UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIS ANTONIO LEIVA HÉRCULES

Impacto no Desempenho em Aplicações de Tempo Real Utilizando Criptografia

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Valter Roesler

Porto Alegre
2014

AGRADECIMENTOS

Este trabalho de graduação representa para mim a conclusão de mais uma fase da minha vida, mas mais do que isso, representa ter realizado um sonho. Desde o último dia de estudo no ensino médio na Guatemala, sonhava em estudar fora do país e poder graduar-me em uma universidade reconhecida e de maior notoriedade quando comparada às universidades do meu país. No dia 21 de fevereiro de 2015 estarei conquistando o que há sete anos atrás tanto desejava. Tudo isso não teria sido possível sem a intervenção de várias pessoas e órgãos, os quais desejo agradecer neste espaço.

Primeiramente, gostaria de agradecer a minha família. Agradeço aos meus pais, Antonio e Lizeth, que desde pequeno me deram o melhor estudo que estava em seu alcance, me ensinaram a valorizar a educação, e por terem me apoiado, auxiliado e dado a força necessária para poder estudar em outro país, distante deles. A meus irmãos Andrea e Roberto por depositarem na minha pessoa incentivo e confiança.

Gostaria de agradecer ao MEC e aos governos do Brasil e da Guatemala pela criação do programa de convênio PEC-G, que me possibilitou participar de uma experiência extraordinária. Agradeço também à UFRGS e seus professores, em especial ao meu orientador, professor Valter, por ter me guiado na realização deste trabalho. Agradeço aos colegas Felipe e Mateus do Mconf pela ajuda e suporte.

Por fim, agradeço a meus amigos, a minha namorada Stael por apoiar e estar sempre ao meu lado, e aos meus colegas, também amigos, Gustavo e Alexandre, pelo grande apoio e companheirismo durante toda essa caminhada. Dessa forma, foi possível crescer acadêmica, profissional e pessoalmente. A todos, muito obrigado!

RESUMO

Segurança é uma das mais importantes preocupações no desenvolvimento de aplicações na área de informática, sendo cada vez mais uma característica desejada pelos desenvolvedores e usuários. Tem sua importância elevada quando envolve algum tipo de informação sigilosa ou relevante para os usuários, tornando-se quase obrigatória em novas aplicações. Em aplicações de transmissão de dados em tempo real, como transmissão de voz e transmissão de vídeo, um bom desempenho do acréscimo da segurança se torna fundamental. O presente trabalho é um estudo sobre o desempenho do *Secure Real-time Transport protocol* (SRTP) que fornece segurança ao *Real-time Transport Protocol* (RTP), e sobre o impacto no desempenho na implantação dos mecanismo de segurança para o *Adobe´s Real Time Messaging Protocol* (RTMP). A segurança do RTMP se divide em dois métodos: o RTMPE e o RTMPS. O RTMPE é um mecanismo de criptografia próprio da *Adobe* e especifico para o RTMP. O segundo método é o RTMPS, que é basicamente o protocolo RTMP sobre uma conexão *Transport Layer Security*/*Secure Sockets Layer* (TLS/SSL), e sua forma de funcionamento é parecida com a do HTTP e o HTTPS. SSL é um protocolo que opera na camada acima do TCP, sendo em alguns casos implementada sobre UDP. A implementação foi feita no Red5 Media Server, que é um servidor de código aberto escrito em Java e que suporta vários formatos de vídeo e áudio, assim como os protocolos RTMP, RTMPE e RTMPS, que são os protocolos utilizados para a análise de desempenho.

Palavras-chave: RTMP. RTMPS. RTP. SRTP. SSL.

Impact on Performance Real-Time Applications Using Encryption

ABSTRACT

Security is a major concern in developing applications in information technology, increasingly being a desired feature by developers and users. Has its great importance when it involves some kind of secret information or relevant to users, making it almost mandatory in new applications. In data transmission real-time applications such as voice and video transmission, a good performance increase in safety is fundamental. The present work is a study on the performance of the Secure Real-time Transport Protocol (SRTP), which provides security to the Real-time Transport Protocol (RTP), and the performance impact of the implementation of security mechanism for Adobe's Real Time Messaging Protocol (RTMP). The safety of RTMP is divided into two methods: RTMPE and RTMPS. The RTMPE is an encryption mechanism itself from Adobe and specific for RTMP. The second method is the RTMPS, which is basically the RTMP protocol on a connection Transport Layer Security/Secure Sockets Layer (TLS/SSL), and their mode of operation is similar to that of HTTP and HTTPS. SSL is a protocol that operates above the TCP layer, and in some cases implemented over UDP. The implementation was done in Red5 Media Server, which is an open source server written in Java and supports multiple video and audio formats as well as the RTMP protocol, RTMPE and RTMPS, which are the protocols used for performance analysis.

