Faculdade de Engenharia da Universidade do Porto

Plataforma de comunicação multimédia para dispositivos de baixo custo com o uso de protocolos seguros

Diogo Lima Monteiro Costa Leite

Mestrado Integrado em Engenharia Informática e Computação

Empresa: VisionSpace Technologies, Lda.

Orientador: José Manuel De Magalhães Cruz

Junho de 2014

Plataforma de comunicação multimédia para dispositivos de baixo custo com o uso de protocolos seguros

Diogo Lima Monteiro Costa Leite

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Daniel Augusto Gama de Castro Silva (Prof. Doutor)

Vogal Externo: Marco Paulo Amorim Vieira (Prof. Doutor)

Orientador: José Manuel De Magalhães Cruz (Prof. Doutor)

_____

14 de Julho de 2014

Resumo

Este projeto tem como finalidade o desenvolvimento de uma plataforma de comunicação que permita aos seus utilizadores trocarem entre si áudio e vídeo. É um requisito da aplicação a utilização de um dispositivo de baixo-custo, como Raspberry-Pi, para as comunicações, correndo sobre um sistema operativo Linux, e ainda a implementação na linguagem Java.

Outro requisito importante, passa por garantir a segurança de toda a informação trocada e dos dados dos utilizadores, nos aspetos de confidencialidade e integridade. A segurança é também essencial no que toca à comercialização e promoção do próprio produto, pois torna-o apetecível aos olhos do utilizador/consumidor. Durante o processo de desenvolvimento serão usados protocolos já existentes para a realização de comunicações multimédia sendo estes Session Initiation Protocol (SIP) sobre Transport Layer Security (TLS) e Secure Real-Time Transport Protocol (SRTP), sendo o primeiro necessário para a realização da fase de estabelecimento de sessão e o segundo para a transferência dos dados multimédia.

Perante os requisitos da aplicação foi atingido o objetivo principal de obter uma aplicação funcional onde fosse possível realizar comunicações multimédia, sendo estas focadas em transmissão de vídeo onde se faz uso dos protocolos SIP e RTP. Já em relação ao objetivo da segurança toda a aplicação encontra-se segura recorrendo ao uso do protocolo RTP com o uso do padrão Advanced Encryption Standard (AES) e com a implementação do algoritmo Diffie-Hellman para a troca de chave sem a divulgação desta em um canal publico, ficando apenas para melhoria futura, a implementação do TLS para garantir que nenhuma parte da aplicação se encontra legível para quem tentar interferir na conversa.

Abstract

This project has as a main goal, the development of a communication platform that allows its users to exchange audio and video with each other. One of the requirements of the application is the use of a low-cost device, like Raspberry-Pi, to the communications, running through Linux; besides, it is also required the implementation in Java language.

Another important requirement is to guarantee the safety in all the information exchanged and user's data, mainly in the confidentiality and integrity matters. Safety is also essential in commercialization and promotion of the product itself, since it makes him appealing for the user/consumer. During the development process it will be used already existing protocols for establishment of multimedia communication: Session Initiation Protocol (SIP) over Transport Laver Security (TLS) and Secure Real-Time Transport Protocol (SRTP). The first one will be useful to establish the session and the second one will be needed for the multimedia data transfer.

Towards the requirements for the application, the main goal was achieved, obtaining a functional application where it is possible to make multimedia communications, mainly focused on video, using SIP and RTP protocols. As for the goal of safety, the whole application is secure due to the use of the RTP protocol with the Advanced Encryption Standard (AES) and with the implementation of Diffie-Hellman algorithm for the key exchange can occur without the spreading of this type of information in a public channel. Leaving only for future improvement, the TLS implementation to guarantee that each part of the application is available for users which are external to the conversation.

Agradecimentos

Apresento os meus sinceros agradecimentos a todos aqueles que me ajudaram em todas as alturas da minha vida até chegar a esta fase. Neste processo algumas pessoas tomaram especial destaque.

Em primeiro lugar são as pessoas mais próximas que foram mais importantes para mim em todos os passos até aqui, os meus pais e a minha avó, que sempre estiveram ao meu lado a apoiar-me e a incentivarem-me, sempre me ajudaram a tomar as decisões necessárias mas deixando-me sempre escolher aquilo que pretendia.

À Flávia Sequeira agradeço todo o apoio que me deu para a finalização do meu curso, assim como a ajuda nas várias decisões e análises que foram necessárias fazer ao longo do desenvolvimento da dissertação.

Ao Professor José Manuel De Magalhães Cruz, orientador da minha dissertação de mestrado que sempre me ajudou dando sempre conselhos úteis e soluções para os problemas que tive de ultrapassar.

A todos os meus colegas que sempre me apoiaram ao longo destes muitos anos e que sempre se preocuparam comigo e a todos os momentos que passamos juntos e que tornaram todo este tempo académico único e memorável.

Conteúdo

Lista de Figuras

Lista de tabelas

Abreviaturas

AES	Advanced Encryption Standard
DH	Diffie-Hellman
IETF	Internet Engineering Task Force
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SIP	Session Initiation Protocol
SRTCP	Secure Real-time Transport Control Protocol
SRTP	Secure Real-time Transport Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
ZRTP	Z and Real-time Transport Protocol

1 Introdução

1.1 Contexto

Nos dias de hoje, cada vez mais a comunicação é realizada através de aplicações multimédia pois, é cada vez mais fácil aceder à Internet em qualquer ponto e assim realizar chamadas sem custo para o utilizador. Por outro lado cada vez mais as pessoas começam a optar por computadores de baixo-custo, e neste caso o Raspberry-Pi insere-se neste meio sendo que é um computador com características simples que possibilita um investimento reduzido para os utilizadores.

Nos últimos tempos os problemas de confidencialidade na Internet têm-se vindo a avolumar, representando assim um motivo de desconfiança para os utilizadores/consumidores de diversas aplicações; assim, neste momento, pretende-se que qualquer aplicação desenvolvida transmita ao utilizador um nível mínimo de segurança, sendo necessário o uso de protocolos seguros na sua implementação.

Este projeto insere-se no âmbito da empresa VisionSpace Technologies com o intuito de entrar no mercado das comunicações multimédia e futuramente disponibilizar aos seus clientes uma aplicação que se distinga das já existentes. Para tal são definidos dois objetivos garantir a segurança e a qualidade da comunicação, tudo isto para um dispositivo de baixo-custo que torna o produto mais apelativo.

1.2 Motivação

A motivação centra-se na criação de uma plataforma de comunicação multimédia protegida por técnicas criptográficas padrão. Pretende-se uma plataforma que dê aos seus utilizadores segurança no dia-a-dia.

É um produto com interesse atual pois hoje em dia as pessoas usam mais frequentemente software para comunicações multimédia: encontrando-se cada vez mais a trabalhar fora do seu país de origem usam formas de comunicação fácil.

Por outro lado a utilização de dispositivos Raspberry-Pi, produto de baixo custo, poderá vir a chamar mais utilizadores pois tem um investimento muito reduzido.

1.3 Objetivo

O principal objetivo deste projeto é desenvolver uma aplicação funcional de realização de comunicações multimédia para um dispositivo, de baixo custo, neste caso um Raspberry-Pi. Um segundo objetivo é o estudo dos métodos para melhorar a segurança de todas as

comunicações realizadas pelo sistema por forma a garantir a satisfação do utilizador neste aspeto.

Os dispositivos do tipo Raspberry-Pi, são dispositivos de pequeno porte com todo o hardware inserido numa única placa. A sua memória de armazenamento é apenas um cartão de memória SD e todo o restante hardware é constituído com peças não muito evoluídas mas que permitem o bom funcionamento do sistema. Todos os sistemas operativos que funcionam neste dispositivo são baseados em UNIX, sistema operativo multitarefa e multiutilizador.

A segurança nestes sistemas baseia-se em aspetos como confidencialidade, ou seja, limitar o acesso a utilizadores sem autorização; integridade, para que a informação não seja alterada por terceiros; disponibilidade, que garante que o sistema pode ser usado o máximo de tempo; autenticidade, que garante que a informação tenha origem em fontes bem determinadas.

1.4 Estrutura do documento

Para além da introdução, este relatório contém mais 4 capítulos. No capítulo 2, é descrito o estado da arte onde são apresentados os protocolos que podem ser usados para este tipo de comunicações e alguns conceitos básicos e são focados todos os protocolos que permitem realizar o estabelecimento das sessões, é também descrito o seu funcionamento e as trocas de mensagens e os protocolos de troca de dados multimédia em tempo-real onde se apresenta o que cada protocolo necessita enviar. No capítulo 3 é explicada a metodologia de desenvolvimento do projeto onde é apresentada a arquitetura seguida ao longo de projeto e toda a sua estruturação, sendo então referido os protocolos a usar e todas as suas configurações. Já no capítulo 4 é apresentada a implementação seguida, sendo referida neste o que se usou para o servidor e para o cliente, onde são referidas as API utilizadas e como foram usadas ao longo do desenvolvimento do projeto. Ainda neste capítulo foi criada um subcapítulo onde se realizam um conjunto de testes quer de qualidade quer de desempenho. Por fim no capítulo 5 são então tiradas as conclusões de todo o projeto, onde se refere o que foi feito do que estava planeado, das conclusões do uso das API utilizadas e ainda o trabalho a realizar no futuro.

2 Revisão Bibliográfica

2.1 Introdução

Nos dias de hoje já existem muitas aplicações focadas na realização de comunicações multimédia, como é o caso do Skype; porém, existem muitas formas de implementar estas aplicações através do uso de diferentes protocolos. No caso do Skype são utilizados protocolos proprietários, enquanto outras aplicações focam-se em protocolos de uso livre.

Os protocolos proprietários são desenvolvidos para um programa em específico, não sendo disponibilizadas informações para uso no desenvolvimento de outra aplicação. Por outro lado, os de uso livre são desenvolvidos para qualquer pessoa poder utilizar e até poder vir a melhorar o protocolo. Há uma grande variedade de protocolos de uso livre, uns focados no estabelecimento da comunicação inicial e posteriormente a criação do canal para a realização das comunicações multimédia e outros para a troca dos dados multimédia.

Nos protocolos para a realização da comunicação inicial temos com mais destaque o Session Initiation Protocol (SIP) [RFC 3261] que, por sua vez, disponibiliza uma variante, Secure Session Initiation Protocol (SIPS) [RFC 5630], que faz com que nas comunicações todas as mensagens enviadas sejam encriptadas com o uso do Transport Layer Security (TLS) [RFC 5246]. Para a realização das trocas de dados multimédia temos o Real-time Transport Protocol (RTP) [RFC 3550] que também tem uma variante Secure Real-time Transport Protocol (SRTP) [RFC 3711] que melhora a segurança deste protocolo. Existem também protocolos que estão desenhados de forma mais completa como é o caso do H.323 [H.323 2014] que permite realizar todas as comunicações quer de inicio de sessão quer de multimédia, pois este junta uma série de protocolos existentes como o RTP. Por fim, temos o protocolo ZRTP [RFC 6189], que realiza todas as fases da comunicação sem necessitar do uso de outros protocolos já existentes, apesar de se basear no SRTP para a transferência dos dados multimédia.

2.2 Protocolos de comunicação

Neste ponto serão apresentados os protocolos que realizam tarefas de criação de sessões, mostrando os passos que realizam e as mensagens. Dentro dos protocolos de criação de sessão apenas existe um que possui unicamente essa funcionalidade, o SIP, enquanto o outro, H.323, desempenha as funcionalidades de criação de sessão, juntamente com as configurações necessárias para a realização das comunicações multimédia.

2.2.1 Session Initiation Protocol e Transport Layer Security (SIP e TLS)

O SIP é um protocolo semelhante ao HTTP, "pedido-resposta", que providencia a configuração de uma sessão permitindo o controlo de comunicações multimédia. É um padrão da Internet Engineering Task Force (IETF), [RFC 3261].

A arquitetura do SIP é tipicamente constituída por um agente de utilizador, entidade que inicia e recebe as sessões SIP e as comunicações multimédia; um servidor de registo, responsável por registar o utilizador num domínio que depois permita o uso dos serviços disponibilizados por esse servidor. Se o utilizador que pretende realizar a chamada não tiver o endereço do utilizador a quem pretende contactar pode contactar o servidor proxy ou o servidor de reencaminhamento para o obter [Alsmairat 2009].

Para manter a segurança existem protocolos que visam proteger a integridade, confidencialidade e autenticidade da comunicação. Para isso temos dois tipos de comunicação *intra-domain* e *inter-domain*. A comunicação *intra-domain* tem apenas um domínio SIP, assim só precisa de ter um servidor proxy no meio da comunicação entre os utilizadores. Por outro lado, a comunicação *inter-domain* é adequada a situações em que ambos os utilizadores envolvidos na conversação pertençam a domínios diferentes, sendo assim preciso passar por dois servidores proxy.

O servidor proxy SIP é o componente que cria a sessão entre os utilizadores, gerando assim a chave de sessão, ou seja, um id ou valor criado para a identificação deste utilizador, fica responsável por alterar os parâmetros para situações novas e informar os utilizadores em sessão destas alterações. Este também é responsável na comunicação *inter-domain* por validar os certificados digitais de cada um.

A diferença entre o servidor proxy e o servidor de reencaminhamento é que o primeiro trata de configurar toda a sessão, ou seja, redireciona o pedido para a entidade seguinte. Enquanto o segundo retorna apenas o endereço que pretendemos obter.

Associado ao agente de utilizador, entidade que realiza o registo, existem algumas regras a seguir quanto ao início e fim das sessões. Primeiramente, é necessário esconder do utilizador os procedimentos operacionais e as especificações de segurança, limitando assim a intervenção do utilizador na configuração da sessão. É também responsável por declarar a sua segurança e capacidades operacionais ao servidor proxy e por atualizar as suas capacidades.

O agente de utilizador precisa de se registar com um domínio SIP antes de aceder aos serviços e recursos. Para manter a segurança, o servidor de registo tem a responsabilidade de providenciar ao utilizador registado um certificado digital. Assim, associa o utilizador registado a uma chave pública, sendo esta a chave partilhada por este com todos os outros onde estes poderão usar essa chave para encriptar os dados a enviar e o utilizador registado pode recorrer à sua chave privada para desencriptar e ler os dados.

A IETF definiu como implementar o SIP com a proteção TLS para resolver problemas de confidencialidade e integridade e como encriptar sessões multimédia com o SRTP. Contudo, neste momento apenas alguns clientes SIP suportam TLS [Lago-Fernández 2012]. O TLS visa proteger a confidencialidade e a integridade da comunicação, o problema é que é apenas possível implementá-lo usando o protocolo TCP, e não UDP.

O TLS é implementado em 3 fases, a fase de autenticação, a fase da geração de chaves e distribuição e a fase de confirmação. Durante a fase de autenticação o servidor proxy SIP autentica o agente do utilizador. Se a autenticação tiver sucesso o servidor proxy gera uma chave de segurança para a sessão TLS entre os agentes dos utilizadores e distribuída através da sessão SIP. Por fim os agentes de utilizadores confirmam se partilham o mesmo modo de segurança [Subramanian 2010].

Funcionamento do protocolo SIP

O funcionamento do protocolo SIP consiste na troca de mensagens simples, baseadas em mensagens de texto identificadas por uma palavra.

