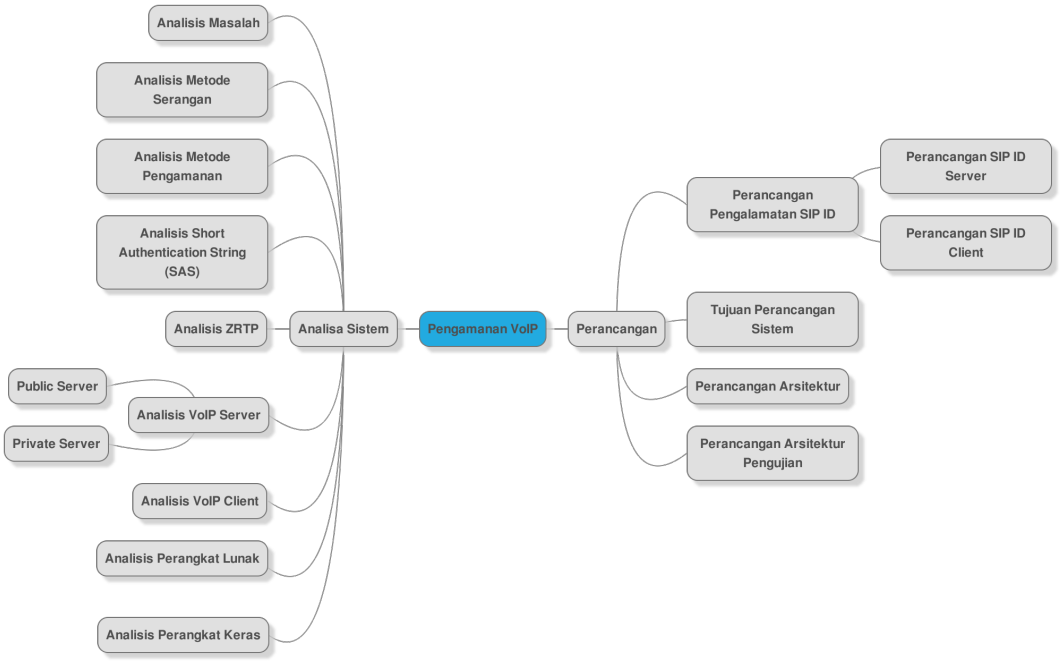
BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi jelas. Metode untuk menguji, menilai, dan memahami sistem pemikiran yang kompleks dengan memecahnya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur-unsur itu menjadi jelas. Pada Gambar 3.1, diperlihatkan *mind map* agar dapat memberikan gambaran dari penelitian yang dilakukan.



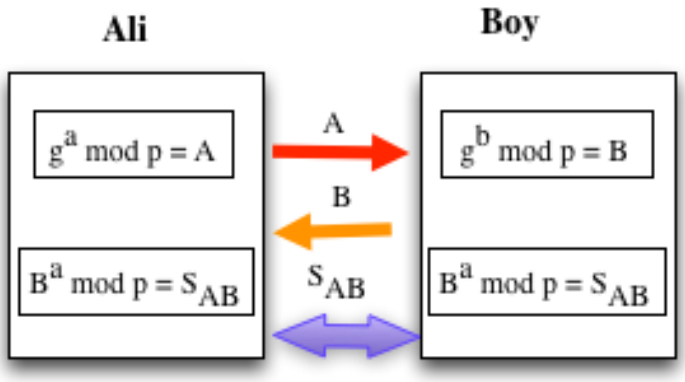
Gambar 3. 1 Mind Map Penelitian

3.1.1 Analisis Masalah

VoIP *server* berfungsi sebagai penyedia layanan VoIP. *Initiator* (pemanggil) akan menghubungi VoIP *server* dan memberi informasi bahwa akan menghubungi *responder* (penerima panggilan). Seluruh proses tersebut berjalan di jaringan internet. Karena itu aspek keamanan merupakan hal yang perlu diperhatikan untuk menjaga keberlangsungan komunikasi VoIP.

Terdapat tiga metoda yang bisa digunakan untuk mengamankan komunikasi VoIP, yaitu TLS (*Transport Layer Security*), TLS menyamarkan layer protokol dari VoIP dalam sebuah jaringan, sehingga sulit untuk dideteksi atau diblok. SRTP (*Secure Realtime Transport Protocol*), meng-enkripsi suara dari client ke server, tetapi pengguna yang memiliki akses ke server dapat membaca suara tersebut. *Zimmerman Realtime Transport Protocol* (ZRTP), ZRTP menggunakan public *key cryptography* sehingga enkripsi dapat tersampaikan dari pemanggil dan penerima tanpa diketahui server ataupun pihak ketiga.

Pada SRTP terdapat *Diffie Hellman* merupakan *key exchange* yang digunakan untuk pertukaran public key, pada Gambar 3.2 diperlihatkan cara kerja DH *key exchange*.



Gambar 3. 2 Alur *DH Key Exchange*

Pada Gambar 3.2, diilustrasikan Ali dan Boy sedang bertukar kunci. Hasil dari *shared secret* yang diterima akan digunakan sebagai kunci untuk membuka pesan yang di-enkripsi seperti dijelaskan pada Tabel 3.1.

Tabel 3. 1 Deskripsi *DH Key Exchange*

Ali	Boy
1. Memilih angka rahasia ‘a’ 2. Mengkalkulasi ‘A’ menggunakan (nilai p dan g telah ditentukan)	1. Memilih angka rahasia ‘b’ 2. Mengkalkulasi ‘B’ menggunakan (nilai p dan g telah ditentukan)

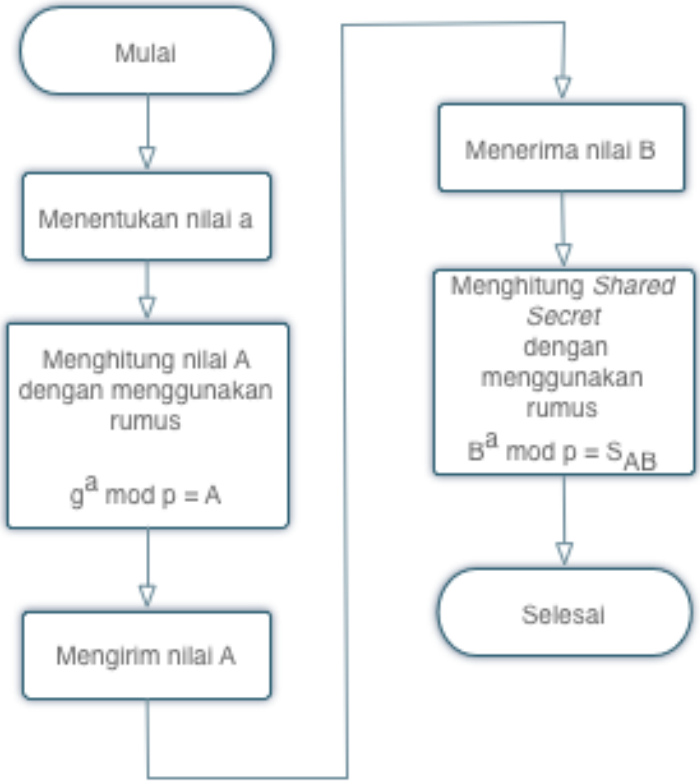
$g^a \bmod p = A$ 3. Mengirim 'A' ke Boy 4. Menerima 'B' dari Boy 5. Mengkalkulasi *Shared Secret* menggunakan $B^a \bmod p = S_{AB}$	$g^b \bmod p = B$ 3. Mengirim 'B' ke Ali 4. Menerima 'A' dari Ali 5. Mengkalkulasi *Shared Secret* menggunakan $A^b \bmod p = S_{AB}$