Keywords: RTMP. RTMPS. RTP. SRTP. SSL.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AES	Advanced Encryption Standard
AMF	Action Message Format
AS	Action Script
CA	Certificate Authority
CSR	Certificate Signing Request
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized zone
DNS	Domain Name System
DoS	Denial-of-service
HMAC	Hash-based Message Authentication Code
IP	Internet Protocol
ISP	Internet service provider
LXC	Linux container
MAC	Message Authentication Code
MD5	Message-Digest algorithm 5
MitM	Man in the Middle
MKI	Master Key Identifier
OFB	Output Feedback
RSA	Ron, Adi & Leonard public-key algorithm
RC4	Rivest Shamir & Adelman Cipher 4
RTMP	Adobe´s Real Time Messaging Protocol
RTMPE	Adobe´s Real Time Messaging Protocol Encrypted
RTMPS	Adobe´s Real Time Messaging Protocol over SSL
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SHA-1	Secure Hash Algorithm
SDES	Session Description Protocol Security Descriptions
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SRTP	Secure Real-time Transport Protocol
SRTCP	Secure Real-time Transport Control Protocol

SSL	Secure Sockets Layer
SWF	Shockwave Flash extension file
TLS	Transport Layer Security
URI	Uniform Resource Identifier
VoIP	Voice over IP
XOR	Exclusive or

SUMÁRIO

1 INTRODUÇÃO

A Internet foi criada com a finalidade de comunicar dois dispositivos fim a fim, preferencialmente sem a interferência nem interceptação de terceiros. Existem casos onde a comunicação não segue o mesmo padrão fim a fim, esse é o caso de aplicações mais modernas, as quais são feitas para utilizar e aproveitar ao máximo os servidores e conteúdo distribuído. Neste trabalho, abordaremos aplicações de videoconferência onde é desejada a comunicação fim a fim. Para que tal comunicação ocorra sem a ocorrência de problemas como o citado acima, tem se procurado diversas soluções que proporcionem essa segurança desejada, já que atualmente segurança é uma das maiores preocupações, não só no desenvolvimento de aplicações, mas também na infraestrutura de rede (GORALSKI, 2009).

Quando o termo segurança surge, se pensa em criptografia, mas o termo segurança engloba varias áreas importantes de proteção, geralmente essas áreas são divididas em seis: Confidencialidade, disponibilidade, autenticação, identidade, autorização e integridade (FIRESTONE et al. 2007).

Confidencialidade entre um emissor e um receptor significa que apenas o emissor e o receptor podem interpretar os dados (FIRESTONE et al. 2007). O fato desses dados poderem ser interceptados por terceiros exige que os dados sejam criptografados para que, caso sejam interceptados, não possam ser interpretados (KUROSE; ROSS, 2012). Para estabelecer essa ligação encriptada, o remetente e receptor realizam uma troca de chave criptográfica de forma segura. Dessa forma, após ocorrer essas trocas de chave, cada um dos lados utiliza a chave para cifrar e decifrar o fluxo de dados.

Disponibilidade não só é a garantia de que os serviços e recursos oferecidos por uma infraestrutura estarão protegidos de esgotarem tais recursos em função de um ataque, mas também estarem acessíveis quando necessário (FIRESTONE et al. 2007). A disponibilidade requer proteção contra ataques de *Denial-of-service* (DoS). Ataques de negação de serviço, ou ataques DoS, são os ataques mais populares em segurança de rede, eles consistem em congestionar uma rede de forma a utilizar todos os recursos possíveis da infraestrutura para esgotar esses recursos. Com os mecanismos de defesa, tem-se que o fluxo de ataque poderá ser bloqueado antes que ele entre na rede, evitando dessa forma, o congestionamento e esgotamento de recursos que esse fluxo de ataque causaria (ROCHA, 2012).

Frequentemente, autenticação e identidade são referenciadas como sendo idênticas, mas podem ter dois significados. Um *endpoint* (terminal) pode autenticar dados para provar que os dados são válidos (criados da forma correta), mas um *endpoint* pode autenticar dados

sem autenticar a identidade. Um *endpoint* pode autenticar sua identidade mediante a apresentação de credenciais criptografadas a fim de provar sua identidade (FIRESTONE et al. 2007). Autenticação e identidade confirmam que, tanto o remetente quanto o destinatário são autênticos (KUROSE; ROSS, 2012).

Autorização é diferente de autenticação, pois autorização é o mapeamento de uma identidade autenticada para um conjunto de permissões ou capacidades para esse usuário autenticado. Atualmente a maioria dos sistemas seguros de videoconferência utilizam autenticação e autorização junto a algum tipo de conta com usuário e senha (KUROSE; ROSS, 2012).

Integridade permite assegurar que os dados recebidos pelo receptor sejam os mesmos dados que foram enviados pelo remetente, ou seja, que não houve nenhuma alteração nos dados durante a transmissão, seja por terceiros ou por erros na rede. Os métodos mais comuns de fornecer integridade são através da autenticação do conteúdo inteiro dos pacotes de dados transmitidos (FIRESTONE et al. 2007).