Este protocolo engloba sempre uma comunicação com um servidor, uma vez que não é possível efetuá-la de ponto-a-ponto, ou seja, efetuar a comunicação diretamente entre os dois utilizadores. Deste modo, a comunicação inicia-se num utilizador que manifesta vontade de contactar outro utilizador, enviando a mensagem "INVITE" onde se identifica a ele próprio e envia ao servidor o utilizador de destino. Posto isto, o servidor acede à sua base de dados de utilizadores e descobre para onde tem de reencaminhar então o pedido de contato. Sabendo o seu destino, envia então a mensagem "INVITE" que recebeu anteriormente para o destino. Se tudo correu corretamente é enviada então uma resposta de sucesso "OK" que passa pelo servidor e é reencaminhada para o utilizador que iniciou a comunicação.

Assim o utilizador que iniciou a comunicação, após obter uma mensagem de sucesso, pode então confirmar com o utilizador de destino enviando uma mensagem "ACK", e estarão prontos para iniciar as comunicações multimédia.

Todas estas etapas constituintes do protocolo SIP acima descritas encontram-se ilustradas na figura 1.

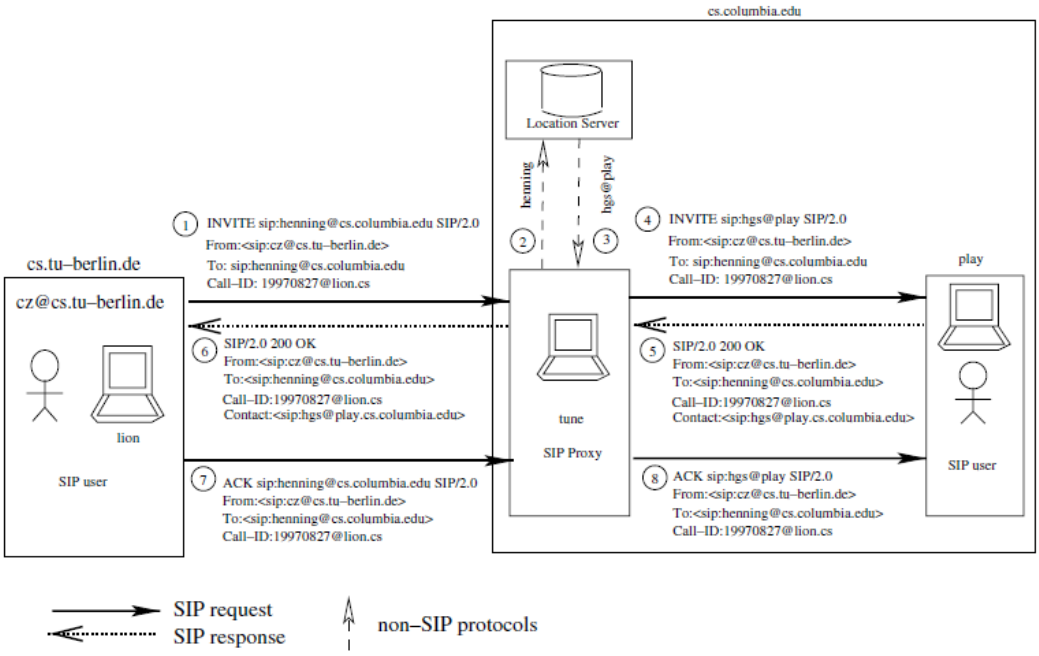

Figura 1- Funcionamento do protocolo SIP (in [Wang 2004])

O protocolo SIP emprega diferentes mensagens pré-definidas para a realização das trocas de mensagens, tal como podemos ver exemplificado de seguida.

INVITE: A mensagem contem a informação de ambos os utilizadores para iniciar a conexão entre os agentes.

BYE: mensagem enviada para iniciar o fim da sessão

ACK: mensagem que responde à mensagem INVITE e outras do tipo 2xx, 3xx, 4xx.

CANCEL – cancela um pedido pendente

REGISTER – cliente regista o seu endereço num servidor

OPTIONS- Servidor é questionado acerca das capacidades

As mensagens de respostas são:

1xx: respostas informativas como *Trying* e *Ringing* que indica o progresso da chamada.

2xx: respostas de sucesso e de aceitação como OK

3xx: mensagens de reencaminhamento do servidor SIP por diferentes nós para estabelecer a rota para a conexão.

4xx: Erro de cliente é usado pelo servidor ou pelo agente do utilizador

5xx: erro de servidor por não estar disponível.

6xx: erros globais quando o servidor sabe que não vai ter sucesso

Os pacotes de envio das mensagens no SIP são constituídas pelos seguintes dados:

Alert-info: providencia um toque diferente para a chamada

Call-id: id para identificação da chamada

Contact: URI da fonte

Cseq: um contador que aumenta cada vez que é usado pelo servidor SIP.

From: URI da fonte

Max-forwards: numero máximo de hops que pode ter até atingir o destino

Retry-after: diz quando o recurso está novamente disponível para voltar a tentar.

Subject: Indica o assunto da sessão

To: URI do recetor

Via: guarda o caminho do SIP.

As entidades/utilizadores do SIP são identificados por um URI que tem a forma de um endereço de "e-mail" como por exemplo, user@host.com.

Para haver proteção nas informações trocadas pelo protocolo SIP, é adicionado o TLS que permite, através do *TLS Handshake Protocol (*conjunto de passos a realizar para o estabelecimento do TLS), efetuar a troca das mensagens de forma segura. Para isso, primeiramente, é necessária a realização de troca de mensagens "Hello", para definir os algoritmos que serão usados nas comunicações, como o método de criptografia e de compressão, a versão do protocolo e o identificador da sessão. De seguida, é então necessário realizar a troca de um parâmetro de criptografia, onde será criada uma pré-chave de encriptação, de forma a encriptar os dados antes de se obter os certificados digitais, e também a troca dos certificados para se poderem autenticar a cada um deles. Após estes passos iniciais é gerada uma chave mestre da pré-chave, que é enviada como "ChangeCipherSpec" pelo cliente. Estes passos são apresentados na figura 2.

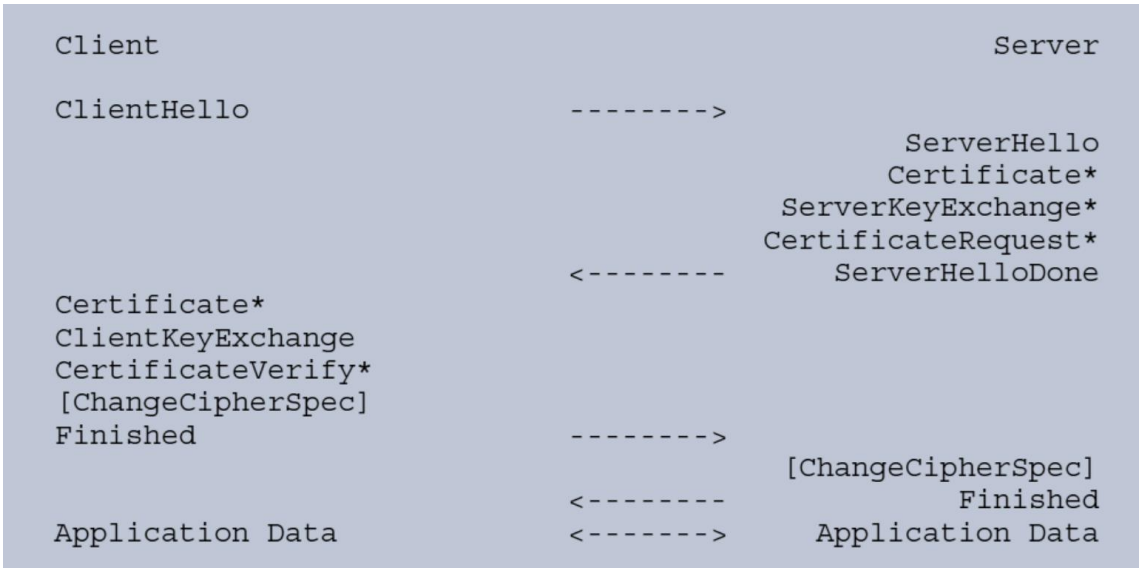

Figura 2 - Comunicações efetuadas pelo TLS antes do uso do protocolo SIP (in [RFC 5246])

2.2.2 H.323

Este protocolo aborda o mesmo problema que o protocolo SIP, ou seja fornece os mesmos tipos de funcionalidades, mas de formas muito diferentes com mecanismos distintos [Wang 2004]. O H.323 é recomendado pela ITU-T e o SIP é recomendado pela IETF.

H.323 não é um protocolo individual mas sim um conjunto de protocolos para o controlo de chamadas e sinalização e baseia-se em outros protocolos. Este protocolo define esses protocolos que vão providenciar as sessões de comunicação e de áudio e vídeo.

Está preparado para funcionar em qualquer rede quer seja de ponto-a-ponto, ou de um ou mais segmentos de rede. É bastante usado pelos produtores de aplicações VoIP, é usado também em várias aplicações na Internet como "GnuGK" e "NetMeeting" e também pelas empresas para serviços de áudio e vídeo sobre redes IP.

Funcionamento do protocolo H.323

Este protocolo visa estabelecer a configuração inicial e posteriormente dar apoio na troca de informações multimédia. Baseia-se no uso de outros protocolos para a troca das mensagens e informações necessárias. Desenvolve-se ao longo de uma série de passos. Inicialmente, através do uso do sistema RAS (Registo, Admissão e Estado), o H.323 realiza o registo do utilizador usando a mensagem RRQ para pedido do registo e a mensagem RCF para confirmação do registo por parte do servidor. De seguida, realiza o pedido de admissão, ou seja, pedido de informação da localização do utilizador que pretende contactar e recebe como resposta a mensagem ACF (Aceitação de ligação). Obtendo todos os dados para realizar o início da comunicação com o outro utilizador, usando o protocolo H.225.0 (Call Signaling Messages), comunica diretamente com o utilizador de destino. Mas para este poder realizar esta comunicação precisa de efetuar também o passo de admissão, em que pede a localização do

utilizador que iniciou a comunicação. Após a sessão estabelecida serão então avaliadas e negociadas as capacidades de cada terminal fazendo uso do protocolo H.245. Por fim os dois terminais podem iniciar as transferências multimédia com os canais RTP/RTCP. Na figura 3 podemos ver então as trocas de mensagens realizadas pelo H.323.

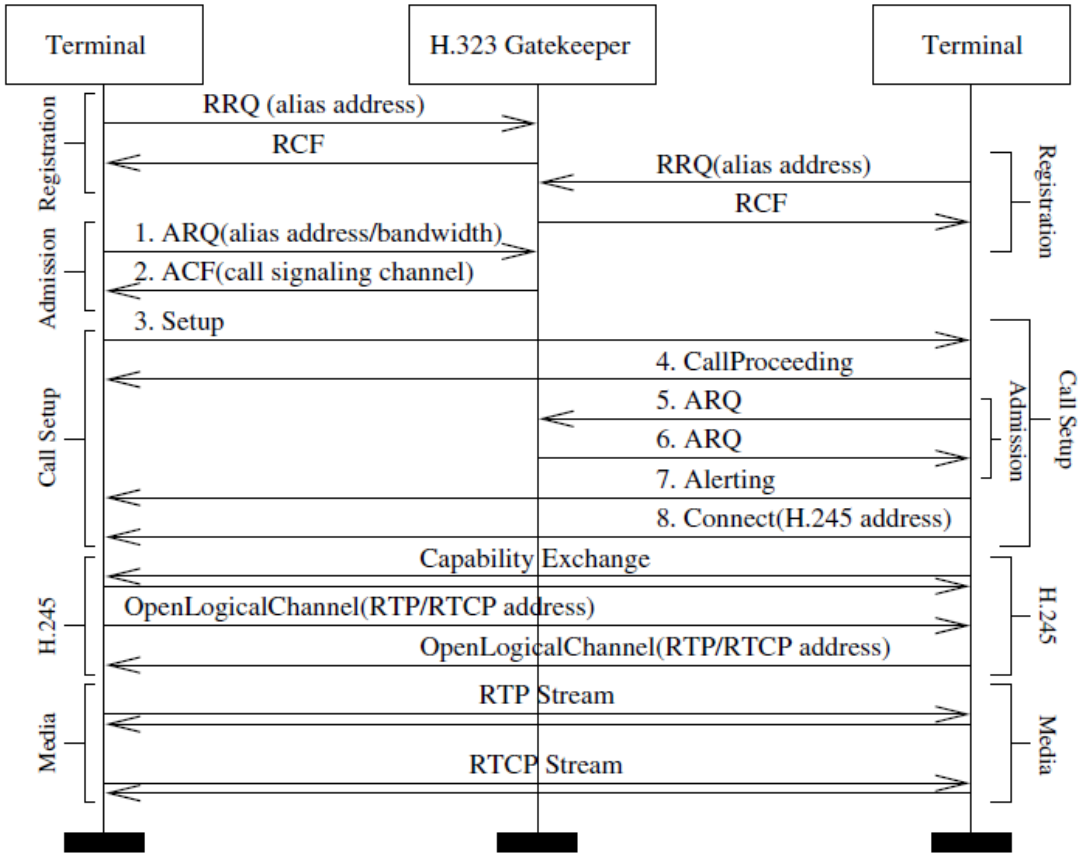

Figura 3 - Comunicações realizadas pelo protocolo H.323 para a realização de comunicações multimédia (in [Wang 2004])

2.2.3 Conclusões

O protocolo H.323, em comparação com o protocolo SIP, torna-se mais complexo, pois possui um conjunto de diferentes protocolos. O protocolo SIP em relação ao H.323 providencia uma complexidade menor, maior extensibilidade, o que significa que futuramente, permitirá ser mais explorado e desenvolvido com outros fins e uma melhor escalabilidade pois permite uma mais fácil inserção num meio diferente, sendo assim possível trabalhar com novos componentes. [Schulzrinne 1998].

Perante estes factos o protocolo SIP parece ser o mais adequado a utilizar, pois permitirá disponibilizar um serviço mais leve e menos complexo, apesar do H.323 estar melhor ao nível de interoperabilidade, ou seja, o sistema encontra-se mais desenvolvido para evitar a ocorrência de erros, e por já providenciar a fase de trocas multimédia ao contrário do SIP.

2.3 Protocolos de transferência multimédia

Neste ponto serão apresentados os protocolos que realizam tarefas de transferência multimédia, mostrando os passos que realiza para o seu sucesso assim como toda a constituição das mensagens. Para a transferência multimédia apenas o protocolo RTP é relevante, por exemplo, o H.323 também usa o RTP para a parte multimédia. O que se pode acrescentar a este protocolo são então métodos seguros para a troca de dados e o que faz o Secure RTP. Estes serão os protocolos abordados neste ponto.

2.3.1 Real-time Transport Protocol (RTP)

O RTP é usado intensivamente em comunicações e sistemas de entretenimento que envolvam transferências de dados multimédia, como telefones e videoconferência. É um padrão da IETF [RFC 3550] desenvolvido pelo "Audio-Video Transport Working Group" e tendo sido publicada a primeira versão em 1996, como "RFC 1889".

Este protocolo é entendido como uma rede de ponto-a-ponto, transmitindo dados em tempo real, quer seja em multicast ou unicast. O protocolo RTP consiste em duas partes: o RTP e o Real-time Transport Control Protocol (RTCP) que tem como função obter estatísticas da transferência dos pacotes RTP tais como quantidade de pacotes perdidos e tempos de transferência. O RTP é desenhado para transportar os dados em tempo real, enquanto o RTCP é usado para monitorizar a qualidade do serviço [Mohammed 2012].