Dari tahapan proses DH pada tabel 3.1, untuk membuktikan bahwa Ali dan Boy memiliki *shared secret* yang sama maka perlu dilakukan simulasi. Pada simulasi DH ini, nilai p dan g yang merupakan *public key* telah ditentukan. Nilai p=17 dan g=3. Nilai *public value* tersebut akan digunakan dalam simulasi *Diffie Hellman* pada tabel 3.2

Tabel 3. 2 Simulasi *Diffie Hellman*

Ali	Boy
1. Private key a=15 2. Menghitung nilai A $g^a \ mod \ p = A$ $3^{15} \bmod 17 = 6$ 3. Mengkalkulasi *Shared Secret* (S_{AB}) menggunakan $B^a \ mod \ p = S_{AB}$ $12^{15} \bmod 17 = 10$	1. Private key b=13 2. Menghitung nilai B $g^b \ mod \ p = B$ $3^{13} \bmod 17 = 12$ 3. Mengkalkulasi *Shared Secret* (S_{AB}) menggunakan $A^b \ mod \ p = S_{AB}$ $6^{13} \bmod 17 = 10$

Pada tabel 3.2 dapat disimpulkan bahwa Ali dan Boy memiliki nilai *Shared Secret* yang sama yaitu 10. Nilai tersebut didapat dari pertukaran *public key* Ali dan Boy yang berasal dari *private key* masing masing. Sebelum memulai perhitungan, Ali dan Boy menentukan *public value* yang akan digunakan. Ali dan Boy akan menggunakan *public value* dengan nilai yang sama.



Gambar 3. 3 Diagram Alur *Diffie Hellman*

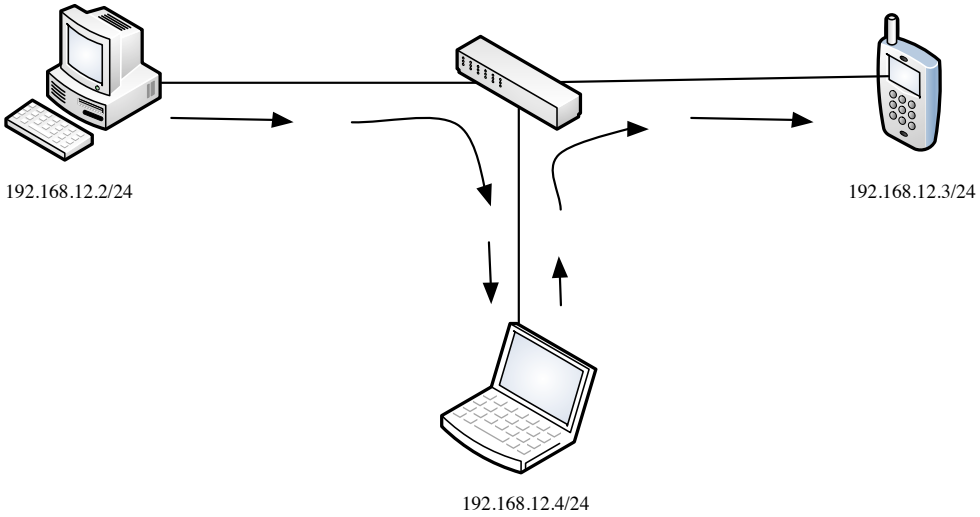
Gambar 3.3 memperlihatkan diagram alur *Diffie Hellman* dari *User* B terhadap *User* A. *DH key exchange* memang dirancang untuk melakukan pertukaran kunci pada sesi yang tidak aman. Dari seluruh deskripsi pada Tabel 3.1 dan Gambar 3.3, tidak terdapat fungsi untuk menentukan apakah Boy memang tujuan dari Ali. Sehingga dimungkinkan untuk terjadi penyerangan *MiTM* terhadap DH *key exchange*. Pada protokol SRTP yang menjalankan metode DH *key exchange* pun tidak terdapat fungsi untuk memverifikasi tujuan dari masing masing *endpoint.* Ini merupakan celah yang dapat dimanfaatkan oleh penyerang

3.1.2 Analisis Metode Serangan

Penyadapan VoIP harus dilakukan secara *realtime* ketika percakapan VoIP terjadi, karena ketika percakapan terputus tidak ada data yang dikirimkan di jaringan. Untuk membaca percakapan tersebut, penyerang harus dapat menginterfensi pesan percakapan tanpa diketahui pengguna. Teknik penyerangan

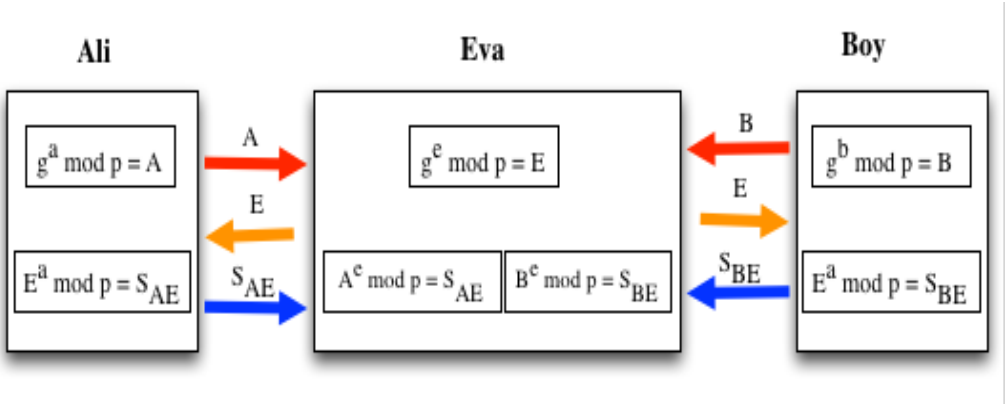
ini disebut *Man in The Middle Attack.*

Metode penyerangan *Man in The Middle (MiTM) Attack*, memposisikan penyerang sebagai relay diantara komunikasi VoIP, membaca pesan lalu meneruskannya.



Gambar 3. 4 Penyerangan *Man in The Middle*

Penyerangan MiTM pada VoIP tanpa sistem keamnan terjadi karena data yang dikirimkan tidak melaui proses *kriptografi*. Sedangkan pada SRTP yang merupakan protokol keamanan untuk VoIP, penyerangan terjadi karena memanfaatkan celah pada DH *key exchange.* Penyerang dapat mengetahui *shared secret* karena penyerang berperan sebagai *relay* pada saat proses *key exchange terjadi*. Pada Gambar 3.5 diilustrasikan penyerangan terhadap DH *key exchange*.