Sem mecanismos para garantir esses seis pontos de segurança, a infraestrutura de rede e os terminais estariam abertos a qualquer tipo de ameaça. Sendo assim, a elaboração de protocolos também tem priorizado inserir mecanismos que auxiliam em proporcionar esses quesitos de segurança mencionados acima.

1.1 Objetivos

Avaliar o impacto que o acréscimo de segurança agrega no processamento da CPU, no uso da memória, no tempo de *handshake* e na diferença de tempo entre pacotes, bem como descobrir se essa implementação é viável.

1.2 Organização do trabalho

O restante deste trabalho apresenta, no capítulo 2, um esclarecimento dos principais protocolos que proporcionam essa segurança, bem como uma analise teórica de seu desempenho em aplicações de transmissão de dados em tempo real.

O capítulo 3 apresenta o protocolo RTMP e suas variações nas quais são acrescentadas segurança, o modo de funcionamento e as ferramentas utilizadas para o desenvolvimento da análise prática do impacto no desempenho quando essa segurança é acrescentada.

O capítulo 4 apresenta a metodologia, ambiente, cenários e métricas utilizadas. No capítulo 5 são apresentados os resultados e, finalmente, as conclusões no capítulo 6.

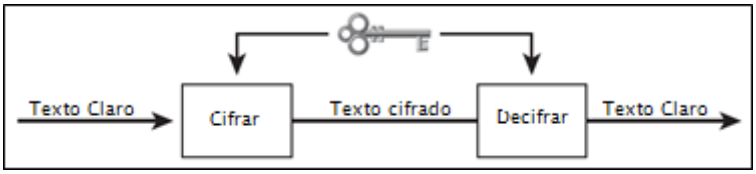
2 PRINCÍPIOS TEÓRICOS

Criptografia é a ciência ou prática de escrever em segredo. As funções criptográficas são geralmente definidas como algoritmos ou protocolos, portanto, regras que determinam como os dados são transformados de texto simples (dados não criptografados) para texto encriptado (dados criptografados) (LOSHIN, 2013).

A criptografia moderna depende basicamente de três tipos de funções: criptografia simétrica, criptografia assimétrica e funções *hash*.

Criptografia simétrica: Criptografia simétrica, ou criptografia de chave única, é o algoritmo que utiliza unicamente uma chave para cifrar o texto (cifragem) e decifrar o texto (decifração). Simétrico significa que os processos de cifrar e de decifrar são inversos um do outro (LOSHIN, 2013). O remetente cifra (com a chave) os dados que deseja enviar, e obtém como resultado o texto cifrado, logo envia para o destinatário o texto cifrado, dessa forma, se alguém interceptar o texto que está sendo enviado, não poderá interpretar os dados, pois precisará da chave para decifrar os dados e interpretá-los. Logo o destinatário decifra o texto cifrado junto com a chave e assim como resultado terá o texto claro e, com isso, poderá interpretar sem problemas os dados que foram enviados. Algoritmos de videoconferência cifram os dados usando configurações de criptografia simétrica utilizando uma única chave de tamanho fixo. A Figura 2.1 mostra este procedimento.

Figura 2.1 – Criptografia simétrica



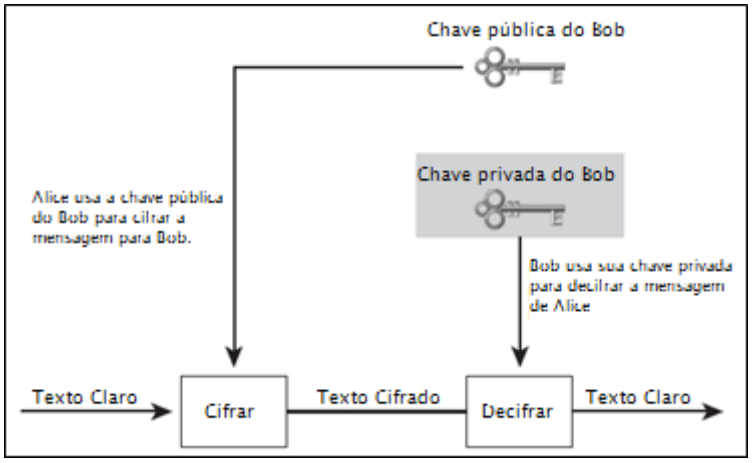
Fonte: Firestone (2007, p. 299).

A indústria de videoconferência tem utilizado cada vez mais o AES (Advanced Encryption Standard) como algoritmo de criptografia. O AES-128 é considerado como sendo altamente seguro e utiliza uma chave de 128 bits. Os algoritmos como esse, em geral são suficientemente rápidos para serem utilizados em aplicações de tempo real (FIRESTONE et al. 2007).