As aplicações tipicamente correm RTP sobre UDP para fazer uso dos serviços de multiplexagem e *checksum*, podem também usar outros protocolos como o TCP. No RTP se for usado ao mesmo tempo o áudio e vídeo numa conferência, eles são transmitidos em separado.

Existe um conjunto de protocolos que se funcionam em conjunto com o RTP como o SIP, Jingle, H.225 e H.245 que são usados para realizar o início das sessões.

Funcionamento do protocolo RTP

Este protocolo providencia serviços de entrega ponto-a-ponto para dados em tempo–real num pacote de especificações que é constituído por um conjunto de pequenos pacotes como se pode visualizar na figura 4.

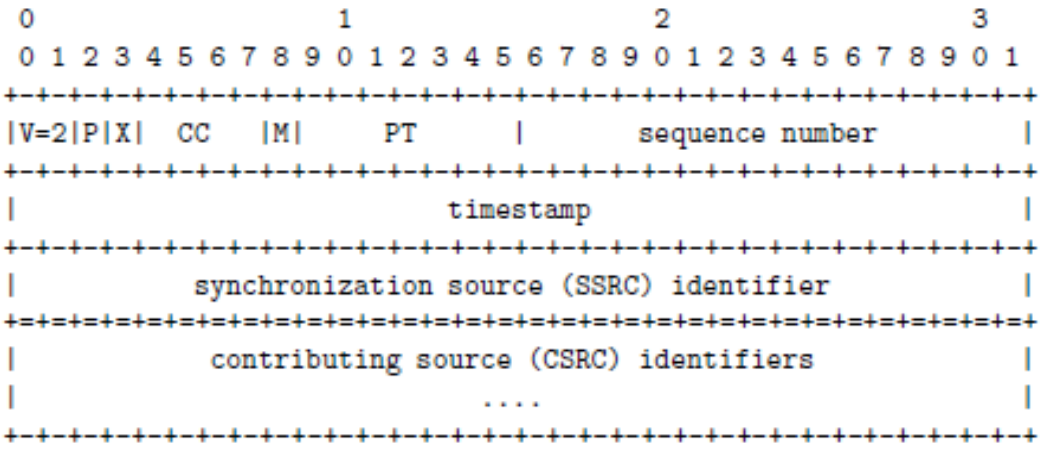

Figura 4 - Constituição do pacote de especificações do RTP (in [RFC 3550])

A explicação das partes constituintes deste pacote encontram-se no Anexo I. Após este pacote é então enviado o pacote com os dados de multimédia.

2.3.2 Real-time Streaming Protocol (RTSP)

O Real Time Streaming Protocol é um complemento ao protocolo RTP pois acrescenta-lhe a hipótese do utilizador iniciar e terminar a transferência de dados quando deseja. No fundo, o que este protocolo permite é dar ao utilizador algum controlo sobre a altura em que os dados são enviados pelo protocolo RTP após ter sido estabelecida a ligação entre dois utilizadores.

2.3.3 Secure Real-time Transport Protocol (SRTP)

O SRTP define da mesma forma o perfil do RTP mas providencia confidencialidade, integridade e proteção às mensagens trocadas pelo RTP. Foi também desenvolvido SRTCP, que é o pacote de RTCP para controlo estatístico do RTP mas com segurança. Da mesma forma que o RTP. O SRTP tem como pré-definida a encriptação baseada na cifra AES e HMAC-SHA-1 para a autenticação de mensagens [Alexander 2009]. O SRTP usa chaves simétricas, que devem ser negociadas durante o estabelecimento da sessão.

Funcionamento do protocolo SRTP

O SRTP baseia-se na estrutura do RTP acrescentando ao pacote de especificações do RTP a encriptação das partes mais importantes como se pode ver na figura 5.

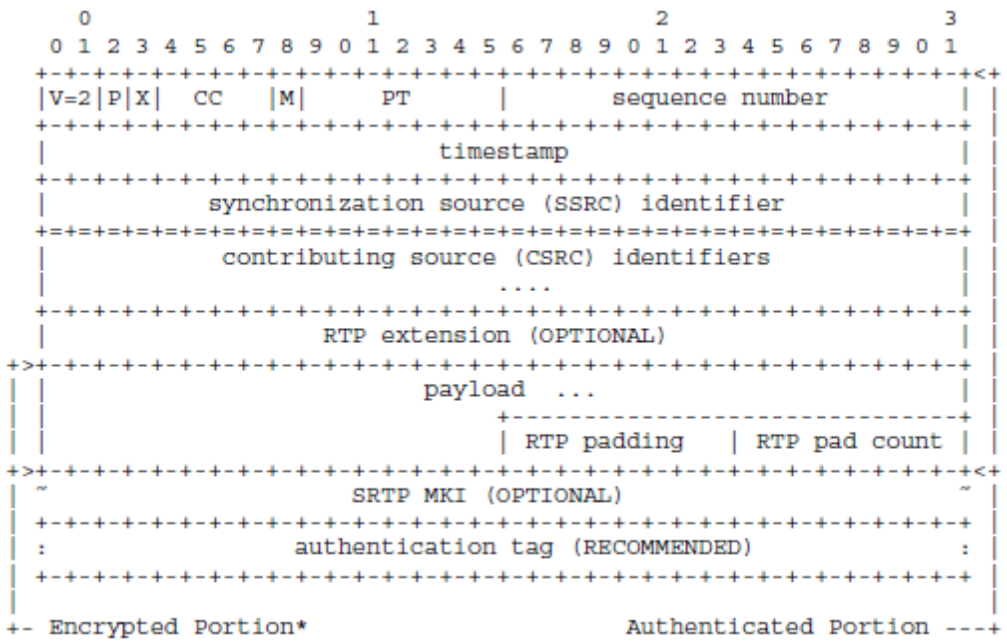

Figura 5 - Constituição do pacote de especificações do SRTP (in [RFC 3711])

A parte encriptada consiste na encriptação do RTP payload, campo onde se encontram os dados multimédia. Esta parte encriptada pode ser do mesmo tamanho do texto ou maior.

2.3.4 ZRTP

O ZRTP é um protocolo para comunicações multimédia entre dois pontos, que contem a fase de estabelecimento de sessões para chegar a acordo para a troca de chaves e os parâmetros para estabelecer sessões SRTP.

Este protocolo não é baseado em chaves públicas ou certificados digitais, mas sim nas chaves Diffie-Hellman, que são geradas em cada sessão. O método Diffie-Hellman é um algoritmo de criptografia específico para a troca de chaves baseado em logaritmos discretos. Estas chaves contribuirão para a geração da chave de sessão e parâmetros para as sessões de SRTP.

Apesar de inicialmente o ZRTP precisar de usar um protocolo de sinalização como o SIP, após isso a negociação de chaves passa a ser unicamente efetuada pelo ZRTP.

Funcionamento do protocolo ZRTP

Este protocolo engloba todos os passos desde o estabelecimento da sessão até à sessão multimédia, fazendo uso do SRTP que se encontra já inserido no ZRTP. As comunicações iniciam-se através do envio "ZRTP Hello" do utilizador que quer iniciar a comunicação; assim é possivel confirmar se o destino suporta o protocolo e que tipo de algoritmos eles têm em

comum. Como resposta ao "ZRTP Hello" apresenta "ZRTP HelloACK" que simplesmente informa das suas capacidades. Esta comunicação é feita em ambos os sentidos.

Após esta fase inicial será então iniciada a fase de troca de chaves que se inicia quando é enviada a mensagem "ZRTP Commit". Logo de seguida, é enviado o "ZRTP DH1" e o "ZRTP DH2" através dos quais são enviados os valores públicos para depois serem então calculadas as chaves SRTP. A parte que inicia a conversa tem de gerar a sua chave antes de enviar a mensagem "Commit" e o recetor gera a sua chave antes de enviar o "DH1". O funcionamento deste protocolo encontra-se então apresentado na Figura 6.

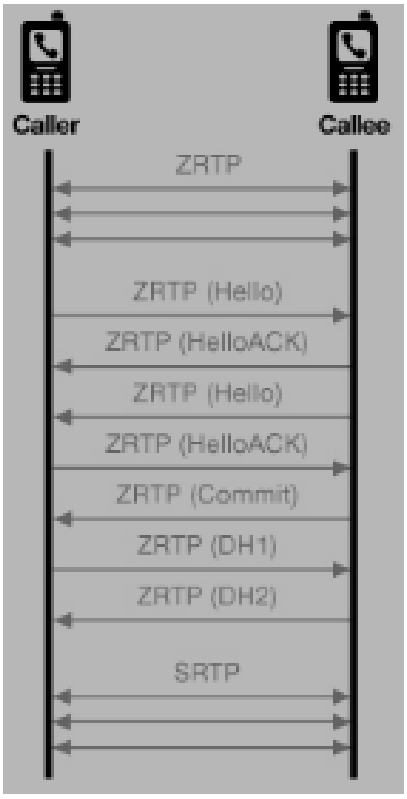

Figura 6 - Funcionamento do protocolo ZRTP para a realização de comunicações multimédia (in [RFC 6189])

2.3.5 Conclusões

Podemos ver que o SRTP não é mais que um acréscimo de informações ao RTP apenas para garantir a segurança das informações trocadas no RTP. Assim podemos concluir que para realizar troca de mensagens com segurança entre dois utilizadores é necessário usar o protocolo SRTP.

O ZRTP também tem, tal como o H.323 uma maior complexidade, mas não tão elevada, pois realiza uma série de processos para chegar a acordo quanto ao método de criptografia, dando ao SIP melhor fiabilidade.

2.4 Métodos e normas de criptografia

Neste ponto serão apresentados alguns métodos que ajudam as aplicações a tornarem-se mais seguras quer na altura da negociação quer na altura de encriptar os dados. Por um lado, dispomos de sistemas de criptografia de chave simétrica que permite encriptar um conjunto de dados, e, por outro, algoritmos que permitem melhorar a segurança quando se pretende obter uma dessas chaves simétricas.

2.4.1 Advanced Encryption Standard (AES)

O AES é uma especificação para a encriptação de dados estabelecida pelo Instituto Nacional de Padrões e Tecnologia (NIST) em 2001, sendo esta uma cifra de blocos que utiliza chaves secretas com tamanhos de 128, 192 ou 256 bits. Este sistema de encriptação simétrica é considerado o mais eficiente e robusto da atualidade pois permite proteger os dados e garantir a qualidade das comunicações multimédia. [Wang 2011].

2.4.2 Diffie-Hellman (DH)

O Diffie-Helman é um algoritmo que negoceia a troca de uma chave secreta comum entre os dois utilizadores que pretendem trocar dados. Este algoritmo permite a duas partes que não se conhecem gerar e partilhar uma chave através de um canal inseguro.

Funcionamento do protocolo DH

Este algoritmo baseia-se em cálculos de logaritmos discretos. Através da partilha de alguns valores que são obtidos por cada um dos utilizadores, de forma independente e com requisitos rigorosos para o correto funcionamento, é possível então obter uma chave simétrica sem nunca a ter de passar por um canal público.

Inicialmente os dois utilizadores entram em acordo em usar um mesmo valor de *p*, um número primo, e *g* uma raiz primitiva do módulo de *p*. Após chegarem a acordo destes valores, o primeiro utilizador que pode usar por exemplo a variável *a* escolhe para esta um inteiro secreto, ou seja, nunca vai divulgar este valor, permanece local. Com esta variável o primeiro utilizador realiza o seguinte cálculo: $A = g^a \bmod(p)$. Após a realização deste, envia então para o segundo utilizador o valor de A.

O segundo utilizador tem dois procedimentos a realizar, sendo o primeiro receber o valor obtido pelo primeiro utilizador, e o segundo proceder ao mesmo mecanismo que o primeiro, ou seja, a uma variável b, atribuir um inteiro secreto. Com este valor realiza então a conta $B = g^b \bmod(p)$ e envia o valor de B para o primeiro utilizador.

Após estes passos cada utilizador tem apenas de fazer o cálculo da chave secreta usando no caso do primeiro utilizador o valor do segundo o *B* e o seu valor secreto *a* realizando o

seguinte cálculo: chave = B^a mod(p). Para o segundo utilizador utiliza-se o mesmo procedimento, ou seja, o valor do primeiro utilizador *A* e o seu valor secreto *b* e realizar o cálculo: chave = A^b mod(p).

2.4.3 Conclusões

Os protocolos e algoritmos de segurança referidos anteriormente em toda a revisão bibliográfica não são comparáveis pois eles complementam-se uns aos outros. Sendo então o TLS uma forma de garantir inicialmente a credibilidade do utilizador e evitar o envio das informações desprotegidas. Por outro lado o Diffie-Hellman permite ao utilizador realizar a obtenção de uma chave comum em um canal aberto numa ligação ponto-a-ponto. Este pode complementar o TLS ou fornecer uma alternativa incompleta quando não é possível recorrer ao TLS.

2.5 Aplicações existentes que usem estes protocolos

Até ao momento não existem muitas aplicações que recorram à utilização destes protocolos, sendo a aplicação Jitsi [Jitsi 2014] a única que faz uso do SIP e do RTP, sendo o primeiro usado com TLS e o segundo usa o SRTP ou o ZRTP. Esta aplicação permite assim com o uso destes protocolos realizar chamadas entre dois utilizadores conhecidos que usem estes protocolos. Esta aplicação funciona para os sistemas operativos Windows, Linux e Mac OS X.

Outra aplicação que poderia ser usada como referência, não só pelo seu sucesso como também pela sua qualidade seria a aplicação Skype. No entanto, pelo facto de ser uma aplicação com protocolos proprietários não é possível determinar em quais se baseiam. Porém é possível saber que faz uso dos métodos de transporte UDP e TCP e do padrão de criptografia AES com chave de 256 bits.

2.6 Conclusões

Perante os protocolos existentes que possam realizar comunicações multimédia, é necessário efetuar dois passos. Primeiramente, é necessária a criação da sessão e o segundo a transferência dos dados multimédia.

Para o início de sessão existem os protocolos mais reconhecidos, o SIP e o H.323. Ambos desempenham as mesmas funções quando nos referimos aos protocolos de comunicação mas de formas diferentes, o SIP e o H.323 fazem uso do envio de mensagens de texto mas recorrendo a diferentes tipos de cabeçalhos nas mensagens, sendo no SIP cabeçalhos por ele criados enquanto no H.323 essas informações provêm de outros protocolos que já definem esses cabeçalhos das mensagens.

Já acerca dos protocolos para a transferência de multimédia só existe um que permite efetuar todas essas transferências o RTP, havendo depois possibilidade de acrescentar segurança

com o protocolo SRTP ou então, usando o ZRTP. A única diferença que pode ser apontada ao último protocolo é a falta da garantia da autenticidade dos utilizadores, sendo que nas comunicações multimédia é baseado no RTP.

O AES é o melhor método a ser utilizado pois fornece um serviço com segurança e rapidez e é considerado a melhor solução de confidencialidade nos sistemas de comunicações multimédia entre dois pontos. Uma vez que, o AES necessita de uma chave comum aos dois utilizadores, será possível recorrer ao algoritmo Diffie-Helman para que ambas as partes possam obter a respetiva chave sem a revelar em canais desprotegidos.