Gambar 3. 5 Penyerangan *MiTM* pada *DH Key Exchange*

Pada Gambar 3.5, Eva berperan sebagai penyerang pada komunikasi antara Ali dan Boy. Untuk mendapatkan *shared secret* dari Ali dan Boy, Eva hanya perlu menukar nilai awal Ali dan Bob dengan miliknya. Untuk membuktikan maka akan dilakukan simulasi penyerangan terhadap metode *Diffie Hellman.* Nilai public value yang digunakan yaitu p=17 dan g=3

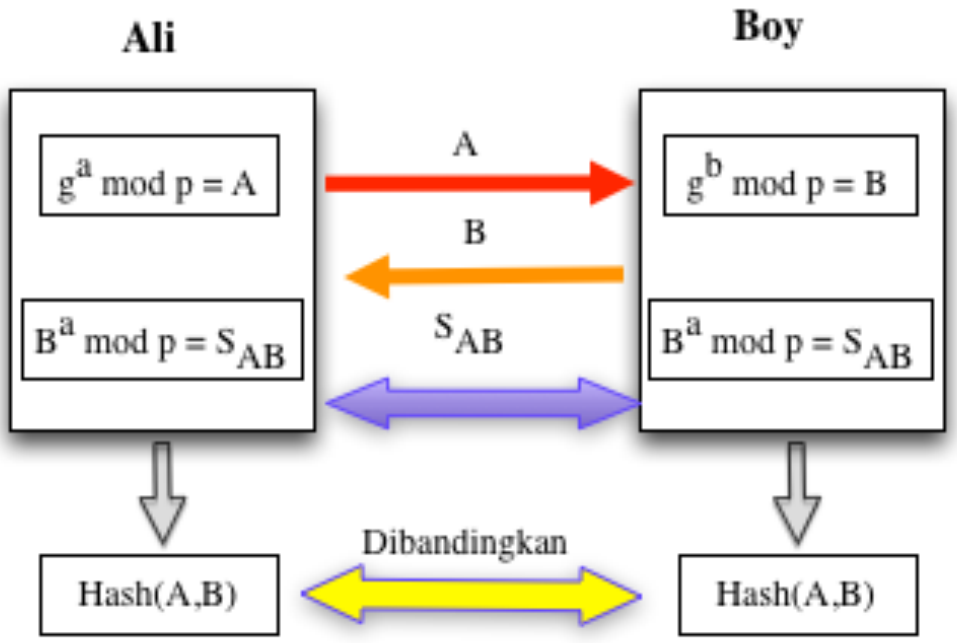
Tabel 3. 3 Simulasi Penyerangan *Diffie Hellman*

Ali (A)	Eva (E)	Boy (B)
1. Private key a=15 2. Menghitung nilai A $g^a \bmod p = A$ $3^{15} \bmod 17 = 6$ 3. Mengkalkulasi *Shared Secret* (S_{AE}) menggunakan $B^a \bmod p = S_{AE}$ $4^{15} \bmod 17 = 13$	1. Private key e=12 2. Menghitung nilai E $g^e \bmod p = E$ $3^{12} \bmod 17 = 4$ 3. - Mengkalkulasi *Shared Secret* (S_{AE}) menggunakan $A^e \bmod p = S_{AE}$ $6^{12} \bmod 17 = 13$ - Mengkalkulasi *Shared Secret* (S_{BE}) menggunakan $B^e \bmod p = S_{BE}$ $12^{12} \bmod 17 = 4$	1. Private key b=13 2. Menghitung nilai B $g^b \bmod p = B$ $3^{13} \bmod 17 = 12$ 3. Mengkalkulasi *Shared Secret* (S_{BE}) menggunakan $E^b \bmod p = S_{BE}$ $4^{13} \bmod 17 = 4$

Tabel 3.3 disimulasikan penyerangan terhadap metode *Diffie Hellman* , dimana Eva berperan sebagai penyerang. Eva menerapkan serangan *MiTM* dengan mengikuti proses yang sedang berjalan dan melakukan *relay*. Sehingga Ali dan Boy menghitung *shared secret* bukan berdasar pada private key Ali atau Boy melainkan menggunakan private key Eva. Hal tersebut bisa terjadi karena *DH* tidak memiliki fungsi untuk authentikasi tujuan/*endpoint*. Proses authentikasi *endpoint* pun tidak terjadi pada protokol SRTP tempat proses DH *key exchange* ini berjalan. Maka diperlukan metode lain untuk memverifikasi kedua *endpoint* tersebut.

3.1.3 Analisa Short Authentication String (SAS)

SAS diperlukan sebagai metode untuk memverifikasi kedua *endpoint* pada proses DH *key exchange*. Pada Gambar 3.6, diperlihatkan cara kerja SAS. Nilai SAS yang ditampilkan ke pengguna didapatkan dari hasil *hash* nilai A dan B, bukan dari *shared secret* A dan B.



Gambar 3. 6 SAS pada *DH key exchange*

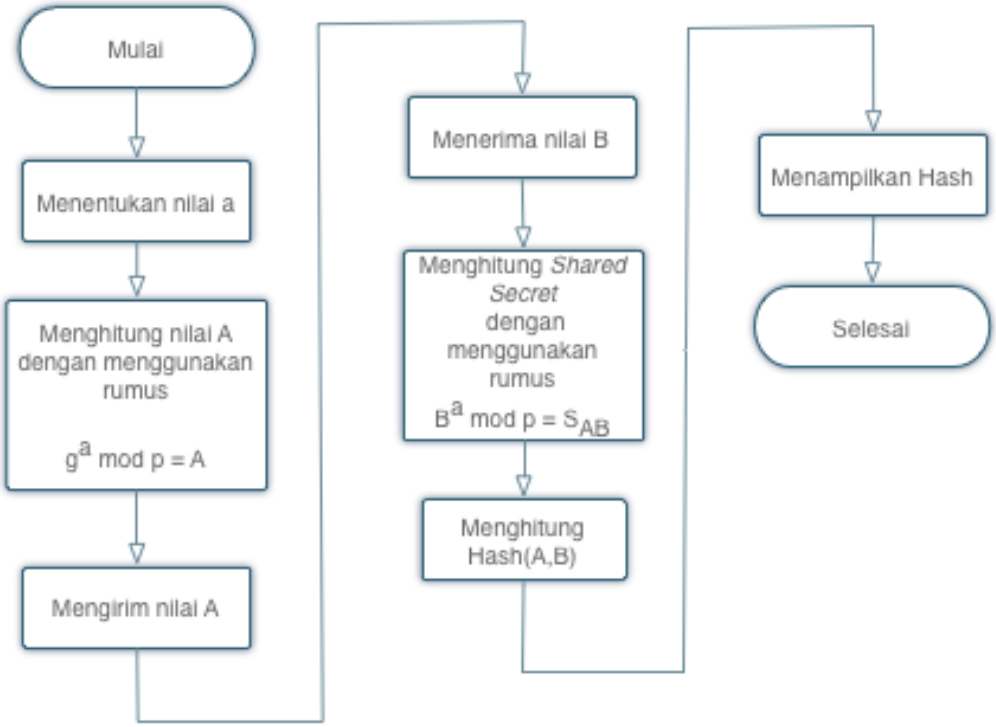
Tabel 3.4 menerangkan simulasi SAS pada *Diffie Hellman*. Fungsi Hash dihitung dari nilai perkalian antara *public key* kedua *endpoint* lalu di encode menggunakan Base32.