Criptografia assimétrica: Criptografia assimétrica, ou criptografia de chave publica, é o algoritmo que utiliza duas chaves: A chave privada e a chave pública. Assimétrica

significa que o processo de criptografia com a chave pública só pode ser invertido (decifrado) utilizando a chave privada e vice-versa (LOSHIN, 2013). Dessa forma, cada *endpoint* cria duas chaves, a chave privada que guarda secretamente, e a chave pública, disponibilizada para os outros *endpoints* com os quais realizará comunicação. Para enviar uma mensagem cifrada, o remetente deve conhecer a chave pública do destinatário, sendo assim, com essa chave o remetente cifra os dados que deseja enviar para o destinatário, após envia os dados cifrados, e quando o destinatário receber os dados cifrados, poderá com sua chave privada decifrar os dados recebidos, e dessa forma, interpretar os dados que lhe foram enviados. Por fim, se um terceiro interceptar os dados durante a transmissão, como não possui a chave privada do destinatário, não poderá interpretá-los. A Figura 2.2 demostra esse processo.

Figura 2.2 – Criptografia assimétrica



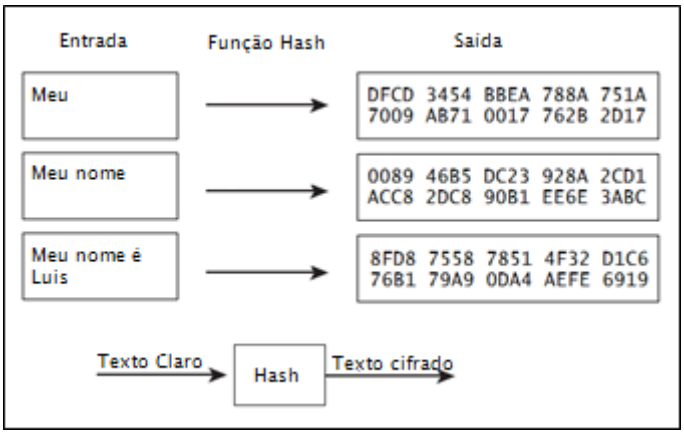
Fonte: Firestone (2007, p. 299).

Funções *Hash*: As funções *Hash*, também chamadas de *message digests* ou *one-way encryption functions*, são funções que aceitam um texto não cifrado de qualquer tamanho e, como saída geram um texto resumido fixo de pequeno tamanho (LOSHIN, 2013). O fato de ser *one-way* diz que é computacionalmente inviável executar o *hash* no sentido contrário, ou seja, dado um valor de saída *hash* os atacantes não serão capazes de montar uma sequência de bytes de entrada que gerem o *hash* de saída.

Funções *hash* garantem integridade dos dados, pois o receptor pode verificar se algum intruso alterou os dados durante a transmissão. A integridade dos dados impede ataques *man in the middle* (MitM) em sinais de controle ou no fluxo dos dados (FIRESTONE et al. 2007). Um remetente fornece um mecanismo para que o receptor possa validar a integridade dos dados pela adição de um *hash* no final de cada pacote (FIRESTONE et al. 2007). Qualquer

mudança no pacote poderá ser percebida, já que essa é outra característica das funções *hash*. Qualquer mínima mudança na entrada do *hash* provocará um resultado completamente diferente na saída. Existem vários algoritmos *hash*, mas dentre esses, o algoritmo mais utilizado é o *Secure Hash Algorithm 1* (SHA-1), o qual gera um valor de saída de 128 bits. A Figura 2.3 exemplifica o funcionamento das funções *hash*.

Figura 2.3 – Função hash



Fonte: Hércules, L. A. L. (2014).

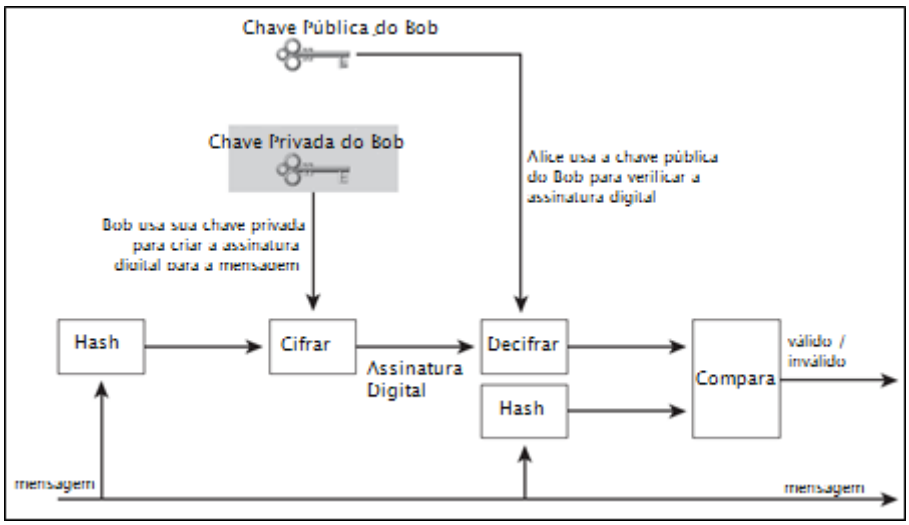
Para verificar não unicamente a integridade, mas também a autenticidade, existe um recurso adicional que as funções *hash* utilizam: O *Hash-based Message Authentication Code* (HMAC). HMAC é a combinação de uma chave secreta junto à função *hash*, HMAC pode ser utilizada com qualquer função criptográfica *hash*. Por exemplo: MD5 ou SHA-1, combinados com uma chave secreta previamente compartilhada (KRAWEZYK, 1997). *Endpoints* que enviam multimídia criptografada, geralmente oferecem confidencialidade e integridade, por tanto a criptografia proporciona a confidencialidade e HMAC proporciona a integridade.