3 Projeto de desenvolvimento

3.1 Introdução

Este capítulo será focado na estruturação do processo de criação do projeto, englobando a metodologia de desenvolvimento e a metodologia de teste. Na metodologia de desenvolvimento será apresentada a estrutura do sistema a realizar assim como as mensagens que se virão a trocar. Será também abordado o tipo de ferramentas necessárias à conceção do produto, assim como as bibliotecas disponíveis. Na metodologia de teste será abordada a forma como o programa será analisado e testado na fase final para garantir o correto funcionamento da aplicação.

Todo o sistema encontra-se divido em duas partes, o Utilizador e o Servidor, demonstrando de seguida os requisitos necessários.

Requisitos Utilizador:

Tabela 1 - Requisitos funcionais (Utilizador)

Identificador	Descrição
FU1	O utilizador pode autenticar-se na aplicação
FU2	O utilizador pode inserir o nome do utilizador que pretende contactar
FU3	O utilizador pode iniciar a comunicação quando pretende
FU4	O utilizador pode iniciar e terminar as transferências multimédia quando desejar

Tabela 2 - Requisitos não-funcionais (Utilizador)

Identificador	Descrição
NFU1	Deverá existir uma aplicação para o dispositivo Raspberry-Pi
NFU2	A aplicação deve possuir uma interface gráfica
NFU3	Deverá ser segura em todos os pontos de comunicação garantindo a confidencialidade da informação

Requisitos Servidor:

Tabela 3 - Requisitos funcionais (Servidor)

Identificador	Descrição
FS1	No Servidor um Administrador pode autenticar-se com acesso a dados exclusivos
FS2	No Servidor o Administrador pode aceder aos utilizadores registados, aos dados destes e eliminá-los
FS3	No Servidor o Administrador pode aceder a uma listagem das mensagens trocadas

Tabela 4 - Requisitos não-funcionais (Servidor)

Identificador	Descrição
NFS1	Deverá existir uma plataforma Web no Servidor acessível através de um browser
NFS2	Deverá existir uma máquina virtual para alojar o Servidor
NFS3	Deverá encontrar-se sempre disponível o Servidor

3.2 Arquitetura e Metodologia de desenvolvimento

Um processo de comunicação físico inicia-se com o *login* do utilizador registado. Após concretizada a fase inicial será pedido que este insira o *username* da pessoa que pretende contactar e posteriormente iniciar a chamada fazendo uso das comunicações SIP. Através desta comunicação inicial será possível obter a localização do utilizador que se pretende contactar, e iniciar-se a troca de chave simétrica, recorrendo ao algoritmo Diffie-Hellman. Segue-se a transmissão de dados com recurso ao protocolo RTP com encriptação simétrica, ou seja, recorrendo ao protocolo SRTP, como referido posteriormente.

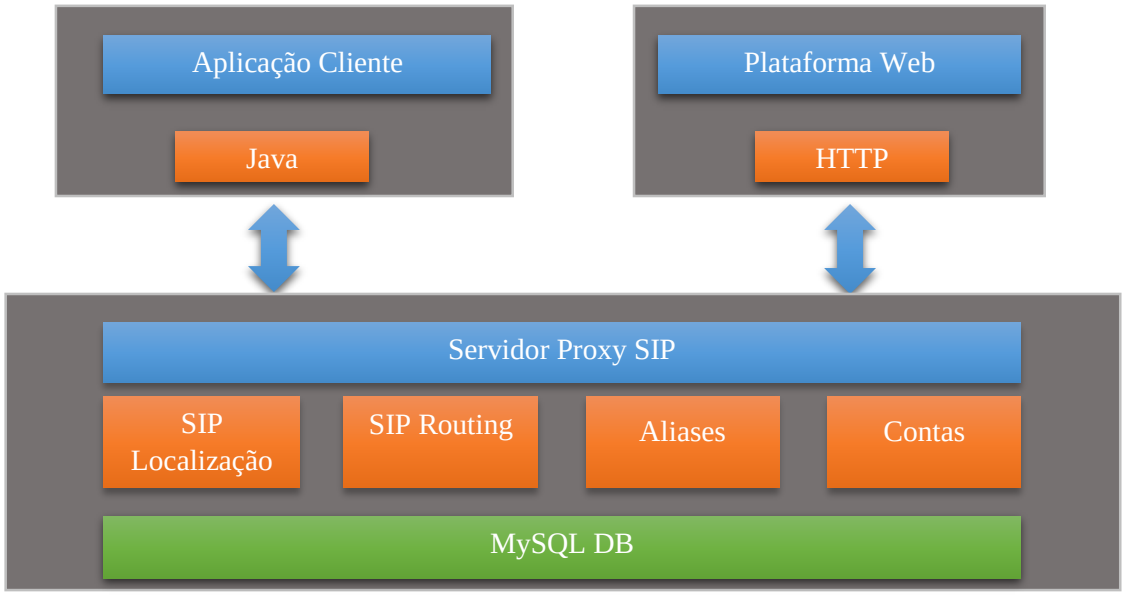

Figura 7 - Arquitetura de alto nível

A figura 7 representa a arquitetura de alto nível de toda a aplicação, sendo esta dividida em três partes. A aplicação cliente onde conterá as APIs para a implementação dos protocolos e todo o desenvolvimento envolvido para fornecer ao utilizador a hipótese de contactar outro utilizador. O servidor *proxy* SIP que conterá a informação de todos os utilizadores e realizará a interpretação das mensagens recebidas e redirecionará essas para o utilizador pretendido. Por fim a plataforma Web que dará ao Administrador da aplicação a hipótese de ter acesso aos dados armazenados pelo servidor como por exemplo os utilizadores e o tipo de mensagens

enviadas. Quer a aplicação cliente quer a plataforma web estarão interligados com o servidor proxy SIP.

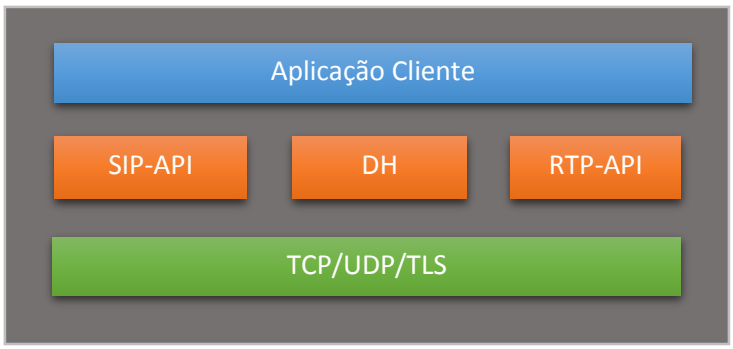

Figura 8 - Arquitetura de alto nível da aplicação cliente

Na figura 8 é então possível analisar mais detalhadamente a aplicação cliente onde se pode ver as APIs e algoritmos a serem usados, assim como a possibilidade de canais a usar na transferência dos dados quer de comunicação quer de multimédia. No canal de transferência é referido os três modos em conjunto mas estes não se encontram ao mesmo nível pois o TLS usa o TCP para a transferência dos dados. Nesta aplicação após a realização da troca das mensagens SIP segue-se a seguinte estrutura para a troca de mensagens da Figura 9 incluindo o algoritmo Diffie-Hellman e o protocolo RTP com o uso de criptografia de chave simétrica.

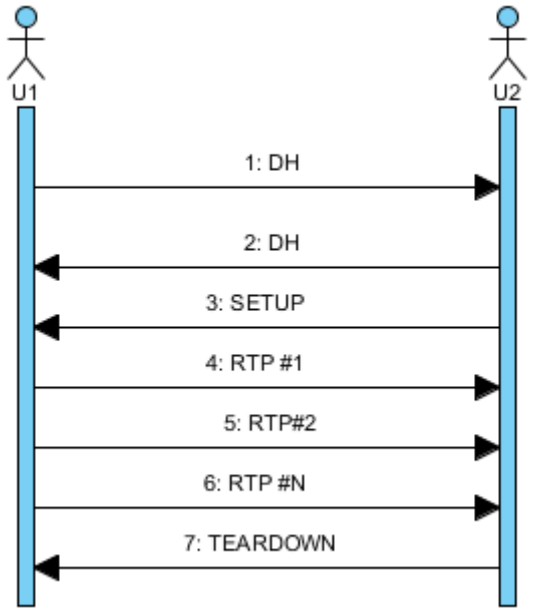

Figura 9 - Diagrama de sequência do processo após o SIP

3.2.1 Servidor Proxy SIP

Este servidor funcionará apenas numa fase inicial do contato, onde são trocadas as mensagens SIP, e mediará o utilizador que realiza o pedido e aquele que o vai receber, tal como se pode observar na Figura 10. Inicialmente este servidor terá como finalidade realizar o encaminhamento para o destinatário, pelo que é possível para um dos utilizadores realizar comunicações sem saber a correta localização do outro utilizador. Para isso basta apenas enviar as informações com referência ao destinatário e este servidor transmitirá as informações até ser estabelecido um método para a comunicação multimédia, pois aqui será realizada uma ligação de ponto a ponto. Por outro lado, estará também a verificar se o utilizador que está a realizar a comunicação pode ou não proceder à ligação com o utilizador de destino.

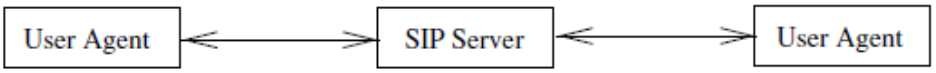

Figura 10 - Rede para o funcionamento do protocolo SIP (in [Wang 2004])

O servidor também terá a seu cargo interpretar o recebido para confirmar a identidade do utilizador e a quem se destinam as mensagens recebidas. Estará responsável por receber os pedidos de registo (REGISTER), cujo processo de funcionamento está demonstrado na Figura 11, o conteúdo desta mensagem permitirá ao utilizador atualizar a sua localização enviando o novo endereço para o servidor.

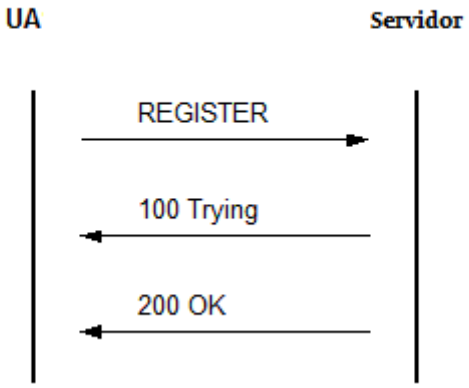

Figura 11 - Registo de utilizador no servidor

3.2.2 Protocolo de comunicação SIP com UDP ou TLS

Para o estabelecimento da comunicação SIP serão usadas as funcionalidades do servidor referido em cima para a troca das mensagens necessárias até à iniciação das comunicações multimédia, a referir posteriormente. Nesta fase da comunicação será então usado uma sequência de mensagens representadas na Figura 12.

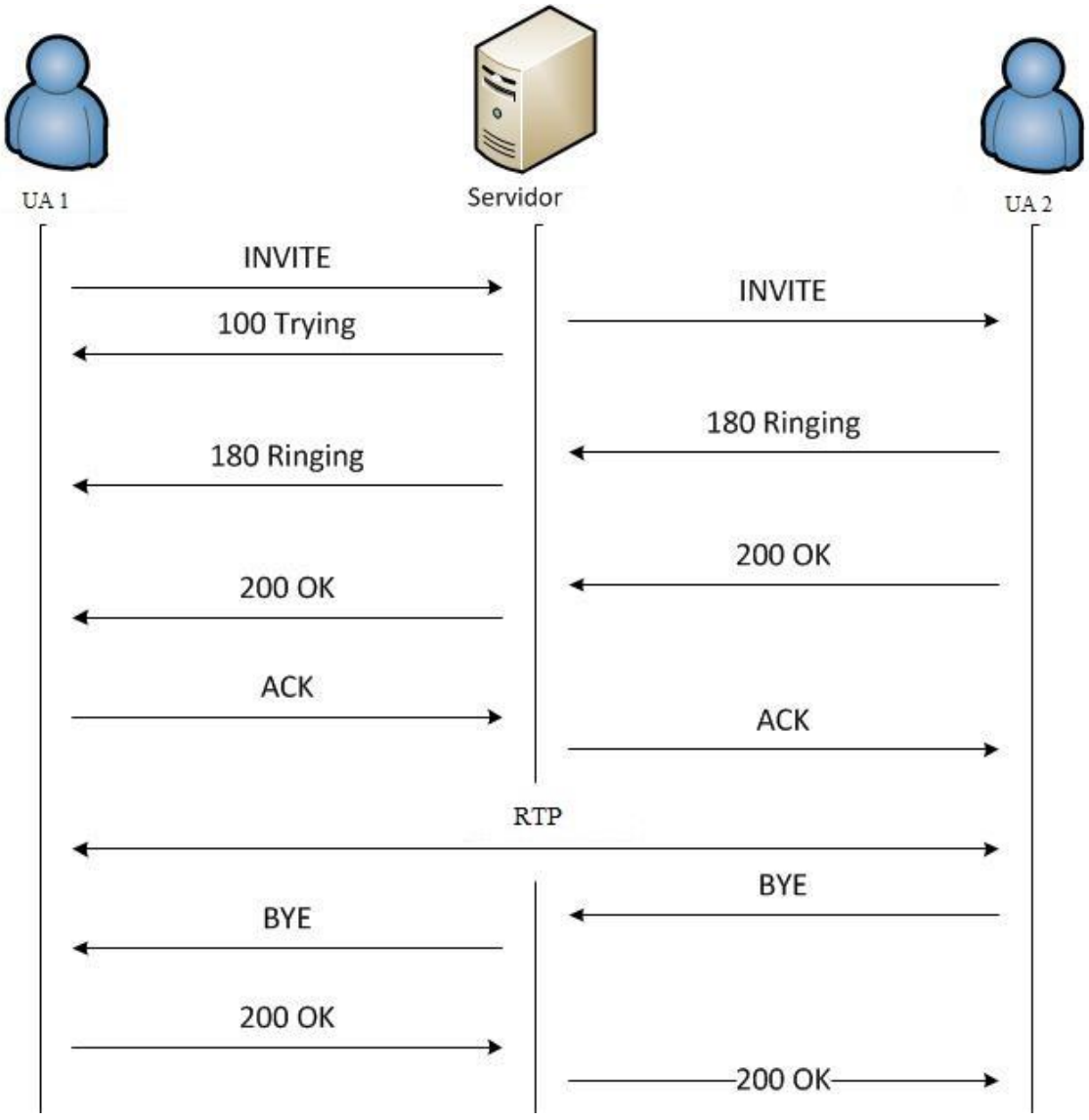

Figura 12 - Troca de mensagens realizada pelo protocolo SIP – UA, agente de utilizador

Cada uma destas mensagens representa um pedido, discriminando os dois tipos de utilizadores envolvidos no processo aquele que vai iniciar o contato (utilizador 1) e o destinatário (utilizador 2) sendo estes:

- ***INVITE*** – O utilizador 1 recorre a esta mensagem com a finalidade de contactar pela primeira vez o utilizador 2 de forma a obter a aprovação deste para poder iniciar a conexão;

- ***RINGING*** – É uma mensagem enviada automaticamente pelo utilizador 2 representado a confirmação da receção da chamada. Serve também para informar o utilizador 1 para aguardar pela aceitação da chamada pelo utilizador 2;
- ***OK*** – Confirmação do utilizador 2 que recebeu a chamada e que aceita esta e quer iniciar as comunicações;
- ***ACK*** – Mensagem enviada pelo utilizador 1 que recebeu o *OK* do utilizador 2 para iniciar a chamada. Nesta mensagem seguem então os dados para se passar para uma comunicação de ponto-a-ponto;
- ***RTP*** – Localiza-se aqui a construção do RTP a ser explicado posteriormente;
- ***BYE*** – Enviado pelo utilizador que pretender terminar a chamada;
- ***OK*** – Confirma com esta mensagem o pedido de *BYE* por parte do outro utilizador.