Tabel 3. 4 Simulasi SAS pada DH

Ali	Boy
1. Private key a=15 2. Menghitung nilai A $g^a \bmod p = A$ $3^{15} \bmod 17 = 6$ 3. Mengkalkulasi *Shared Secret*	1. Private key b=13 2. Menghitung nilai B $g^b \bmod p = B$ $3^{13} \bmod 17 = 12$ 3. Mengkalkulasi *Shared Secret*

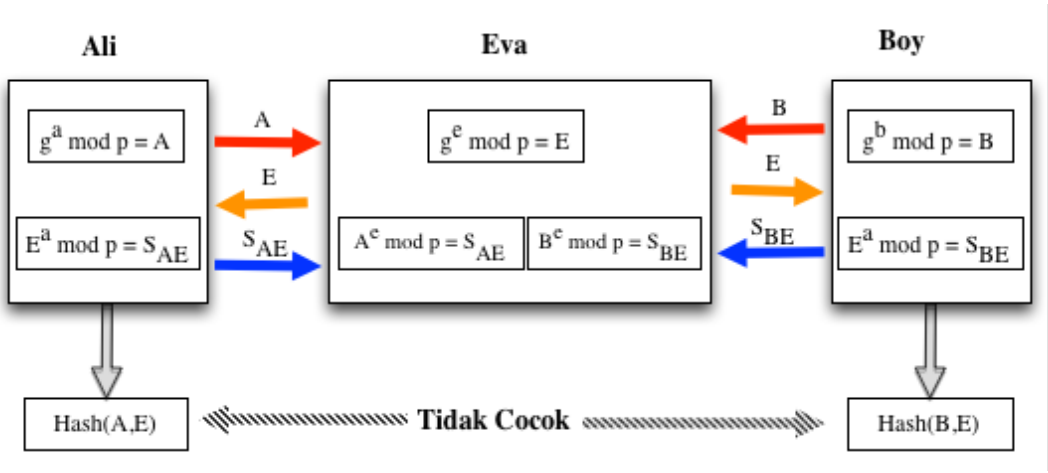
(S_{AB}) menggunakan $B^a \ mod \ p = S_{AB}$ 12^{15} mod 17 = 10 4. Menghitung SAS *Hash (A,B)* Hash (6,12) = G4ZA====	(S_{AB}) menggunakan $A^b \ mod \ p = S_{AB}$ 6^{13} mod 17 = 10 4. Menghitung SAS *Hash (A,B)* Hash (6,12) = G4ZA====

Pada Gambar 3.7, diilustrasikan diagram alur SAS pada *Diffie Hellman* dari user B ke user A. SAS bertugas mengambil nilai A dan B yang merupakan hasil perhitungan dari *Diffie Hellman*. Nilai A dan B akan digunakan untuk menghitung hash. Untuk lebih jelas implementasinya proses tersebut dapat dilakukan simulasi. Setelah hash didapatkan, hash tersebut akan ditampilkan untuk diverifikasi oleh masing masing pengguna melalui suara. Metode SAS digunakan untuk menutupi kekurangan dari metode *Diffie Hellman key exchange* dan SRTP yang tidak dapat memverifikasi masing masing *endpoint*. Jika nilai SAS berbeda, maka dapat disimpulkan telah terjadi serangan *MiTM.*



Gambar 3. 7 Diagram Alur SAS

Pada Gambar 3.8 diilustrasikan penyerangan MiTM pada *Diffie Hellman key exchange* yang telah dilengkapi metode SAS. Pada Gambar 3.8, Ali akan berkomunikasi dengan Boy dan Eva akan melakukan penyadapan pada komunikasi tersebut. Eva melakukan serangan MiTM dengan berperan sebagai relay pada proses *key exchange* dan berhasil mendapatkan *shared secret* Ali dan Eva. Lalu saat awal komunikasi, SAS akan ditampilkan Ali dan Eva saling mengkonfirmasi dan mendapatkan hasil SAS yang berbeda. Komunikasi VoIP akan terus berlangsung, namun Ali dan Eva mendapatkan peringatan bahwa telah terjadi penyadapan berdasarkan hasil SAS yang berbeda. Walaupun Eva telah mendapatkan shared secret, SAS menjadi berbeda karena SAS dihitung berdasarkan nilai dari *preshared key initiator* dan *responder*. Dalam kasus tersebut, nilai *initiator* benar namun nilai *responder* salah karena nilai *responder* telah berubah menjadi nilai Eva. Sehingga Ali dan Boy, menghitung SAS berdasarkan nilai Eva bukan berdasarkan nilai Ali atau Boy.



Gambar 3. 8 SAS pada penyerangan *MiTM*

Pada Tabel 3.5 menerangkan simulasi terhadap penyerangan *MiTM* yang telah menggunakan SAS. SAS dibuat berdasarkan kepada public key dari kedua endpoint. Ketika terjadi penyerangan MiTM, public key yang diterima berbeda karena Eva melakukan penukaran public key miliknya. Sehingga perhitungan SAS dibuat berdasarkan public key milik Eva dan masing masing endpoint. Hal tersebut yang menyebabkan Ali dan Boy memiliki nilai Hash yang berbeda.

Tabel 3. 5 Simulasi SAS terhadap penyerangan *MiTM*

Ali (A)	Eva (E)	Boy (B)
1. Private key a=15 2. Menghitung nilai A $g^a \bmod p = A$ $3^{15} \bmod 17 = 6$ 3. Mengkalkulasi *Shared Secret* (S_{AE}) menggunakan $B^a \bmod p = S_{AE}$ 4. Membuat Hash Hash(A,E) Hash(6,4)= **GI2A====**	1. Private key e=12 2. Menghitung nilai E $g^e \bmod p = E$ $3^{12} \bmod 17 = 4$ 3. - Mengkalkulasi *Shared Secret* (S_{AE}) menggunakan $A^e \bmod p = S_{AE}$ $6^{12} \bmod 17 = 13$ - Mengkalkulasi *Shared Secret* (S_{BE}) menggunakan	1. Private key b=13 2. Menghitung nilai B $g^b \bmod p = B$ $3^{13} \bmod 17 = 12$ 3. Mengkalkulasi *Shared Secret* (S_{BE}) menggunakan $E^b \bmod p = S_{BE}$ $4^{13} \bmod 17 = 4$ 4. Membuat Hash Hash(B,E) Hash(12,4)= **GQ4A====**

Ali (A)	Eva (E)	Boy (B)
	$B^e \bmod p = S_{BE}$ $12^{12} \bmod 17 = 4$	

3.1.4 Analisis Metode Pengamanan VoIP

Pada sistem pengamanan VoIP, terdapat tiga metode yang dapat digunakan, yaitu;

1. TLS (*Transport Layer Security*), menyamarkan layer protokol, membuat protokol voip menjadi sulit untuk dideteksi dan di blok pada jaringan. Sertifikat yg dibuat berisi *public* dan *private key*. Kelemahan nya sertifikat yg digunakan oleh A ke B, dapat dicuri dan digunakan untuk membuka koneksi.
2. SRTP (*Secure Realtime Transport Protocol*), mengenkripsi suara dari *end client* ke *server*. tapi yang memiliki *server* dapat mendengarkan/membaca suara tsb. Terdapat juga kelemahan pada *DH key exchange* yang tidak dapat meng-authentikasi *endpoint*.
3. ZRTP (*Zimmerman Realtime Transport Protocol*), menggunakan *public key* kriptography untuk mendapatkan *end-to-end enkripsi*. sehingga menutup kemungkinan penyerangan menggunakan metode *Man In The Middle Attack*. ZRTP juga menutup celah pada *DH key exchange* dengan menggunakan metode *SAS (Short Authentication String*).