Baseados no conceito de HMAC, mas com chave publica, existem as assinaturas digitais, ou assinaturas de mensagens, forma pela qual pode se realizar autenticação e integridade usando criptografia de chave pública para cifrar os valores de *hash*. A assinatura de mensagens se baseia no fato de que os dados decifrados por uma chave pública só podem ter sido cifrados pela chave privada correspondente.

Funciona da seguinte maneira, o remetente calcula o *hash* da mensagem que vai ser enviada. Pode ser utilizando qualquer algoritmo. Por exemplo, MD5 ou SHA-1, em seguida cifra o *hash* usando sua chave privada, esse *hash* resultante é chamado de assinatura digital. Qualquer destinatário pode decifrar o *hash* e, em seguida, verificar o *hash* com o conteúdo da mensagem. Assim, da mesma forma que a criptografia assimétrica, todos os destinatários

devem conhecer a chave publica do remetente para poder realizar a verificação do *hash*. Praticamente, essa é a forma do certificado digital garantir a autenticidade dele. O certificado digital se diferencia principalmente por ele ter sido gerado por uma Autoridade Certificadora. A Figura 2.4 mostra o funcionamento das assinaturas digitais.

Figura 2.4 – Assinatura digital



Fonte: Firestone (2007, p. 299).

Portanto, existem inúmeros protocolos que combinam vários ou todos os mecanismos de segurança acima mencionados, porém de formas distintas. Os principais desses protocolos serão apresentados a seguir.

2.1 SRTP

O Secure Real-time Transport Protocol (SRTP) é um protocolo que fornece segurança para o Real-time Transport Protocol (RTP), é um protocolo altamente utilizado para envio de *streaming* de vídeo e áudio. O RTP conta com um protocolo de controle, o Real-time Transport Control Protocol (RTCP), o qual se encarrega de fornecer informações de controle e estatística do fluxo RTP fora da banda (conexão diferente da qual trafegam os dados) (BEGEN et al. 2011). Dessa forma, deixa o RTP encarregado unicamente de carregar a mídia em si. O SRTP fornece confidencialidade, integridade, autenticação de mensagens e proteção contra ataques de *replay* para esses dois protocolos (BAUGHER et al. 2004).

O SRTP não só define um conjunto de transformações padrão de criptografia, mas também permite que novas transformações possam ser introduzidas futuramente, levando em consideração alguns cuidados. Uma transformação consiste em capturar o *payload* (carga útil) e alguns dados associados de uma aplicação, como entrada, e transformar esses dados de entrada, de modo a garantir a privacidade do *payload* e a integridade do *payload* junto com os dados associados (KOHNO et al. 2003).

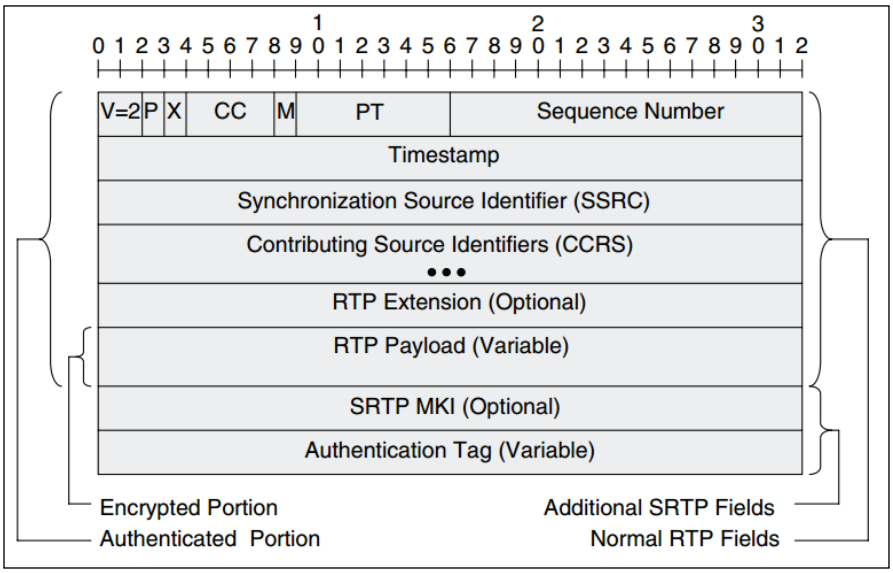
Com um gerente de chaves apropriado, o SRTP garante segurança para aplicações *unicast* e *multicast*, também pode alcançar uma baixa expansão dos pacotes e, portanto um alto *throughput* (vazão de dados). Além disso, prova que fornece a proteção adequada para ambientes heterogêneos, ou seja, ambientes onde existem redes com e sem fio. Para atingir essas características, suas transformações padrão são descritas baseando-se em cifra de fluxo aditivo para criptografia. O SRTP faz uso de uma função *hash* com chave para a autenticação da mensagem e utiliza um índice implícito para sequência e sincronização, esse índice no SRTP é baseado no número de sequência do pacote RTP e no número de índice para o SRTCP (BAUGHER et al. 2004).