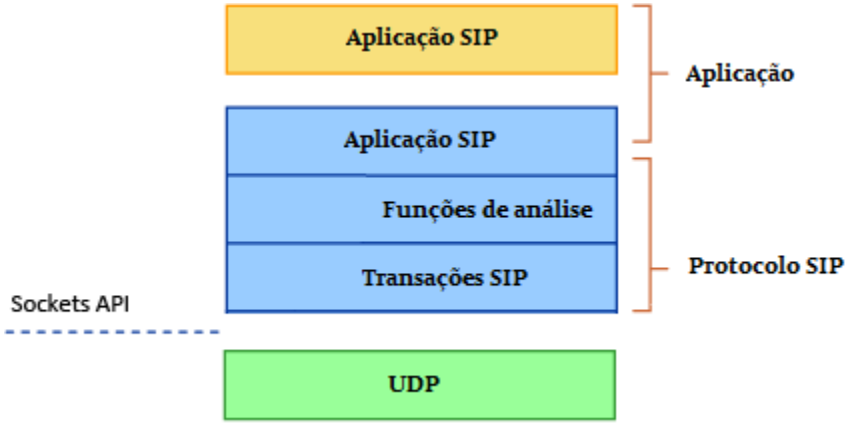

Figura 13 - Arquitetura da aplicação SIP

Aplicação SIP – Esta é a parte onde será implementada a sequência de mensagens a ser enviadas e o tratamento dos dados recebidos.

Funções de análise – Nesta parte é onde se construirá as mensagens com os dados indicados na Aplicação SIP, ou seja, quando o utilizador na aplicação SIP chama a função para criação da mensagem por exemplo com o tipo "REGISTER".

Transações SIP – Aqui serão então enviadas as mensagens construídas para o servidor e estará também pronto a receber mensagens por parte do servidor que fará a ligação aos sockets.

UDP – Transporte de pacotes usando o protocolo UDP realizado por meio de um socket.

Foi adotado o protocolo UDP em vez do TCP, pois embora cada um deles seja capaz de concretizar o mesmo tipo de envio, o UDP torna-se mais rápido pois não efetua um controlo de erros nem de pacotes perdidos, permitindo assim uma maior eficiência da aplicação. Este controlo de pacotes perdidos e erros é realizado pelo próprio pacote SIP sendo que contém campos que permitem verificar estes mesmos.

Para tornar o protocolo SIP mais seguro será usado o protocolo TLS em vez do UDP, uma vez que o primeiro permitirá garantir a autenticidade do utilizador e do servidor, através do uso de certificados digitais, e posteriormente definir uma chave simétrica para encriptar as mensagens RTP numa fase posterior.

3.2.3 Secure Real-time Transport Protocol e Real-time Streaming Transport Protocol (SRTP e RSTP)

Após o estabelecimento de sessão SIP com TLS todos os dados serão transferidos ponto-a-ponto como ilustrado na figura 9 seguindo a implementação base do RTP.

Figura 14 - Construção da rede RTP

Para a implementação deste protocolo será necessário tratar em primeiro lugar da construção do pacote RTP, seguidamente encriptar o conteúdo útil a enviar de um utilizador para o outro, conferindo-lhe proteção e confidencialidade. Através do uso do protocolo SRTP, que se baseia na adição da criptografia de chave simétrica ao RTP, podemos garantir que aqueles dados não serão inteligíveis a quem tentar interferir na conversa. Este protocolo encripta então o campo que contém os dados, *payload*.

Para o controlo dos dados enviados, o pacote RTP abarca um conjunto de variáveis que permitirão manter o outro utilizador informado. A variável *Sequence Number* fornece informação acerca da ordem de leitura dos pacotes, através de uma sequência numérica dos conjuntos de pacotes. Por sua vez, a variável *Payload Type* é responsável por discriminar o tipo de multimédia a ser enviado no momento.

Estas são as variáveis mais importantes enviadas juntamente com o pacote de dados que permitirão a correta interpretação destes. Ao mesmo tempo é possível usar o protocolo RSTP para permitir ao utilizador controlar as comunicações multimédia, iniciando e terminando a transferência dos dados quando muito bem desejar.

3.2.4 Diffie-Hellman (DH)

O algoritmo DH, Diffie-Hellman pode ser uma forma de combinar uma chave criptográfica simétrica, que por sua vez será usada para proteger a ligação entre os dois utilizadores que se encontram a realizar a conversação. Assim o Diffie-Hellman oferece uma solução simples de como se pode realizar uma troca de chaves em um canal aberto. O esquema que se segue demonstra o funcionamento do processo em questão.

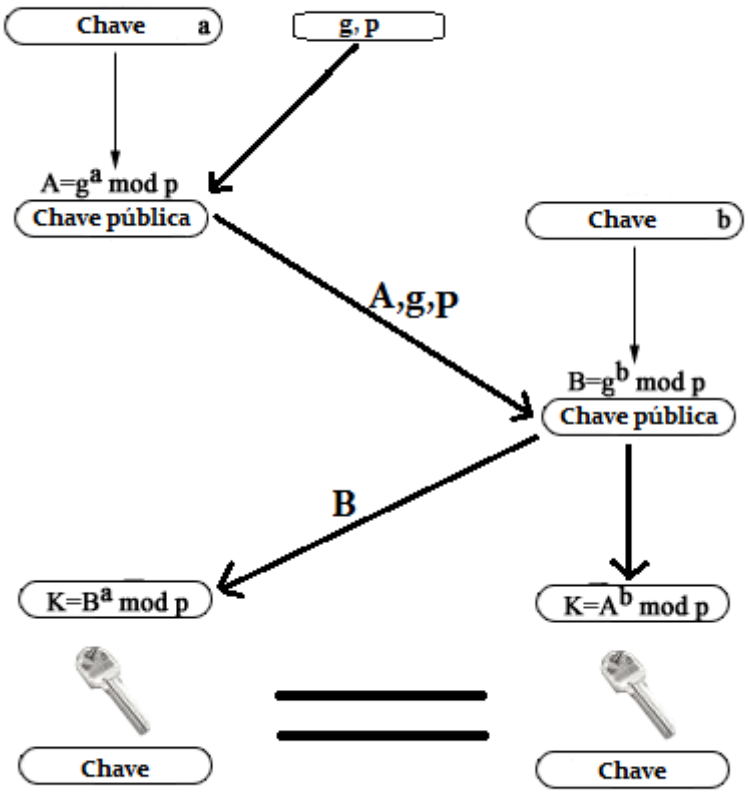

Figura 15 - Processo detalhado do Diffie-Hellman

O DH não é uma solução alternativa ao TLS mas por vezes um complemento quando nos referimos à obtenção de uma chave de encriptação comum. Como normalmente a autenticidade das partes já é garantida com recurso aos certificados digitais no TLS, não é correntemente usado em simultâneo. Sendo que com estas duas formas podemos definir de forma segura uma chave de encriptação. Quando não é possível realizar a troca de chave recorrendo ao TLS. Podemos então recorrer ao DH para obter essa chave. Usando o DH não é possível garantir o mesmo nível de segurança como no TLS mas permite melhorá-la bastante tornando a aplicação segura.

3.2.5 Interface para Utilizador e Administrador

Quer para o utilizador quer para o administrador pretende-se que haja uma interface simples para a realização das comunicações e também uma forma de se poder aceder aos dados existentes no servidor responsável por armazenar todas as informações dos utilizadores registados e das comunicações em curso.

Para o utilizador seria uma interface simples que lhe permita realizar as operações mais simples como login, indicar utilizador a contactar, realizar chamada e desligar chamada.

Para o administrador será necessária uma interface que permita aceder aos utilizadores registados e a todos os dados relativos às comunicações realizadas.

3.3 Metodologia de teste

No final do desenvolvimento da aplicação, será preparado um ambiente de teste onde se pretende ter um conjunto de utilizadores a realizar algumas chamadas simples com conteúdos multimédia, num ambiente local, dentro da mesma rede interna, usando um servidor em uma máquina virtual.

A segurança de todo o ambiente de comunicação é garantida pelo uso de protocolos seguros que usam métodos de encriptação que segundo os criadores destes protocolos são os melhores métodos de encriptação. [applicacaomultimedia 2014]

Para podermos verificar se os dados foram bem transmitidos será realizada uma tentativa de os ler à chegada em duas situações, uma com a chave de cifra e outra com uma chave diferente.

Sendo que o desenvolvimento de todo o projeto vai estar centrado nos dispositivos de baixo desempenho, neste caso o Raspberry-Pi, vai ser necessário efetuar uma série de testes à rapidez de processamento nos diferentes pontos por onde passam as comunicações. Podemos assim ir verificando até que ponto é possível utilizar estes dispositivos com capacidades de processamento limitadas em tarefas de encriptação e processamento de vídeo.

4 Implementação de plataforma

4.1 Introdução

Ao longo deste capítulo será apresentado toda a estrutura da aplicação desenvolvida e explicadas as decisões tomadas. Será também feita, uma apresentação detalhada das comunicações entre os utilizadores e entre os utilizadores e servidor.

4.2 SIP Server (Kamailio) e Plataforma Web (Siremis)

Neste ponto será demonstrada a estrutura do servidor e a sua plataforma Web e as ferramentas usadas para atingir o pretendido.

Para a implementação do servidor será utilizado o "SIP proxy/server", Kamailio [Kamailio 2014], desenvolvido em "open source", que disponibiliza de forma quase imediata todos os recursos necessários para a realização de servidor das comunicações SIP.

De forma a ser possível ter acesso a um maior número de informações acerca dos utilizadores registados e da sua localização, foi utilizada outra ferramenta que é compatível com o servidor Kamailio: a plataforma Web Siremis [SIREMIS 2014], que possibilita a um Administrador obter acesso aos detalhes de algumas informações importantes das comunicações SIP que estão a ser realizadas.

A instalação deste servidor ficará alojada em uma máquina virtual, em princípio em ambiente Linux baseado em Debian.

O Kamailio foi configurado acedendo aos seus ficheiros de configuração ativando os módulos necessários para funcionar como servidor SIP. Para isso foi inicialmente ativado o módulo que permite criar a respetiva base de dados, fazendo esta uso do MySQL. Para a ativação do uso do MySQL foi então adicionado às configurações o uso do mesmo, como demonstrado nos anexos. Após a adição destes dados pode-se então proceder à respetiva instalação que já incluirá este módulo e os necessários de origem. Posteriormente é então criada a base de dados que é gerada automaticamente com um comando. O Kamailio após a instalação encontra-se em escuta numa porta predefinida, normalmente a 5060, através dos protocolos UDP e TCP. Em baixo encontra-se uma listagem das mensagens devolvidas pelo servidor, Kamailio, quando este é iniciado.

Ilustração 1 - Iniciação do servidor proxy SIP

```
* Stopping Kamailio SIP server: kamailio [ OK ]
* Starting Kamailio SIP server: kamailio
loading modules under /usr/local/lib64/kamailio/modules/
Listening on
     udp: 127.0.0.1:5060
     udp: 192.168.2.142:5060
     tcp: 127.0.0.1:5060
     tcp: 192.168.2.142:5060
Aliases:
     tcp: sip-server.vst:5060
     tcp: localhost:5060
     udp: sip-server.vst:5060
     udp: localhost:5060
                                          [ OK ]
```

A constituição da base de dados do Kamailio baseia-se principalmente na aglomeração dos seus utilizadores e respetivos dados, da localização de cada utilizador quando inicia sessão, e no controlo de pedidos como "INVITE".

Este servidor foi instalado em uma máquina virtual localizada na rede local da empresa com o "alias", "sip-server", com o sistema operativo Ubuntu 13.04 (GNU/Linux 3.8.0-35-generic x86_64).

Quanto ao Siremis, é uma plataforma Web bastante completa e que possibilita várias utilizações. Sendo o mais importante, a possibilidade de usufruir do acesso aos utilizadores registados e às informações referidas anteriormente. Estas informações são apresentadas tal como é possível verificar na imagem abaixo disposta.

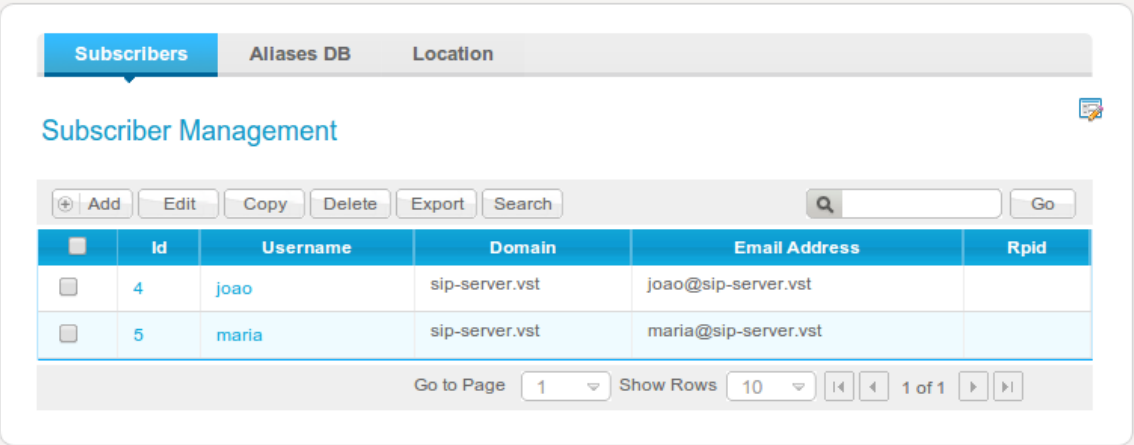

Figura 16 - Plataforma Web – Interface para o Administrador para visualização dos utilizadores que se encontram registados

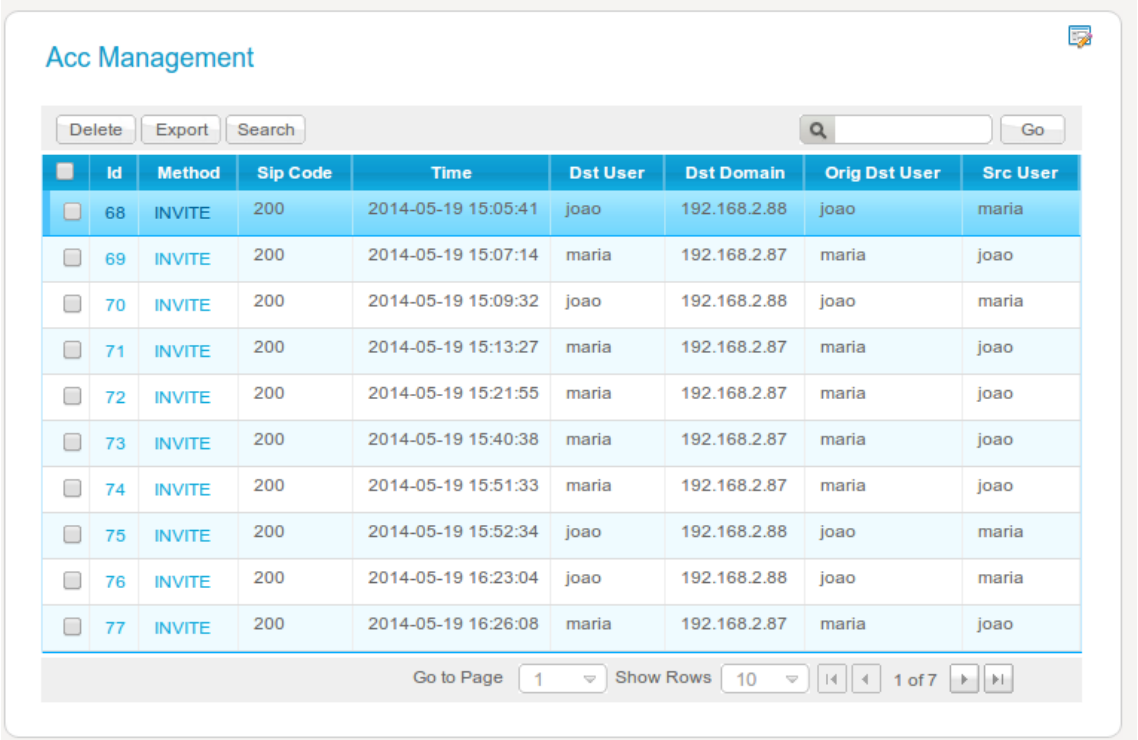

Figura 17 - Plataforma Web - Registo mensagens SIP – Acc é o mesmo que Accounting

Aqui podemos visualizar as ligações pedidas por cada um dos utilizadores, com dados de quem está a tentar contactar quem e a sua localização na rede.