Pemilihan terhadap ZRTP sebagai pengaman komunikasi pada VoIP pada penelitian ini didasarkan atas kemampuannya untuk mendeteksi serangan *MiTM* dengan metode SAS. Selain itu enkripsi yang dilakukan oleh ZRTP terjadi secara *end-to-end* atau *peer-to-peer*, sehingga hanya kedua endpoint dapat mengamankan komunikasi secara langsung tanpa memerlukan pihak ketiga.

3.1.5 Analisis ZRTP

Zimmerman Realtime Transport Protocol (ZRTP) merupakan protokol yang menegosiasikan *keys* atau kunci dan informasi yang dibutuhkan untuk membuat

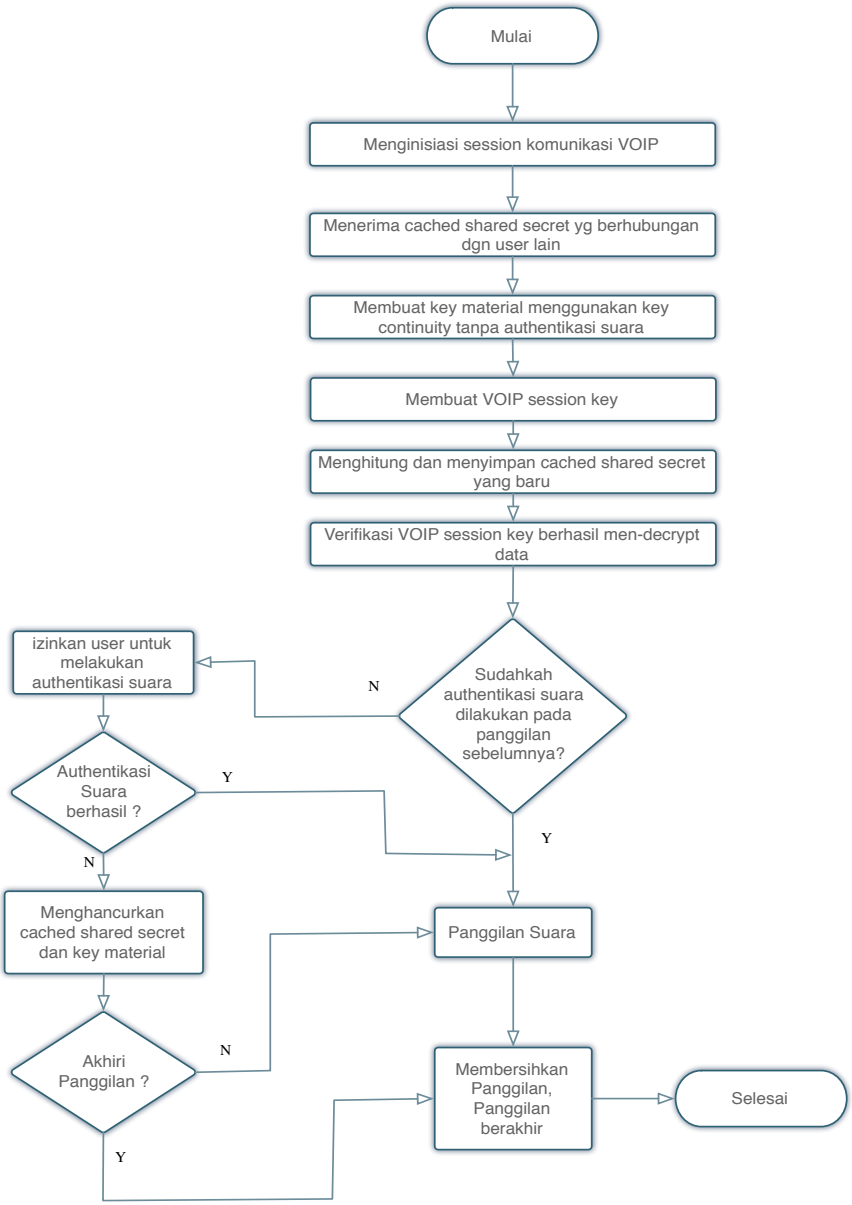
komunikasi SRTP.

Kelemahan *Diffie Hellman* yang membuat *Master Key* pada SRTP dalam menanggulangi serangan *MiTM* menjadi landasan perlunya penambahan metode pengamanan pada komunikasi VoIP.

Terdapat tiga langkah utama yang dilakukan ZRTP dalam pengamanan komunikasi VoIP, yaitu

1. Mendeteksi apakah pasangan mendukung protokol ZRTP
2. *Key Agreement Fase,* melakukan pertukaran informasi shared key antar host
3. *Secure Fase*, mengkonfirmasi data kriptografi dan meneruskan nya ke mode SRTP.

Pada Gambar 3.9 akan memperlihatkan diagram alur cara kerja ZRTP



Gambar 3. 9 Alur Cara Kerja ZRTP

Pada Gambar 3.9 diperlihatkan alur kerja dari ZRTP secara umum. Proses dimulai dengan melakukan pertukaran kunci, lalu menyimpan nya. Kunci yang disimpan tersebut akan digunakan untuk kembali pada panggilan berikutnya. Setelah memeriksa apakah telah tersimpan kunci dari sesi sebelumnya, bila tidak ada maka akan dibuat kunci baru. Dalam pembuatan kunci tersebut terdapat