Dessa forma, o SRTP não só atesta a confidencialidade dos pacotes RTP e RTCP, como também garante a integridade dos pacotes RTP e RTCP por inteiro, simultaneamente, e com a proteção contra pacotes repetidos (ataques de *replay*). Esses serviços são independente uns dos outros e facultativos, exceto a proteção de integridade dos pacotes de controle (pacotes SRTCP). Para esses pacotes os serviços são obrigatórios, dado que um erro ou mensagem maliciosa no RTCP poderia prejudicar o processamento do fluxo de dados RTP. O SRTP também possui um recurso para auxiliar no gerenciamento de chaves e conferir ainda maior segurança, um exemplo, seria a possibilidade de configurar a derivação de chaves de sessão para que seja atualizada periodicamente, sendo assim, limita a quantidade de dados transmitidos por uma chave fixa.

Conceitualmente, o SRTP funciona como uma implementação que reside entre a aplicação RTP e a camada de transporte. O SRTP intercepta os pacotes RTP, processa-os, e logo transmite os pacotes SRTP equivalentes para a camada de transporte no lado do envio. Logo o receptor intercepta os pacotes SRTP, processa esses pacotes extraindo a informação correspondente ao pacote RTP equivalente e passa o pacote RTP para a camada superior (BAUGHET et al. 2004). O formato do pacote SRTP é mostrado na Figura 2.5, destacando a parte que é cifrada e a parte que é autenticada. Também são destacados os campos originais do pacote RTP e os campos adicionais referentes aos pacotes SRTP.

Na Figura 2.5, *Encrypted Portion* consiste em cifrar a carga útil dos pacotes RTP, incluindo *padding*, quando este estiver presente. A parte cifrada pode ser do mesmo tamanho do pacote RTP ou maior. As transformações padrão pré-definidas não fazem uso do padding, nesse caso o *payload* do RTP e do SRTP são do mesmo tamanho. As transformações pré-definidas também garantem um baixo custo computacional (BAUGHER et al. 2004).

Figura 2.5 – Formato de um pacote SRTP



Fonte: Systems Cisco (2009, p. 12).

São adicionados dois campos no pacote SRTP que não se encontram no RTP: o *Master Key Identifier* (MKI) e o *Authentication Tag.* O MKI é um campo opcional de tamanho configurável utilizado para informar a chave mestre, a partir da qual serão obtidas as chaves de sessão para cifrar e autenticar. O *Authentication Tag* é um campo recomendado, utilizado para armazenar os dados de autenticação de mensagens do seu cabeçalho RTP e o *payload* referente a ele (SYSTEM CISCO, 2009).

O SRTCP fornece os mesmos serviços de segurança para o RTCP da mesma forma como o SRTP faz com RTP. A autenticação de mensagens no SRTCP é obrigatória, assim protege os campos RTCP e também mantém os contadores de sequência dos pacotes (BAUGHER et al. 2004). Na criação dos pacotes SRTCP são adicionados três campos: O *Encryption Flag* com o índice SRTCP, o MKI referente ao SRTCP e o *Authentication Tag.*