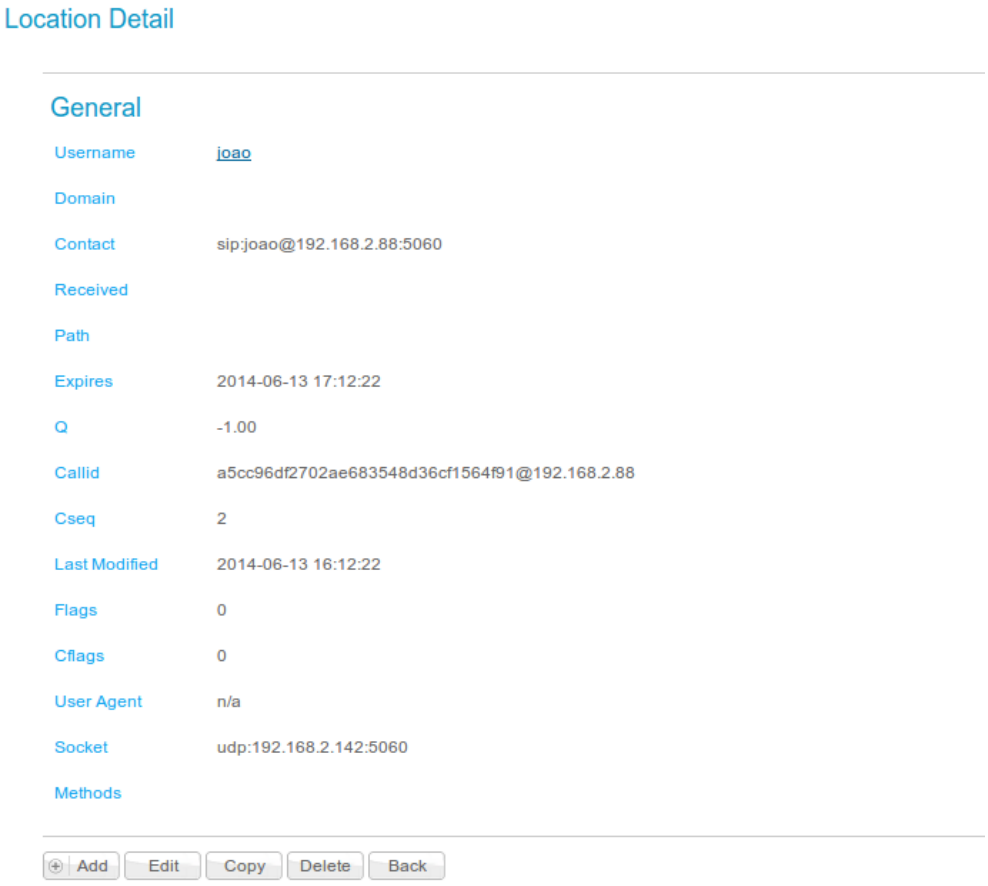

Figura 18 - Plataforma Web - Utilizador e Localização

Com o uso desta opção na plataforma Web podemos obter a localização do utilizador que se registou no campo "Contact" e quando realizou esse pedido no campo "Last Modified". Sendo neste caso que recorreu a mensagem SIP do tipo "REGISTER" para poder efetuar o seu registo no sistema de forma a ser reconhecido.

4.3 Ferramentas do lado do Utilizador

Para a implementação do SIP e do RTP dispomos de algumas bibliotecas como é o caso do PJSIP [PJSIP 2014] que é implementada em C, existe também uma versão em Java desta biblioteca que é a *pjsip-jni* mas que se encontra muito incompleta pois é um "*wrapper*" e ainda não foi totalmente adaptada para Java. Estas últimas bibliotecas encontram-se com bastante documentação mas com poucos exemplos práticos e tutoriais confusos, sem identificação da versão englobando também uma instalação bastante complexa.

Outra biblioteca disponível é a *jsip* [JSIP 2013] mas que apenas foca a comunicação SIP, pelo que é mais conhecida como Jain-SIP. Dispõe de uma documentação mais reduzida mas com bastantes exemplos práticos. Esta biblioteca é uma implementação de "baixo nível" das especificações da Java API para o uso de sinalização no formato SIP.

A biblioteca Jain-SIP permite desde já uma implementação completa do protocolo SIP [RFC 3261], que pode ser incorporada em outras aplicações, e ser usada para realizar

comunicações com servidores HTTP e também como um utilizador SIP. Assim, torna-se útil numa implementação de um utilizador SIP necessário para iniciar a realização das comunicações multimédia. Usando esta biblioteca o projeto foi concretizado mais facilmente.

4.3.1 Session Initiation Protocol (SIP) no Utilizador

Toda a aplicação é iniciada na classe Crypto onde o utilizador efetua o seu login e é registado no servidor a sua localização. É nesta classe que se encontra os dados do utilizador, o nome do servidor e a porta que vai ser usada para a criação do socket com o servidor

O protocolo usado para a transferência das mensagens por parte do SIP é o UDP sem recurso ao TLS pois não foi possível implementá-lo. Uma vez que não seria possível implementar, no espaço de tempo disponível, os protocolos necessários (SIP e o SRTP) bem como o servidor, foi necessário recorrer a bibliotecas open-source. Estas bibliotecas são capazes de implementar estes mesmos protocolos no entanto, quando foi necessário usar o canal TLS verificou-se a ocorrência de incompatibilidades entre o servidor e a biblioteca Java usada, Jsip. Então, apesar de estes dois suportarem TLS, quando aplicados na prática, não funcionaram corretamente aquando o inicio das comunicações.

Após o utilizador ter inserido os seus dados para a realização do login é construída uma mensagem SIP com a identificação REGISTER, pois é esta mensagem que vai permitir à aplicação obter do servidor a aprovação dos dados deste utilizador, retornando diferentes mensagens conforme os dados se encontrarem corretos ou errados.

As mensagens SIP normalmente são constituídas usando a base da estrutura SIP definida no protocolo sendo esta:

- Tipo de registo, neste caso é do tipo REGISTER + SIP-URI
- Call-ID header
- CSeq header (número que representa a sequência de mensagens)
- From header
- To header [sendo neste caso o mesmo que o From header sendo que só estamos a efetuar o registo]
- Max-Forwards header
- Via header
- Contact header

Usando um exemplo prático para demonstrar a constituição da mensagem SIP que o utilizador construiu para enviar para o servidor para se registar e validar, recorrendo ao campo "Authorization":

Ilustração 2 - Mensagem "REGISTER" com autenticação

```
REGISTER sip:joao@192.168.2.142:5060;maddr=192.168.2.142
SIP/2.0.
Call-ID: b4bdab0d64b53d47647eab7eca87b84d@192.168.2.88.
CSeq: 2 REGISTER.
From: "joao" <sip:joao@sip-server.vst>;tag=12345.
To: "joao" <sip:joao@sip-server.vst>.
Via: SIP/2.0/UDP 192.168.2.88:5060;branch=z9hG4bK-323432-
27852f007f4b8b264c4fcb309fa7f02f.
Max-Forwards: 70.
Contact: "joao" <sip:joao@192.168.2.88:5060>.
Content-Type: application/sdp.
Call-Info: <http://www.antd.nist.gov>.
Authorization: Digest username="joao",realm="sip-
server.vst",nonce="U4SYIVOElvWJ0KuwD03hmI6moHxJzgLI",uri="sip:jo
ao@192.168.2.142:5060;maddr=192.168.2.142",response="20c4a69eebf
b9e2760710f6ed7a9539d".
Content-Length: 247
```

O servidor responde e tem ao seu dispor dois tipos de mensagens, consoante a validade dos dados introduzidos. Assim essas respostas virão com a informação do sucedido na primeira linha da mensagem. Se os dados introduzidos pelo utilizador estiverem corretos será enviado, por parte do servidor, uma resposta com o formato "`SIP/2.0 200 OK`". Em caso de se encontrarem incorretos os dados fornecidos, login ou password, será enviado "`SIP/2.0 401 Unauthorized`".

A partir desta altura, o programa desenvolvido espera pela ordem do utilizador para iniciar a ligação com o segundo utilizador. Para tal, o primeiro necessita de introduzir o *username* do utilizador a contactar e iniciar a ligação. Aqui, iniciar-se-ão as trocas das mensagens SIP a fim de ser estabelecida uma comunicação direta entre estes dois e para isso será seguida a base de mensagens trocadas no protocolo SIP.

Na configuração dos parâmetros necessários é criado um *Listener* que estará à escuta na porta configurada inicialmente. Esse *Listener* estará dividido em duas partes, sendo a primeira a função que espera por um pedido e a segunda a que aguarda por uma resposta:

```
public void processRequest(RequestEvent requestEvent) {…}
```

A função supracitada representa a que aguarda um pedido, ou seja, o utilizador que está a ser contactado.

```
public void processResponse(ResponseEvent event) {…}
```

Nesta segunda função, o utilizador que inicia a chamada aguarda pelas respostas do utilizador que pretende contactar.

Cada uma destas funções trata de criar a mensagem correspondente ao que recebe. Abaixo encontra-se uma representação do conteúdo das mensagens trocadas, de acordo com a respetiva ordem pedido-resposta.

Tabela 5 - Sequência das mensagens SIP por parte do utilizador que foi contactado

Recebe (pedido)	**->**	**Envia (resposta)**
INVITE	->	TRYING e após aceitar chamada envia OK
ACK (após a receção deste, são iniciadas as comunicações multimédia)		
BYE	->	OK

Por outro lado quando ocorre a receção de uma resposta, este vai avançar no processo de mensagens SIP, ocorrendo na seguinte ordem cronológica:

Tabela 6 - Sequência das mensagens SIP por parte do utilizador que iniciou o contacto

Recebe	**->**	**Envia**
		INVITE (para iniciar a chamada)
INVITE	->	ACK (após o envio deste ACK, são iniciadas as comunicações multimédia)
BYE	->	OK

Estas mensagens são um pouco diferentes da anterior, pois contém o utilizador que inicia a comunicação e o utilizador que este pretende contactar, sendo esta mensagem ainda um pedido.

Ilustração 3 - Mensagem "INVITE"

```
INVITE sip:maria@192.168.2.142:5060;maddr=192.168.2.142 SIP/2.0.
Call-ID: ab18abd328a8b17f979aea6f856e4495@192.168.2.88.
CSeq: 3 INVITE.
From: "joao" <sip:joao@sip-server.vst>;tag=12345.
To: "maria" <sip:maria@sip-server.vst>.
Via: SIP/2.0/UDP 192.168.2.88:5060;branch=z9hG4bK-323432-
d3936333acbd11ad2161f96e5ddfd2bf.
Max-Forwards: 70.
Contact: "joao" <sip:joao@192.168.2.88:5060>.
Content-Type: application/sdp.
Call-Info: <http://www.antd.nist.gov>.
Proxy-Authorization: Digest username="joao",realm="sip-
server.vst",nonce="U4SYJVOElvnTqw6KMa3REA6yQbHRAub6",uri="sip:ma
ria@192.168.2.142:5060;maddr=192.168.2.142",response="5f9d14c923
38151208b77c8163df8024".
Content-Length: 247.
```

Após a receção desta mensagem por parte do utilizador de destino, este irá criar uma resposta que normalmente exige uma menor complexidade, sendo apenas essencial definir o tipo de mensagem e enviar como resposta ao recebido.

Ilustração 4 - Mensagem "Trying" em resposta ao "INVITE"

```
SIP/2.0 100 Trying.
To: "maria" <sip:maria@sip-server.vst>.
Via: SIP/2.0/UDP
192.168.2.142;branch=z9hG4bK953a.ce3e0143.0,SIP/2.0/UDP
192.168.2.88:5060;branch=z9hG4bK-323432-
d3936333acbd11ad2161f96e5ddfd2bf.
CSeq: 3 INVITE.
Call-ID: ab18abd328a8b17f979aea6f856e4495@192.168.2.88.
From: "joao" <sip:joao@sip-server.vst>;tag=12345.
Contact: "maria" <sip:192.168.2.87:5060>.
Content-Length: 0
```

Durante as trocas das mensagens supracitadas, quando se realiza a troca da mensagem inicial, "INVITE", é retirado daí o endereço do utilizador com quem se vai proceder às trocas multimédia. Este endereço permitirá depois iniciar as comunicações para a realização da troca de chaves recorrendo ao Diffie-Hellman e para o estabelecimento do canal de transferência multimédia.

4.3.2 Diffie-Hellman

Foi necessário neste projeto recorrer ao DH em vez do TLS. Pois devido à impossibilidade de implementar os protocolos SIP e RTP neste projeto foi necessário recorrer a bibliotecas já existentes que não funcionaram corretamente em conjunto com o servidor. Apesar de não estar implementada a aplicação com TLS, cada utilizador inicialmente necessita de realizar o seu *login*. A sua *password* é enviada em público mas recorre-se a um método que transforma a password em caracteres indecifráveis para quem os visualiza, sendo este o MD5.

Na implementação deste algoritmo, DH, é possível usar as bibliotecas de segurança que o próprio Java disponibiliza, através de uma série de funções úteis.