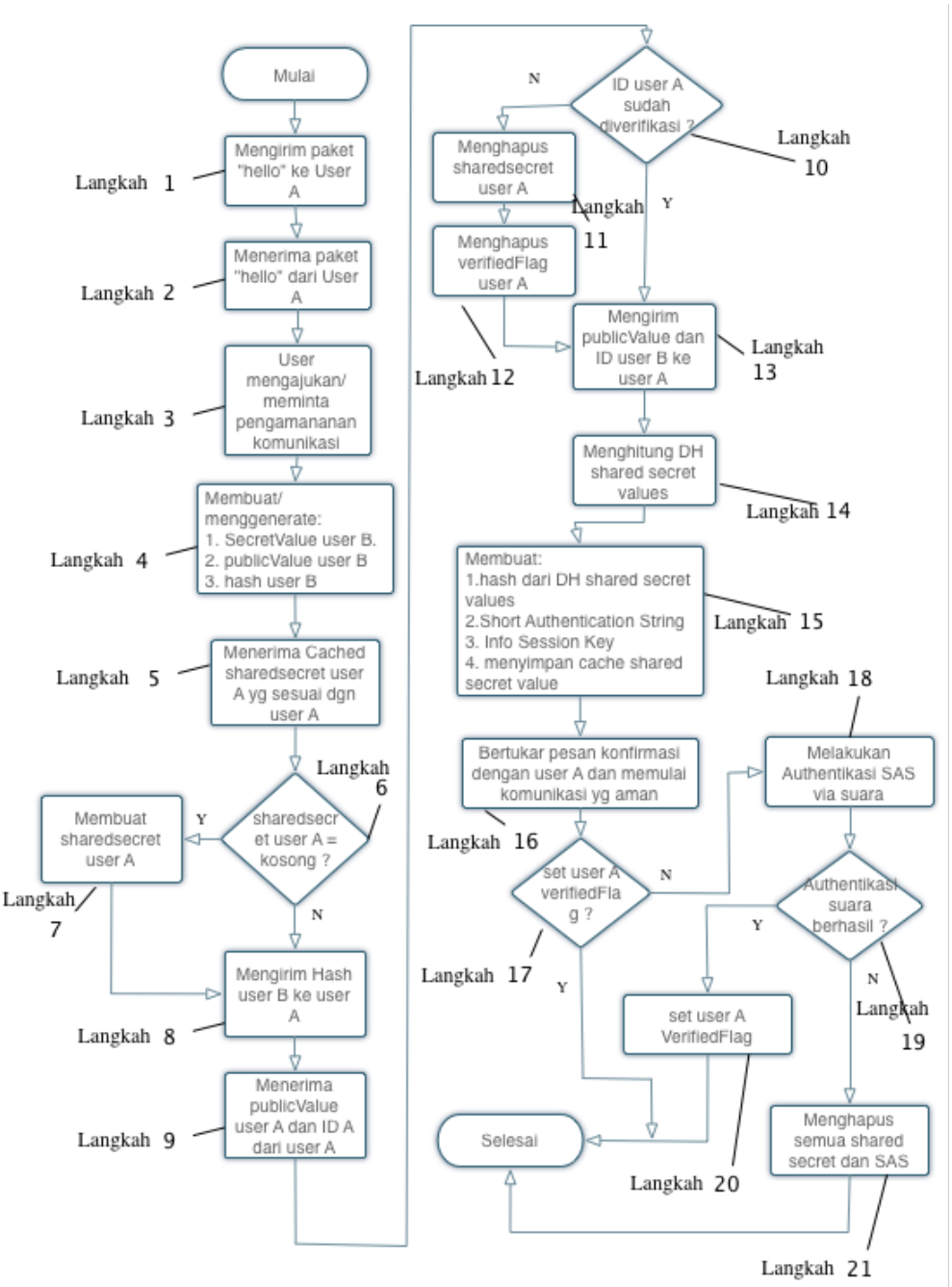
verifikasi kembali menggunakan metode SAS. Ketika SAS berhasil diverifikasi panggilan suara akan mulai diamankan.

Pada Gambar 3.10, merupakan gambaran proses secara lebih detail. Pada gambar tersebut terdapat langkah 1-21 yang menunjuk setiap proses. Gambar 3.10 mengilustrasikan panggilan dari *user* B ke *user* A. Langkah 1-2 merupakan proses dari *Hello Message* dan *HelloACK Message* yang berfungsi untuk memeriksa dukungan terhadap koneksi ZRTP dari setiap *endpoint*.

Langkah 3 menunjukan proses *Commit Message* yang akan menginisiasi *key agreement*. Terdapat tiga jenis *Commit Message* yaitu DH *format*, *Multistream format* dan *Preshared format*. Dalam penelitian ini menggunakan *DH mode*, maka *Commit Message* yang memiliki format DH yang akan dikirim.

Selanjutnya pada langkah 4, user B akan mempersiapkan *key material* yang diperlukan untuk proses *DH key exchange* dan mengirimkan nya pada *DHpart1 message*. Pada langkah 5, *user* B menerima *DHpart1 message* dari *user* A. *User* B akan memeriksa (langkah 6) apakah *shared secret user* A telah disimpan atau belum, jika belum maka *user* B akan menyimpan nya (langkah 7). Lalu akan mengirimkan *hash* kepada *user* A (langkah 8) dan menerima *publicvalue* (langkah 9) dan ID dari *user* A. ID dari *user* akan akan diperiksa apakah sudah diverifikasi atau belum (langkah 10). Jika belum, maka *shared secret* (langkah 11) dan *verifiedflag* (langkah 12) dari *User* A akan dihapus. *PublicValue* dan ID *user* B (langkah 13) akan dikirim ke *User* A untuk dilakukan proses seperti pada langkah 10-12.

Pada langkah 14 dan 15, *user* akan mengkalkulasikan *key material* yang telah didapat dari *hash*, *sharedsecret* dan *publicvalue* dari *User* A dan *User* B lalu dikirim sebagai *DHpart2 message*. Pada langkah 16, *User* akan saling bertukar *Confirm1* dan *Confirm2 message*. Proses authentikasi SAS (langkah 18) akan dilakukan ketika *User* A belum memiliki *verifiedflag*. Jika verifikasi berhasil maka User A akan memiliki nilai *verifiedflag* (langkah 17 dan 20), dan jika gagal maka seluruh *shared secret* akan dihapus (langkah 21).



Gambar 3. 10 Detail Alur ZRTP

3.1.6 Analisis VoIP Server

VoIP Server atau *SIP registar* berfungsi sebagai *server* yang menghubungkan komunikasi VoIP. Terdapat dua cara untuk mengimplementasikan ZRTP sebagai pengamanan data VoIP, yaitu;

1. *Private Server*, VoIP server dikelola sendiri dengan menggunakan open source VoIP *server*. Karena VoIP *server* dikelola sendiri maka dapat mengurangi kemungkinan penyerangan dari pemilik *server*. Contoh dari open source VoIP *server* yaitu; *asterisk*, *freeswitch*, dan *gnusipwitch*. Dalam impelmentasi nya, VoIP *server* dapat tergabung dalam paket VoIP management dan *PBX (Private Branch Exchange*) seperti *trixbox* dan *freepbx*.
2. *Public Server*, menggunakan VoIP *server* yang dikelola oleh pihak luar. Standar keamanan sudah menggunakan ZRTP dan SRTP, namun karena pengelolaan dipegang pihak luar ketersediaan layanan dan keamanan menjadi pertanyaan. Contoh public server yaitu, *ekiga* dan *ostel*.

Untuk mengurangi kemungkinan penyadapan dari *server* maka pada penelitian ini menggunakan *private server* sebagai cara untuk mengimplemetasikan VoIP *Server*.

Tidak semua VoIP *server* telah mendukung protocol ZRTP, oleh karena itu diperlukan analisa untuk mencari VoIP *server* yang telah mendukung ZRTP. ZRTP merupakan authentication protocol yang bersifat *peer-to-peer* (P2P), oleh karena itu ZRTP hanya memerlukan *signalling information* yang tepat agar dapat memulai *encryption key exchange.* Tabel 3.6 menampilkan perbandingan VoiP Server dan dukungannya terhadap pengamanan menggunakan ZRTP

Tabel 3. 6 Perbandingan VoIP Server

Kriteria	Asterisk	Freeswitch	GNU Sipwitch
Sistem Operasi	Linux BSD, Mac OS X, Solaris	Linux BSD, Mac OS X , Solaris, Windows	Linux BSD

Kriteria	Asterisk	Freeswitch	GNU Sipwitch
Lisensi	GPL Propietary	Mozilla Public License	GNU GPL 3.0
Control Panel	Freepbx, Trixbox	-	-
Dukungan ZRTP	Membutuhkan Patch ZRTP	Membutuhkan Library ZRTP	Mendukung

Pada awal penelitian, *Trixbox* 2.8 dengan *asterisk* 1.6 dipilih sebagai VoIP *server* dengan pertimbangan kemudahan instalasi, konfigurasi serta pengoperasian VoIP *Server*. Namun setelah dilakukan ujicoba, koneksi ZRTP tidak bisa dilakukan, dikarenakan *asterisk* 1.6 menggunakan RTP stream untuk mengirimkan file media sedangkan proses ZRTP *key exchange* terjadi pada RTP *stream*. Bukan berarti *Asterisk* tidak mendukung ZRTP, tetapi *Asterisk* menggunakan jalur yang digunakan oleh ZRTP untuk mengirim informasi ke *endpoint*.

