamanan *video conference* pada aplikasi Agievic. Pada pengujian ini akan dilakukan penyadapan terhadap komunikasi *video conference* dengan menggunakan *software* Wireshark. Penyadapan terhadap transmisi *video conference* tersebut dilakukan

baik ketika aplikasi menggunakan mode *secure* maupun mode *plain*.

Setelah dilakukan penyadapan, terdapat perbedaan mengenai data yang dapat ditangkap oleh *software* Wireshark. Pada mode *plain*, data yang ditransmisikan dapat dikenali sebagai data *video* dan *audio*. Berbeda dengan *video conference* yang menggunakan mode *secure*, data yang akan ditransmisikan dienkripsi terlebih dahulu sehingga ketika dilakukan *capturing*, data yang ditangkap pada saat transmisi berlangsung tidak dapat dikenali sebagai data *video* dan *audio*.



Gambar. 10. Data video dan audio pada mode plain

Gambar. 11. Data video dan audio pada mode secure

5. KESIMPULAN

Aplikasi *Agievic* mengimplementasikan *hybrid cryptosystem* baik pada proses otentikasi, *key establishment* dan komunikasi sesuai dengan protocol pade (Prakasa, 2008). Protokol ini didesain untuk resisten terhadap *replay attack* dengan mengimplementasikan *timestamp* pada proses *key establishment* dan *random number* pada proses otentikasi. Otentikasi pada protokol ini menggunakan sistem *challenge-respond* dan proses otentikasi dilakukan oleh kedua belah pihak (*client* dan *server*). Pada proses *key establishment*, *server* membangkitkan *random seed* yang akan digunakan oleh *client* untuk membangkitkan *session key* dalam komunikasi. Pada proses komunikasi, dilakukan modifikasi dengan mengimplementasikan protokol RTP yang ditambahkan proses enkripsi dan dekripsi data *video* dan *audio*. Melalui langkah – langkah pengamanan tersebut ma seseorang dapat melakukan komunikasi *vi conference* dengan menggunakan aplikasi Agie secara aman. Hal ini berdasarkan hasil pengujian dim ketika dilakukan penyadapan, data *video* dan *audio* ya ditransmisikan saat komunikasi berlangsung dal mode *secure* tidak dapat dibaca sehingga menamb kesulitan bagi pihak yang tidak berhak un mengetahui sesi komunikasi *video conference*.

6. DAFTAR PUSTAKA

[1] Ferguson, Niels & Schneier, Bruce. 2003. *Practical Cryptogra* Indiana : Wiley Publishing, Inc.

[2] Hardono, Agung. 2008. *Implementasi Aplikasi Video Confer Menggunakan Algoritma Blowfish Berbasis Java M Framework*. Skripsi tidak diterbitkan. Bandung. Institut Tekno Telkom.

[3] Kadir Abdul. 2004. *Dasar Pemrograman Java*. Yogyaka ANDI

[4] Menezes, Alfred J. et.al. 1997. *Handbook of Apy Cryptography*. Newyork : CRC Press.

[5] Munir, Rinaldi. 2006. *Kriptografi*. Bandung : Informatika

[6] Prakasa, Panji Yudha. 2008. *Rancang Bangun Perangkat L Secure Voice over Internet Protokol (SVoIP)*. Tugas Akhir diterbitkan. Bogor. Sekolah Tinggi Sandi Negara.

[7] Pramudita, Alfa. *Penggunaan Enkripsi AES Dengan M Operasi CBC Dan CTR Pada JAVAScript Dengan Lib Pidcrypt*.

[8] Priyanggoro, Sigit. 2006. *Membuat Media Player dengan Media Framework (JMF) 2.1*. Jakarta : IlmuKomputer.Com.

[9] Rosenberg, Doug & Stephens, Matt. 2007. *Use Case D Object Modeling with UML*. Apress.

[10] Schneier, Bruce. 1996. *Applied Cryptography, Second Edi Protocols, Algorithms, and Source Code in C*. Newyork : Wiley & Sons, Inc.

[11] Sumarkidjo et.al. 2007. *Jelajah Kriptologi*. Jakarta : Lem Sandi Negara RI.

[12] Terry, Andi Rifa. 2007. *Rancang Bangun Dan Analisa Qos Videostreaming Melalui Jaringan Vpn*. Surabaya. In Teknologi Sepuluh Nopember.

[13] Weinstein, Ira M. 2006. *Employing IT-Level Security fo Videoconferencing*. Wainhouse Research.

[14] blog.radvision.com/videoovernetenterprise/2009/09/29/video-conferencing-mass-deployment-survey-says. (Diakses tangga Maret 2010).

[15] capec.mitre.org/data/definitions/90.html. (Diakses tanggal Maret 2010).

[16] cnq.mod.gov.cn/DefenseNews/2009-12/30/content_4115081.h (Diakses tanggal 17 Maret 2010)

[17] genprot.com/Military.htm. (Diakses tanggal 17 Maret 2010).

[18] onefunz.wordpress.com/category/client-server-vs-peer-to-peer (Diakses tanggal 21 Maret 2010).

[19] technet.microsoft.com/en-us/library/cc751396.aspx. (Di tanggal 21 Maret 2010).

[20] www.ehow.com/about_5380893_peertopeer-vs-clientserver-networks.html. (Diakses tanggal 21 Maret 2010).

[21] www.eweek.com/c/a/VOIP-and-Telephony/Video-Conferenci on-the-Rise-Survey-Shows-137811/. (Diakses tanggal 21 M 2010).

[22] www.futuregov.net/articles/2009/jan/28/cambodia-leverages-videoconferencing/ . (Diakses tanggal 21 Maret 2010).

[23] www.ittelkom.ac.id/library/video_conference. (Diakses tangg Nopember 2009).

[24] www.videoconferencingbasic.blogspot.com. (Diakses tangga Maret 2010).