Seguidamente será necessário criar uma divisão entre quem iniciará a troca de dados e quem os receberá, ou seja, no fundo podemos considerar que temos um servidor cliente e um cliente pois o Diffie-Hellman é uma comunicação de ponto a ponto. Para cada um destes será necessário gerar um conjunto de parâmetros que serão necessários para obter a chave pública para isso usam-se as três funções abaixo discriminadas. Nestes parâmetros, o principal a ser definido será o tamanho da chave comum gerada, esta foi definida com o valor mínimo, de 512 bits, pois este valor permitirá um melhor desempenho no Raspberry-Pi sem comprometer a segurança.

```
AlgorithmParameterGenerator paramGen =
AlgorithmParameterGenerator.getInstance ("DH");

paramGen.init(sizedh);

AlgorithmParameters params =
paramGen.generateParameters();
```

Por fim são então gerados todos os valores de entrada que o Diffie-Hellman necessita para depois realizar os cálculos, sendo estes, o seu valor secreto e os restantes dois valores que serão partilhados com o outro utilizador. Portanto, antes da realização da troca de chaves é

obtido, através do *KeyPair*, o valor secreto e os valores públicos sendo estes últimos os que vão ser enviados para o segundo utilizador:

```
KeyPairGenerator aliceKpairGen =
KeyPairGenerator.getInstance("DH");

KeyPair aliceKpair = aliceKpairGen.generateKeyPair();
```

Encontra-se aqui um exemplo dos valores públicos gerados pelo algoritmo.

```
SunJCE Diffie-Hellman Public Key:
p:
    c2478f88 1de9d199 8ef37752 338d4e79 dd96f5a3 5e5af117 2289fed5 4a13f75b
    cac75dc7 72fe080c a61fb2df 28d7bf67 94857ec6 c267999a a11fbc27 11b46d51
g:
    20c736c5 302cc736 b5b51dcc 11281e18 5d6e9531 4e21e44b d69f4807 cbc7088c
    3c76a8ee 8785efe4 98106f14 d6cfc802 87ff2d89 67f55604 f5c1f0b8 e628c38f
```

Para proceder a este envio é então necessário recorrer ao uso de sockets, ponto de comunicação entre duas aplicações em uma rede. Assim, do lado do servidor cliente foi então criado um *ServerSocket* para esperar pela ligação do cliente a este e poder transferir a sua *public key* e os seus valores públicos gerados.

Tal como no primeiro utilizador (o servidor cliente), o cliente segue então os mesmos passos na iniciação do algoritmo Diffie-Hellman e na criação do seu valor secreto. No fim da geração deste cria o *socket* para receber os valores enviados pelo servidor.

Neste primeiro envio de dados por parte do servidor cliente encontra-se então a chave pública e os valores públicos a usar para a geração da chave final. Assim o cliente faz uso destes dados para criar a sua própria chave pública. Pertencendo esta ao cliente, é necessário então enviar para o servidor recorrendo ao *socket* criado anteriormente.

Após cada um ter a chave pública um do outro pode então proceder ao cálculo da chave comum sem a ter de partilhar em canal público fazendo apenas uso dos logaritmos discretos.

```
SecretKey bobDesKey = bobKeyAgree.generateSecret("AES");
```

Por fim é então necessário transformar esta chave obtida no algoritmo Diffie-Hellman em uma chave suportada pelo AES, sendo para isso usada a função referenciada em cima, esta ao gerar a chave simétrica vem sempre por omissão com um tamanho de 256 bits.

4.3.3 Secure Real Time-Transport Protocol

Para a implementação deste protocolo recorreu-se ao software disponibilizado em [applicacaomultimedia 2014].

Nesta parte final da aplicação serão então transmitidos os dados multimédia entre os dois utilizadores onde, tal como no processo de obtenção da chave partilhada com recurso ao Diffie-Hellman, existe também uma divisão neste caso emissor e recetor. Neste caso, o emissor

será aquele que representa o que envia os dados e o recetor o que vai aguardar pela receção destes.

Não foi desenvolvida a possibilidade de transmitir áudio e vídeo em simultâneo, mas foi atingido o objetivo de transferir informação multimédia entre dois pontos com a possibilidade de transferência de vídeo. É possível assim visualizar do lado do cliente um vídeo em *stream*, em tempo real.

Para iniciar as comunicações multimédia foi necessário criar do lado do emissor um *ServerSocket*, pois todos os dados multimédia serão enviados através de *sockets*. Após a criação deste é então iniciado o emissor concretamente onde este irá iniciar outro *socket*, para poder comunicar o que o recetor pretende fazer, pois com o uso do protocolo RSTP o recetor pode iniciar as comunicações multimédia ou terminar estas quando pretender. Neste emissor é então aguardada inicialmente a ligação de um recetor, sinalizada com uma mensagem "SETUP". Após obtida esta mensagem é então adicionado à lista de recetores que se encontram conectados ao emissor para assim ser possível saber para quem se vai ter de enviar os pacotes RTP. Concluído então este processo, é iniciada a construção e divisão dos pacotes de dados e posteriormente o envio destes.

O envio dos pacotes RTP é processado num só local através do uso de um *Listener*, que aguarda pelo timer do emissor para poder enviar o pacote seguinte, enquanto o vídeo não chegar ao fim este *Listener* criará um pacote RTP que enviará para os recetores que se encontram conectados ao emissor, como se pode visualizar no excerto do código a baixo. Em caso de o vídeo que está a ser enviado em *stream* chegar ao fim será reiniciado.

```
RTPpacket rtp_packet = new RTPpacket(MJPEG_TYPE, currentImageNb,
currentImageNb*FRAME_PERIOD, buf,
image_length,Server.clientList.get(0).getKey());

Client.getDatagramSocket().send(new DatagramPacket(packet_bits,
packet_length,aClient.getClientAddress(),aClient.getRTPClientPor
t()));
```

Na construção do pacote RTP é criado um *array* de bytes onde serão adicionados todos os campos necessários e outro pacote que conterá os dados multimédia.

No primeiro, é reservado o byte 0 para a adição da *Version, Padding, Extension e CC*. O byte 1 é então para o *Marker e Payload Type* e o byte 2 e 3 para o *Sequence Number*. Os próximos 4 bytes são então preenchidos com o *Time Stamp* estando agora o *array* preenchido até à sétima posição. Por fim, fica a faltar então o Ssrc que serão mais 4 bytes no *array*, ficando assim preenchido até á decima primeira posição, tendo assim no total 12 bytes.

O segundo *array* é criado com os dados multimédia e, depois encriptado recorrendo ao algoritmo AES (disponível nas bibliotecas disponibilizadas pelo Java). Tal encriptação é feita com recurso às funções indicadas.

```
SecretKeySpec spec = new SecretKeySpec(key, "AES");
Cypher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, spec);
byte[] bytes = cipher.doFinal(text);
```

Inicialmente é então definido, na *função SecretKeySpec()* que se pretende usar, a encriptação AES com a chave "key", tendo esta sido obtida quando se recorreu ao Diffie-Hellman. Logo a seguir é criado um *Cypher* que será este que está responsável pela encriptação dos dados. Inicialmente é dito que se pretende uma *Cypher* do tipo AES e em seguida que vamos proceder à encriptação, recorrendo à chave referida anteriormente. Por fim é então realizada a encriptação do texto, representado aqui por "text", sendo estes os dados multimédia que queremos enviar e que, após esta encriptação, ficarão modificados e ilegíveis para quem não possuir esta "key".

Após estarem construídos os dois *arrays* que representam o pacote RTP com recurso ao AES para encriptação, são então concatenados estes dois *arrays* para por fim serem enviados para o recetor ligado a este emissor.

Pelo seu lado, o recetor terá acesso a uma interface gráfica simples, que lhe permita escolher as opções do RSTP, *Setup* e *Teardown*, para inicializar e finalizar o *stream* do vídeo, respetivamente.

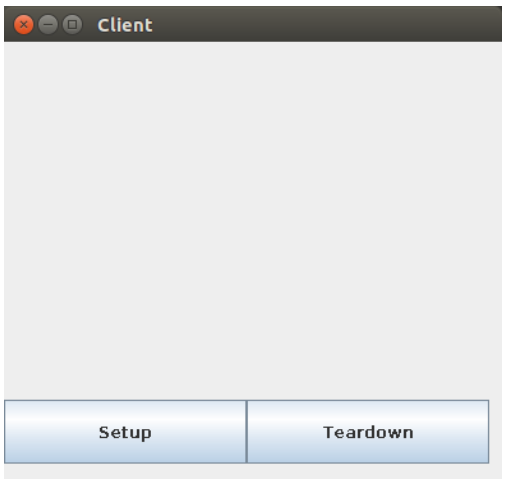

Figura 19 - Interface cliente antes do início do stream do vídeo

É criado, também para o recetor, um *Listener* que estará à espera dos dados que serão enviados por parte do emissor. O *Listener* é apenas iniciado após o recetor clicar no botão *Setup*, pois só aí é que o recetor pretende iniciar a receção de dados multimédia.

Para a realização das trocas das mensagens referidas anteriormente é então criado um *socket*, para poder informar desta forma o emissor de quando iniciar a transmissão ou finalizar.

Se o recetor clicar no botão *Setup* é iniciado o *Listener*. Este estará à escuta no *socket*, onde serão transferidos os pacotes RTP. Cada vez que recebe um, é analisado o pacote e separado conforme os valores recebidos, contendo também a parte dos dados que chegam encriptados. Por isso é então necessário recorrer à desencriptação dessa parte, com a chave obtida com o algoritmo Diffie-Hellman, usando o algoritmo AES e para tal será necessário utilizar as mesmas funções (referidas acima). No entanto, será preciso alterar o que se pretende realizar como demonstrado no excerto de código abaixo, sendo a alteração de "`Cipher.ENCRYPT_MODE`" para "`Cipher.DECRYPT_MODE`".

```
SecretKeySpec spec = new SecretKeySpec(key, "AES");
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.DECRYPT_MODE, spec);
byte[] bytes = cipher.doFinal(text);
```

Por fim é então reproduzido para o cliente os dados multimédia obtidos.

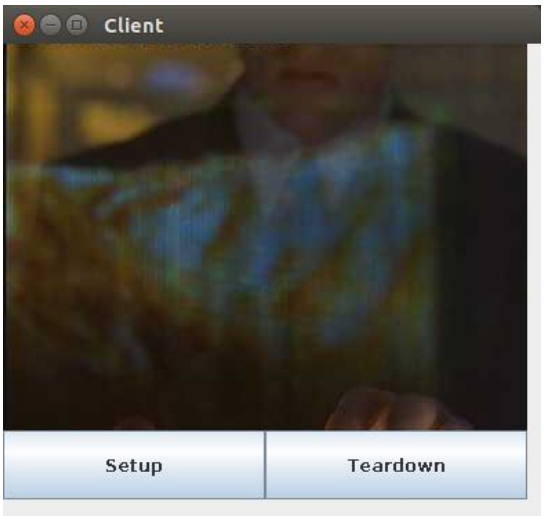

Figura 20 - Interface cliente após início do stream do vídeo

4.4 Testes realizados

Foi realizado um conjunto de testes desde o estabelecimento de uma comunicação de *stream* de dados multimédia até à medição de tempos de processamento na fase da criptografia, pois o dispositivo a usar, Raspberry-Pi, apresenta uma capacidade de processamento limitado.

O primeiro teste a ser concretizado foi a execução de um *stream* de vídeo entre dois dispositivos. Deste modo foi possível verificar a rapidez e qualidade da ligação. Foi possível retirar deste teste que a fluidez e qualidade da imagem é bastante razoável e todos os processos de comunicação realizados ao longo do protocolo SIP, apesar de não darem respostas imediatas, tinham um atraso bastante aceitável, demorando cerca de 5 segundos para efetuar o seu registo. Nas restantes mensagens trocadas com o servidor, são apenas precisos milissegundos para completar o processo a cada mensagem. Por outro lado, a realização da troca de chave recorrendo ao método de Diffie-Hellman, já demora alguns segundos, sendo este acontecimento causado pela existência de recursos limitados, tais como o seu processador e memória RAM, encontrando-se estes dados posteriormente. Por essa razão, e para confirmar a viabilidade da opção, foram realizados alguns testes mais exaustivos.

Inicialmente foi analisado o desempenho do processador, chegando à conclusão que este estava em constante funcionamento, sendo a sua utilização de 100% aquando da execução do algoritmo Diffie-Hellman. Posteriormente foi então realizada uma contagem dos segundos que este demorava a realizar os respetivos cálculos para o algoritmo, podendo estes ser analisados nos gráficos a baixo.

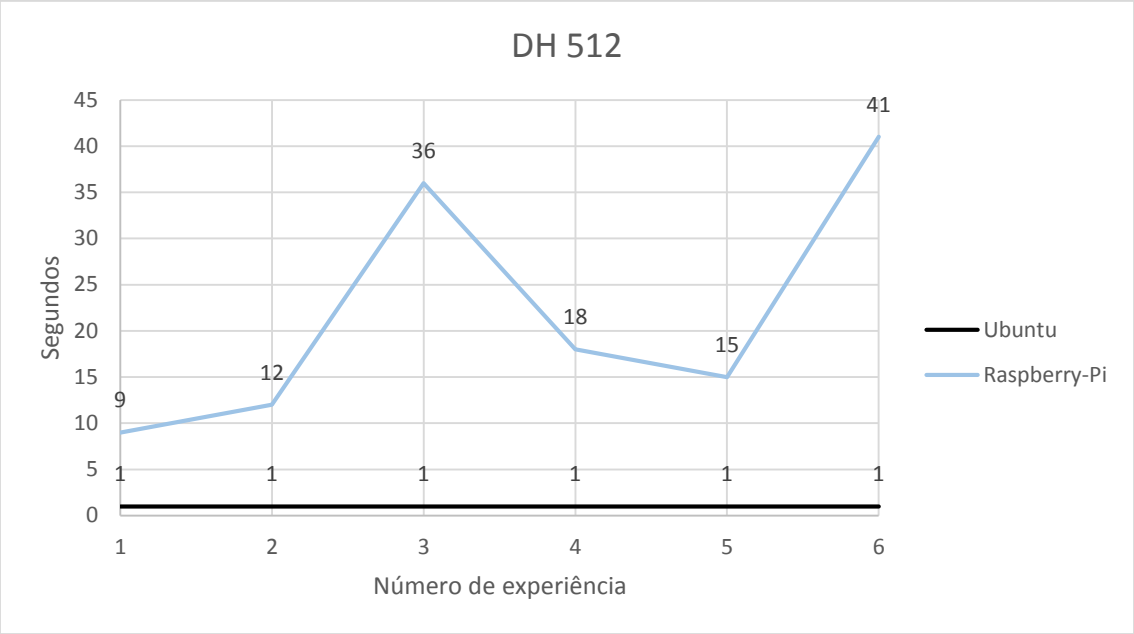

Figura 21 - Tempos obtidos na execução do algoritmo Diffie-Hellman para obtenção de chave de 512 bits

Neste primeiro gráfico podemos observar que foram realizadas seis experiencias onde foram obtidos para o Raspberry-Pi e para o Ubuntu, os dois dispositivos utilizados para efeitos de teste os tempos de execução para o algoritmo Diffie-Hellman na obtenção de uma chave de 512 bits. Pode-se ver que para o Ubuntu o tempo é constante e quase instantâneo pois possui boas características. Enquanto para o Raspberry-Pi os tempos já são mais demorados e inconstantes pois como este dispositivo tem um processamento mais limitado este dependendo de como se encontra a sua disponibilidade para processamento estes tempos variam. Podendo por isso encontrar por vezes alguma discrepância de 9 segundos para 36 segundos, por exemplo.