Sama halnya dengan *Astersik*, *freeswitch* memiliki permasalahan yang sama. Namun pada *freeswitch* proses upgrade harus dilakukan pada bagian *source code*, sehingga memerlukan libzrtp pada proses *compile* aplikasi. Maka diperlukan patch ZRTP pada Asterisk dan libzrtp pada freeswitch, namun ketika penelitian ini dilakukan patch ZRTP sudah tidak tersedia dikarenakan Phill Zimmerman (Pembuat ZRTP), mengarahkan implementasi ZRTP kearah *Silent Circle*. *Silent Circle* merupakan sebuah produk komunikasi milik Phil Zimmerman yang mengedepankan keamanan sebagai fitur utamanya.

Asterisk dan *Freeswitch* sampai penelitian ini dilakukan akhirnya tidak dapat mendukung ZRTP dikarenakan tidak tersedia patch dan *library* ZRTP. Selanjutnya untuk penelitian akan menggunakan GNU *Sipwitch* yg sudah mendukung ZRTP. GNU *Sipwitch* hanya memiliki fitur *SIP registar*, tidak memiliki fitur media seperti *Asterisk* dan *Freeswitch*. GNU *sipwitch*

menggunakan GNU ZRTP yang merupakan penerapan dari ZRTP dibuat menggunakan C++.

3.1.7 Analisis VoIP Client

Untuk melakukan komunikasi VoIP, diperlukan VoIP *Client* pada sisi pengguna. VoIP *Client* saat ini telah tersedia untuk mobile dan desktop.

1. VoIP *Client Desktop*, *X-lite* dan *Jitsi*. *Xlite* belum mendukung pengamanan dengan ZRTP, diperlukan *plugin Zfone* agar dapat menggunakan ZRTP. Phil Zimmerman sebagai pembuat *Zfone* sudah tidak lagi menyediakan *Zfone*. VoIP client selanjunya yaitu Jitsi yang sudah mendukung protokol ZRTP. Jitsi menjadi pilihan VoIP *Client* pada desktop untuk penelitian ini.
2. VoIP *Client Mobile*, *CSipsimple* (*Android*) merupakan aplikasi VoIP client yang sudah mendukung ZRTP. Pada beberapa versi sistem operasi android, sudah ada aplikasi *default* yang dapat digunakan untuk komunikasi VoIP. Namun belum mendukung protokol ZRTP.

Pada penelitian ini akan menggunakan Jitsi sebagai aplikasi VoIP *Client*. Aplikasi *CSipsimple* digunakan untuk menguji komunikasi VoIP menggunakan ZRTP antara *desktop* dan mobile.

3.1.8 Analisis Perangkat Keras

Kebutuhan perangkat keras untuk server dan client yaitu dengan spesifikasi sebagai berikut

Tabel 3. 7 Kebutuhan Server dan Client

Perangkat Keras	*Client*	*Server*
Processor	Dualcore 2.4 Ghz	Dualcore 2.4 GHz
Memory	1024 MB	512 MB
Harddisk	20 GB	10 GB

Perangkat Keras	*Client*	*Server*
Sistem Operasi	Windows 7	Ubuntu 12.04

3.1.9 Analisis Perangkat Lunak

Kebutuhan perangkat lunak yang akan digunakan dalam penerapan ZRTP pada *server, client* dan penguji.

Tabel 3. 8 Kebutuhan Perangkat Lunak

Nama Perangkat	Perangkat Lunak	Keterangan
VoIP *Server*	GNU Sipwitch	GNU Sipwitch digunakan sebagai *SIP Registar* yang mendukung ZRTP
Client	Jitsi	Jitsi digunakan sebagai VoIP *Client*
Penguji	Arpspoof	*Arpspoof* digunakan untuk penyerangan terhadap ARP
	Wireshark	Wireshark digunakan sebgai packet *sniffer*, *RTP decoder* dan *RTP Player*

3.2 Perancangan Sistem

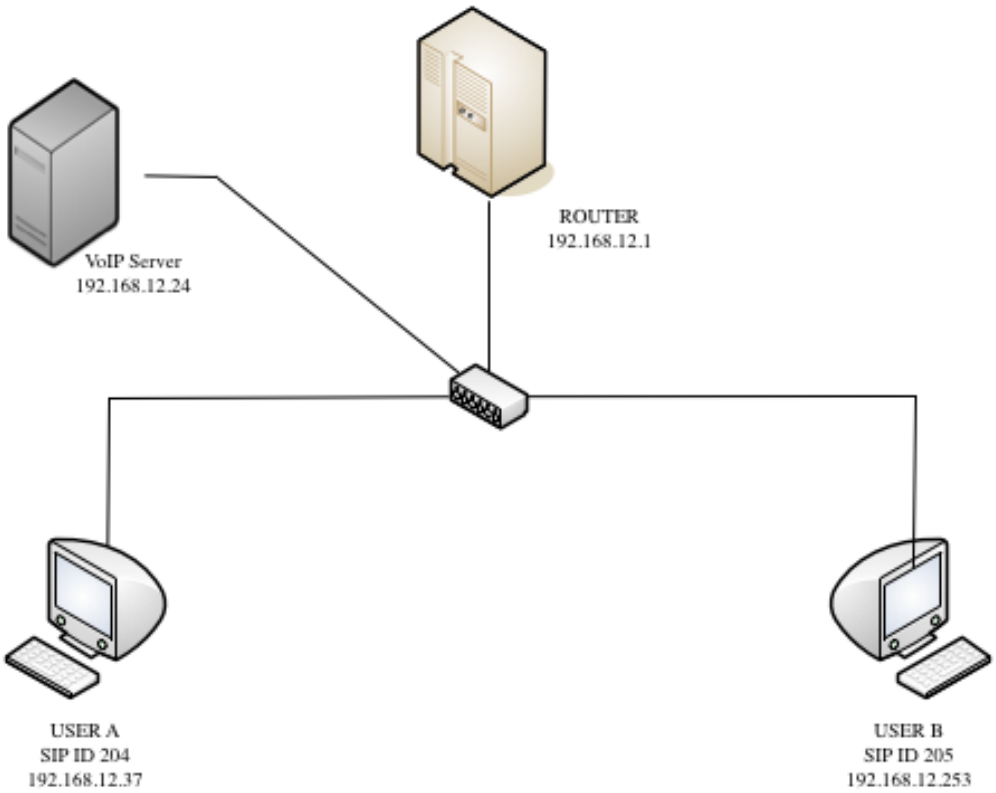
Pada perancangan sistem akan dipaparkan mengenai, perancangan arsitektur sistem, perancangan arsitektur pengujian dan beberapa perancangan lainya.