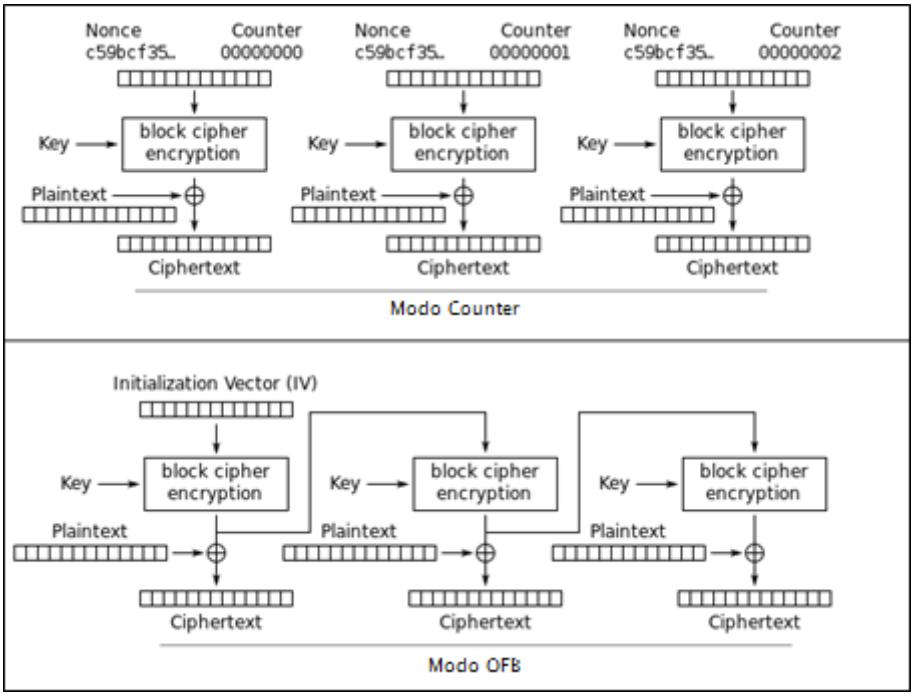
Encryption Flag vai indicar se o pacote está cifrado ou não. O índice SRTCP se faz necessário e é incrementado a cada pacote SRTP enviado, isto para evitar os ataques de replay (SYSTEM CISCO, 2009). O campo MKI do SRTCP é um campo opcional o qual indica a

chave mestre, a partir da qual serão obtidas as chaves de sessão para cifrar e autenticar. O *Authentication Tag*, que nos pacotes SRTCP é obrigatório e de comprimento configurável, é utilizado para armazenar os dados de autenticação de mensagens para o seu cabeçalho RTCP e o seu *payload*.

A transformação de criptografia definida no SRTP mapeia o índice e a chave secreta do pacote SRTP em um segmento *keystream* pseudoaleatório (EASTLAKE 1994). Um *keystream* é definido como um fluxo de caracteres aleatórios que quando combinados com um texto, resulta em um texto cifrado. Cada segmento *keystream* é capaz de cifrar um pacote RTP. O processo de cifrar um pacote consiste em gerar um segmento *keystream* correspondente ao pacote, e em seguida, produzir, bit a bit, a parte cifrada do pacote SRTP. Para decifrar o processo é similar, porém trocando os papéis do texto claro e do texto cifrado.

Embora existam numerosos algoritmos para realizar a criptografia e autenticação de mensagens que possam ser utilizadas no SRTP, serão mostrados unicamente os algoritmos prédefinidos. A cifra padrão é o AES, para cifrar no SRTP são definidos dois modos de funcionamento do AES, o AES no modo *Counter Mode* e o AES no *f8-mode* que é uma variação do modo *output feedback mode* (OFB).

Figura 2.6 – Diagrama do counter mode e do OFB mode



Fonte: Mark (2008).

Conceitualmente, o modo *counter* consiste em cifrar inteiros sucessivos com o objetivo de embaralhar o ponto de partida da sequência inteira. Cada pacote é codificado com um segmento *keystream* distinto e uma chave de sessão. O modo f8 se diferencia basicamente do modo *counter* por ter a característica de passar por um processo de realimentação, ele gera blocos *keystream*, e então é aplicado um XOR com o bloco de texto simples para obter o texto cifrado. Com isso, essa *keystream* será utilizada como entrada junto com a chave de sessão para cifrar o próximo bloco. A Figura 2.6 ilustra as diferenças entre o *counter mode* e o OFB, nessa figura, a saída do *block cipher encryption* se refere à *keystream*.

Existe também a *NULL cipher* (cifra nula), que é utilizada quando não é solicitada confidencialidade para o RTP/RTCP. O *keystream* pode ser pensado como uma sequência de zeros, ou seja, a criptografia terá como função simplesmente copiar a entrada de texto claro para a saída de texto cifrado.

Os algoritmos de criptografia mencionados anteriormente protegem contra confidencialidade, mas para proteção da integridade e autenticação das mensagens, o SRTP faz uso de um outro mecanismo. Uma função *hash* com chave secreta e código de autenticação de mensagem (MAC), o HMAC. O SRTP faz uso específico do HMAC-SHA1 na qual a saída HMAC serão os bits mais à esquerda. Já para proteção contra ataques de *replay* o SRTP faz através de um receptor, que tem como função manter os índices das mensagens previamente recebidas. Sendo assim, o SRTP só processará as mensagens que apresentem um índice não recebido anteriormente, caso contrário, a mensagem será descartada (BAUGHER et al. 2004).