Para verificar se pode vir a ser viável em um Raspberry-Pi aumentar os tamanhos das chaves de encriptação recorrendo ao Diffie-Hellman realizou-se uma alteração na chave comum obtida por este algoritmo passando assim de 512 bits para 1024 bits para se poder obter tempos de processamento de forma a ser possível saber a viabilidade de nestes dispositivos um dia utilizar chaves de maiores dimensões,

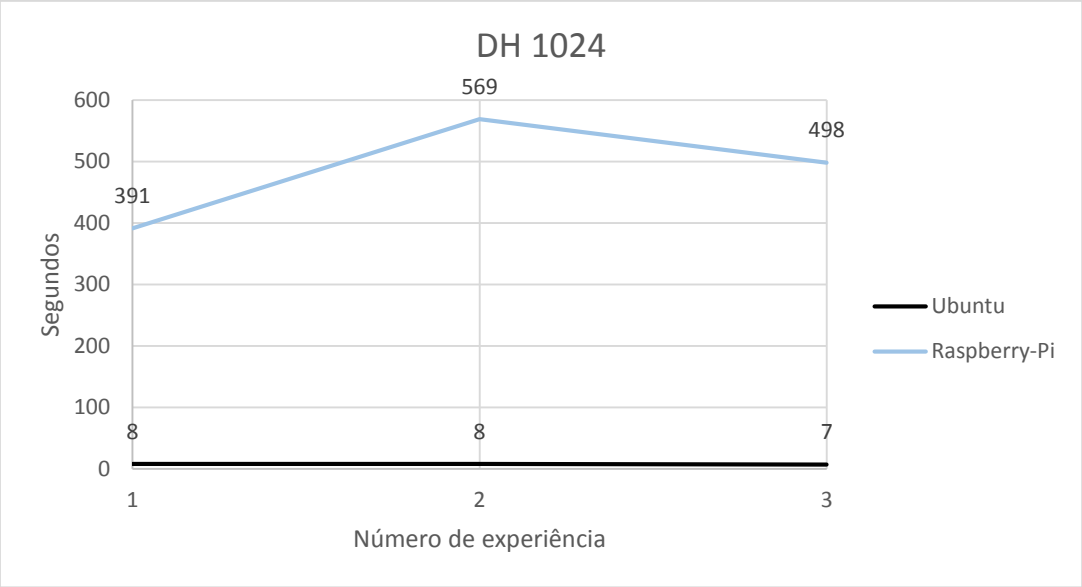

Figura 22 - Tempos obtidos na execução do algoritmo Diffie-Hellman para obtenção de chave de 1024 bits

Neste último gráfico foi então realizado para os mesmos dois dispositivos uma análise ao tempo de execução mas agora obtendo no final uma chave de 1024 bits. É possível então visualizar que os tempos obtidos aumentaram bastante, apesar de no Ubuntu ainda serem aceitáveis no Raspberry-Pi são tempos inaceitáveis pois um utilizador não vai aguardar cerca de 5 minutos para poder realizar a conceção com o outro utilizador.

Dispositivos de testes

Raspberry-Pi:

Processador: 700 MHz ARM1176JZF-S core (ARM11 family, ARMv6 instruction set)

Memória: 512MB RAM

Sistema operativo: Raspbian GNU/Linux 7 (wheezy)

Ubuntu:

Processador: Intel Core Duo P7550 2.26GHz

Memória: 2GB RAM

Sistema Operativo: Ubuntu 14.04 (Trusty Tahr) LTS

Podemos então concluir que o dispositivo Raspberry-Pi não será viável quando no futuro se tentar implementar protocolos mais seguros que exijam maiores capacidades de processamento. Como se pode comprovar pelos dados obtidos o tempo de processamento quando se aumenta o tamanho da chave que se quer obter é exponencial.

5 Conclusão e Trabalho Futuro

5.1 Conclusão

Pode-se concluir que no fim deste projeto foram atingidos quase todos os objetivos previamente estabelecidos, resultando daqui uma aplicação de comunicação multimédia operando de forma segura que se encontra funcional e testada.

O primeiro objetivo a ser atingido relaciona-se com a utilização dos protocolos SIP e RTP, para a criação da aplicação multimédia. Foi possível confirmar que esta se encontrava funcional a partir do momento em que foi possível transferir elementos multimédia entre dois utilizadores.

No uso do primeiro protocolo, foi implementado o protocolo SIP usando o protocolo UDP para o transporte realizado por sockets. Garantiu-se a autenticidade do utilizador através da obrigatoriedade da realização do login sempre que quer iniciar as comunicações SIP.

Para o problema da troca de chave, pois esta não poderia ser trocada através de um canal não seguro, já que estaria sujeito a que alguém intersectasse a mensagem e pudesse ler os dados enviados, recorreu-se ao método de Diffie-Hellman que permite chegar a uma chave comum entre dois utilizadores sem ser necessário passar a chave final em um canal público.

Em relação ao RTP, foi cumprido o objetivo de garantir que a transferência dos dados multimédia se realiza com os dados encriptados entre os dois utilizadores e o utilizador recetor os recebe e os visualiza na sua forma original.

Os testes realizados permitem ver a capacidade que o Raspberry-Pi tem para suportar diferentes tipos de encriptação e de leitura e reprodução multimédia. Foi possível concluir que, quando se aumenta o tamanho da geração da chave de 512 para 1024 bits, como será necessário realizar mais cálculos, a capacidade limitada de processamento deste dispositivo torna inviável o uso dessas chaves, se futuramente forem necessárias tais chaves. Por outro lado a leitura e reprodução de vídeo é bastante fluída para a capacidade de processamento que se tem disponível.

5.2 Trabalho Futuro

Devido ao facto deste projeto ter sido desenvolvido de raiz, existem bastantes melhorias a efetuar. O primeiro passo será implementar o que não foi possível nesta primeira fase experimental: a utilização do TLS e evolução da implementação do RTP, para ser possível suportar então áudio e vídeo; melhoria da interface do utilizador de forma a ficar mais agradável e mais intuitiva.

Outras melhorias e funcionalidades a concretizar são a integração de uma webcam na aplicação para ser possível realizar chamadas de vídeo e áudio entre os utilizadores e permitir a realização de troca de mensagens entre os dois utilizadores.

6 Referências Bibliográficas

[Alexender 2009] Alexander, A. L., A. L. Wijesinha and R. Karne. "An Evaluation of Secure Real-Time Transport Protocol (Srtp) Performance for Voip." In *Network and System Security, 2009. NSS '09. Third International Conference on*, 95-101, 2009.

[Alsmairat 2009] Alsmairat, I., R. Shankaran, M. Orgun and E. Dutkiewicz. "Securing Session Initiation Protocol in Voice over Ip Domain." In *Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on*, 78-83, 2009.

[applicacaomultimedia 2014] tdurand/applicacaomultimedia-ufc · GitHub, Junho de 2014. https://github.com/tdurand/applicacaomultimedia-ufc

[Bang 2012] Bang, J. and D. Kim. "Efficient Rtsp-Based Multiple Buffering and Packet Transmission Methods for Delivering Oma Poc Box Service." *Computer Networks* 56, no. 15 (2012): 3468-3478.

[Bansal 2013] Bansal, A., P. Kulkarni and A. R. Pais. "Effectiveness of Sip Messages on Sip Server." In *Information & Communication Technologies (ICT), 2013 IEEE Conference on*, 616-621, 2013.

[Bhargavan 2013] Bhargavan, K., C. Fournet, M. Kohlweiss, A. Pironti and P. Strub. "Implementing Tls with Verified Cryptographic Security." In *Security and Privacy (SP), 2013 IEEE Symposium on*, 445-459, 2013.

[Bresciani 2007] Bresciani, Riccardo. *The Zrtp Protocol Security Considerations*. Citeseer, 2007.

[Burget 2012] Burget, Radim, Dan Komosny and Kathiravelu Ganeshan. "Topology Aware Feedback Transmission for Real-Time Control Protocol." *Journal of Network and Computer Applications* 35, no. 2 (2012): 723-730.

[Cavusoglu 2005] Cavusoglu, Bulent. "Real-Time Content-Based Video Communications over Heterogeneous Networks." Ph.D., University of Illinois at Chicago, 2005.

[Chen 2010] Chen, Whai-En, Ya-Lin Huang and Yi-Bing Lin. "An Effective Ipv4–Ipv6 Translation Mechanism for Sip Applications in Next Generation Networks." *International Journal of Communication Systems* 23, no. 8 (2010): 919-928.

[H.323 2014] H.323 : Packet-based multimedia communications systems, Junho de 2014. https://www.itu.int/rec/T-REC-H.323/en

[Jitsi 2014] Jitsi, Fevereiro de 2014. https://jitsi.org/

[JSIP 2013] JSIP: JAVA API for SIP Signaling — Project Kenai, Abril de 2013. https://jsip.java.net/

[Kamailio 2014] Kamailio SIP Server, Junho de 2014. http://www.kamailio.org/w/

[Karopoulos 2013] Karopoulos, Georgios, Paolo Mori and Fabio Martinelli. "Usage Control in Sip-Based Multimedia Delivery." *Computers & Security* 39, Part B, no. 0 (2013): 406-418.

[Lago-Fernández 2012] Lago-Fernández, J., F. Gil-Castiñeira, F. J. González-Castaño and A. Román-Portabales. "A New Approach to Authenticating and Encrypting Voice over Internet Protocol Communications." *Software: Practice and Experience*, (2012): n/a-n/a.

[Lau 2006] Lau, Danny Hok-Ching. "Performance Measurements and Modeling of a Java-Based Session Initiation Protocol Implementation." M.A.Sc., Carleton University (Canada), 2006.

[Mohammed 2012] Mohammed, M. T., A. E. Rohiem and A. El-moghazy. "Confidentiality Enhancement of Secure Real Time Transport Protocol." In *Computer Engineering Conference (ICENCO), 2012 8th International*, 43-48, 2012.

[Persohn 2012] Persohn, Kyle. "Real-Time Transport of Internet Telephony Service Utilizing Embedded Resource-Constrained Systems." M.S., Marquette University, 2012.

[PJSIP 2014] PJSIP - Open Source SIP, Media, and NAT Traversal Library, Junho de 2014. http://www.pjsip.org/

[Ranganathan 2003] Ranganathan, Mohan Krishna and Liam Kilmartin. "Performance Analysis of Secure Session Initiation Protocol Based Voip Networks." *Computer Communications* 26, no. 6 (2003): 552-565.

[Rewagad 2013]Rewagad, P. and Y. Pawar. "Use of Digital Signature with Diffie Hellman Key Exchange and Aes Encryption Algorithm to Enhance Data Security in Cloud Computing." In *2013 International Conference on Communication Systems and Network Technologies*, edited by G. S. Tomar, M. Dixit and F. Z. Wang, 437-439. New York: Ieee, 2013.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, July 2002. http://www.rfc-editor.org/rfc/rfc3261.txt

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003, http://www.rfc-editor.org/rfc/rfc3550.txt

[RFC 3711] M. Baugher, D.McGrew, M. Naslund, E. Carrara, K. Norrman,, "SRTP: The Secure Real-time Transport Protocol," RFC 3711, March 2004, http://www.rfc-editor.org/rfc/rfc3711.txt

[RFC 5246] T. Dierks, E. Rescorla,, "The Transport Layer Security (TLS) Protocol," RFC 5246, August 2008, http://www.rfc-editor.org/rfc/rfc5246.txt

[RFC 5630] F. Audet,, "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)," RFC 5630, October 2009, http://www.rfc-editor.org/rfc/rfc5630.txt

[RFC 6189] P.Zimmermann, A. Johnston, J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP," RFC 6189, April 2011, http://www.rfc-editor.org/rfc/rfc6189.txt

[Schulzrinne 1998] Schulzrinne, Henning and Jonathan Rosenberg. "A Comparison of Sip and H. 323 for Internet Telephony." In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 83-86: sn, 1998.

[Shen 2012] Shen, C., E. Nahum, H. Schulzrinne and C. P. Wright. "The Impact of Tls on Sip Server Performance: Measurement and Modeling." *Networking, IEEE/ACM Transactions on* 20, no. 4 (2012): 1217-1230.

[Shojaie 2012] Shojaie, B., I. Saberi, M. Salleh, M. Niknafskermani and S. M. Alavi. "Improving Eap-Tls Performance Using Cryptographic Methods." In *Computer & Information Science (ICCIS), 2012 International Conference on*, 2, 760-764, 2012.

[SIREMIS 2014] SIREMIS Project - Kamailio (OpenSER) Web Management Interface by Asipto, Junho de 2014. http://siremis.asipto.com/

[Skype 2014] Skype, Fevereiro de 2014. http://www.skype.com/pt/

[Subramanian 2010] Subramanian, S. V. and R. Dutta. "Comparative Study of Secure Vs. Non-Secure Transport Protocols on the Sip Proxy Server Performance: An Experimental Approach." In *Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on*, 301-305, 2010.

[Suresh 2011] Suresh, Mahima, Ana Goulart, Unnati Desai and Walt Magnussen. "Experiments with Sip over Tls in an Ng-9-1-1 Testbed." In *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, 1-3. Chicago, Illinois: ACM, 2011.

[Thanthry 2009] Thanthry, N., G. Gopalakrishnan and R. Pendse. "Alternate Encryption Scheme for Voip Traffic." In *Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on*, 178-183, 2009.

[Wang 2011] Wang, C. H. and Y. S. Liu. "A Dependable Privacy Protection for End-to-End Voip Via Elliptic-Curve Diffie-Hellman and Dynamic Key Changes." *Journal of Network and Computer Applications* 34, no. 5 (2011): 1545-1556.

[Wang 2004]Wang, Ligang, Anjali Agarwal and J. William Atwood. "Modelling and Verification of Interworking between Sip and H.323." *Computer Networks* 45, no. 2 (2004): 77-98.

[Wu 2013] Wu, Shuhua, Qiong Pu and Fei Kang. "Practical Authentication Scheme for Sip." *Peer-to-Peer Networking and Applications* 6, no. 1 (2013): 61-74.

[Yang 2013] Yang, Menghui and Hua Liu. "Implementation and Performance of Voip Interception Based on Sip Session Border Controller." *Telecommunication Systems*, (2013): 1-17.

[Zhu 1997] Zhu, Zhenjun. "Multimedia Realtime Transport Protocol over Atm Network." M.Sc., Queen's University at Kingston (Canada), 1997.

7 Anexos

Anexo I

Version (V): 2 bits

Identifica a versão do RTP. A versão usada para a especificação atual é 2.

Padding (P): 1 bit

Indica se há bits extra que estão a ser usados no fim do pacote RTP. E são indicados os números de bits que foram usados a mais.

Extension (X):1 bit

Indica a presença de um *Extension Header* referido mais à frente.

CSRC count (CC): 4 bits

Contem o número de *CSRC identifiers* referido mais à frente.

Marker (M): 1 bit:

Se estiver ativo quer dizer que tem alguma relevância para a aplicação.

Payload Type (PT):7 bits

Indica o formato do RTP payload e determina a interpretação pela aplicação.

Sequence number: 16 bits

A sequência de números incrementa conforme o número de pacotes já enviados. E pode ser usado para ver se se perdeu algum pacote. O primeiro número é um número aleatório para tentar aumentar a segurança.

timestamp: 32 bits

Usado para reproduzir as amostras recebidas nos intervalos de tempo apropriados. Este valor é derivado de uma relógio que incrementa linearmente.

SSRC: 32 bits

O SSRC identifica a fonte da sincronização. Este identificador deve ser escolhido aleatoriamente de forma a evitar que existam duas fontes com a mesma sessão RTP.

CSRC: 0 a 15 itens, 32 bits cada

Identifica as fontes de contribuição para a conversa em causa.