3.2.1 Tujuan Perancangan Sistem

Perancangan sistem merupakan suatu proses tindak lanjut dari tahap analisis. Setelah mendapat gambaran umum mengenai sistem pada tahap analisis lalu penelitin melanjutkan dengan perancangan sistem yang bertujuan untuk memberikan gambaran sistem yang akan dibuat.

3.2.2 Perancangan Aristektur

Perancangan arsitektur pada Gambar 3.11 dibuat untuk memberikan gambaran arsitektur infrastruktur yang digunakan untuk membuat komunikasi VoIP .



Gambar 3. 11 Arsitektur VoIP

Pada Tabel 3.4 menunjukan daftar perangkat yang berada pada perancangan arsitektur simulasi beserta IP address nya.

Tabel 3. 9 Daftar IP Address Perangkat

IP Address	SIP ID	Keterangan
192.168.12.1	-	Router
192.168.12.24	-	VoIP Server
192.168.12.37	204	User A
192.168.12.253	205	User B

3.2.3 Perancangan Pengalamatan SIP ID

SIP ID merupakan identifier unik yang digunakan sebagai alamat dari setiap *VoIP Account*. Seperti halnya IP Address pada jaringan, kesamaan SIP ID dapat menyebabkan gangguan pada sistem VoIP.

3.2.3.1 Perancangan SIP ID Pada *Server*

Pemberian SIP ID pada *client* diawali dengan membuat *account* pada *server*. Penambahan *account* pada *server* dilakukan dengan menambahkan konfigurasi pada file "/etc/sipwitch.conf". File tersebut memiliki struktur XML, tabel 3.5 menampilkan konfigurasi yang dilakukan beserta penjelasan nya.

Tabel 3. 10 Perancangan *SIP ID* pada sipwitch.conf

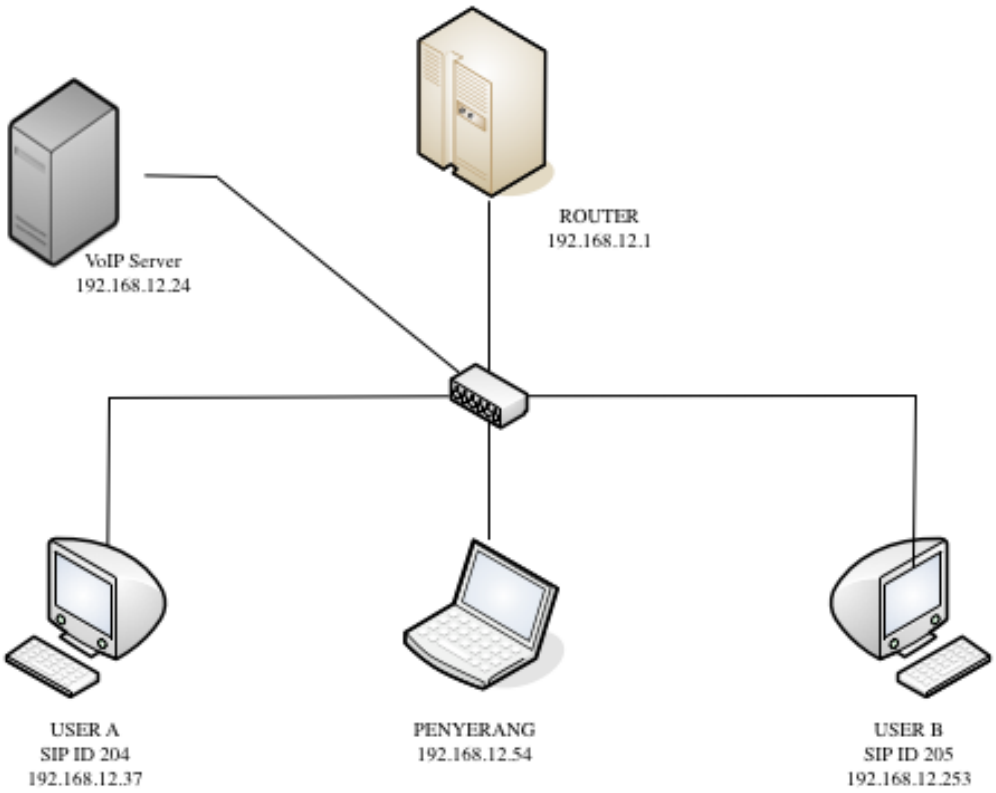
Perintah	User A	User B	Keterangan
<user id>	rahmat1	rahmat2	User id yang merupakan *unique identifier*
<secret>	rahmat1666	rahmat2666	Password dari masing masing account
<extension>	204	205	Extension yang akan digunakan untuk memanggil
<display>	rahmat1	rahmat2	Nama yang akan ditampilkan

3.2.3.2 Perancangan SIP ID Pada *Client*

Pemberian SIP ID pada client diawali terlebih dahulu dengan mendaftarkan SIP account pada server. Setelah SIP account didaftarkan, client yang menggunakan aplikasi *Jitsi* sebagai voip client nya dapat menggunakan account dengan memakai SIP ID dan *secret.* Format yang digunakan dalam konfigurasi SIP ID yaitu "*userid@servername*". *Variabel userid* diganti dengan *user id* dari SIP dan *servername* diganti dengan alamat VoIP server. Contoh konfigurasi pada User A yaitu "rahmat1@192.168.12.24" .

3.2.4 Perancangan Arsitektur Pengujian

Perancangan aristektur pengujian pada Gambar 3.12 untuk menguji keamanan pengguna VoIP terhadap penyadapan menggunakan metode serangan *Man In The Middle Attack* . Pengujian dilakukan terhadap dua sistem voip, sistem voip yang tidak menggunakan pengamanan ZRTP dan sistem voip yang menggunakan pengamanan ZRTP.



Gambar 3. 12 Arsitektur Pengujian

Pada Gambar 3.13 menunjukan ilustrasi dari penyerangan. Penyerang melakukan arp spoofing terhadap User A untuk mengelabui *Address Resoulution Protocol* pada User A. Sehingga User A menganggap sedang berkomunikasi dengan User B. Setelah itu, Penyerang melakukan *packet sniffing* untuk melihat paket SIP dan RTP yang berada pada User A.



Gambar 3. 13 Ilustrasi Pengujian

Pengujian dilakukan menggunakan tiga skenario. Skenario tersebut dibuat dengan tujuan untuk membuktikan keamanan VoIP pada sistem dan cara penyerangan yang berbeda yang ditampilkan pada Tabel 3.11

Tabel 3. 11 Skenario Pengujian

No	Sistem VoIP	Waktu Penyerangan
1	Tidak menggunakan ZRTP	Sebelum komunikasi VoIP berlangsung
2	Tidak menggunakan ZRTP	Pada saat komunikasi VoIP berlangsungx
3	Menggunakan ZRTP	Sebelum komunikasi VoIP berlangsung
4	Menggunakan ZRTP	Pada saat komunikasi VoIP berlangsung