2.2 Secure SIP

O *Session Initial Protocol* (SIP) é um protocolo de iniciação de sessão utilizado amplamente em sistemas de videoconferência. O SIP também é um protocolo de aplicação utilizado para estabelecer, modificar e terminar sessões de comunicação multimídia entre um ou mais participantes (ROSENBERG et al. 2002). Entre essas sessões multimídia estão inclusas conferências multimídia, ensino à distância, telefonia via Internet e aplicações similares. Para tal, o SIP utiliza como porta padrão a porta 5060 (FIRESTONE et al. 2007).

O padrão SIP define um método para estabelecer uma conexão segura SIP, isto é, o SIP faz uso do TLS pela porta 5061. Para a comunicação segura SIP são definidos os SIPS *Uniform Resource Identifier* (URI), usado da mesma maneira que o HTTPS é usado para

conexões seguras HTTP, ou seja, o endereço SIP utilizado habitualmente *sip:URL* é substituído pelo *sips:URL*. A URI SIPS garante o uso de SIP sobre TLS entre cada par de saltos para validar a conexão, fornecendo assim, uma conexão segura fim a fim (WARD, 2006).

O TLS oferece a autenticação de um só lado ou mútua, e fornece criptografia e integridade para o fluxo de dados em ambas as direções. A desvantagem da utilização do TLS é que seu mecanismo de segurança funciona salto por salto. Para a conexão fim a fim ser segura, todos os dispositivos ao longo do caminho devem confiar uns nos outros. As mensagens de sinalização SIP podem especificar o uso de SRTP para a cifragem na transmissão de mídia (FIRESTONE et al. 2007).

2.3 SDES

O *Session Protocol Security Descriptions* (SDES) é uma forma segura de negociar as chaves para o SRTP (ANDREASEN et al. 2006). O SDES define um novo atributo chamado *crypto*, esse atributo é usado para negociar a suíte de criptografia, os parâmetros-chave e parâmetros de sessão. O atributo *crypto* contém informações utilizadas na transmissão, como o algoritmo de criptografia e autenticação e chave mestre. O atributo contém quatro campos: *Tag*, *crypto-suite*, *key-params* e o campo opcional *session-params* (PUANGPRONPITAG et al. 2002).

O campo *tag* é um número decimal utilizado como identificador de um atributo de criptografia. O campo *crypto-suite* é um identificador que descreve os algoritmos de criptografia e autenticação utilizados para o fluxo de dados. Nesse campo existem três parâmetros possíveis: AES_CM_128_HMAC_SHA1_80, AES_CM_128_HMAC_SHA1_32 e F8_128_HMAC_SHA1_80. Cada parâmetro é representado pela Tabela 2.1.

O campo *key-params* é um conjunto de materiais de chaves para *crypto-suite*, consiste no método da chave e informações de codificação. Por fim, o campo *session-params* especifica o tipo de transporte e é um campo opcional.

As chaves são transmitidas como anexo do *Session Description Protocol* (SDP) dentro de uma mensagem SIP, então, um *endpoint* envia a proposta de chave para o outro *endpoint*, e em seguida, a outra parte aceita a chave e o fluxo criptografado de dados prossegue. Todas essas informações remetem afirmar que a camada de transporte SIP deve certificar-se de que ninguém pode ver o seu anexo, ou seja, de que essa deve ser uma conexão SIP protegida

contra terceiros por meio de cifragem e autenticação. Caso contrário, a chave poderia ser interceptada (PUANGPRONPITAG et al. 2012). A principal vantagem desse método encontra-se na sua simplicidade.

Tabela 2.1 – Parâmetros do campo crypto-suite

	AES_CM_128_ HMAC_SHA1_80	AES_CM_128_ HMAC_SHA1_32	F8_128_ HMAC_SHA1_80
Master key length	128 bits	128 bits	128 bits
Master salt length	112 bits	112 bits	112 bits
SRTP lifetime	2^48 packets	2^48 packets	2^48 packets
SRTCP lifetime	2^31 packets	2^31 packets	2^31 packets
Cipher	AES Counter Mode	AES Counter Mode	AES F8 Mode
Encryption key	128 bits	128 bits	128 bits
MAC	HMAC-SHA1	HMAC-SHA1	HMAC-SHA1
SRTP auth. Tag	80 bits	32 bits	80 bits
SRTCP auth. tag	80 bits	80 bits	80 bits
SRTP auth. key len.	160 bits	160 bits	160 bits
SRTCP auth. key len.	160 bits	160 bits	160 bits

Fonte: Andreasen et al. (2006, p. 16).

As descrições acima explicam porque a troca de chaves SDES funciona unicamente através de um SIP Seguro, usando TLS, um canal de comunicação seguro, que trabalha similarmente ao HTTPS.

2.4 TLS/SSL

O principal objetivo do protocolo TLS e de seu antecessor SSL, é proporcionar privacidade e integridade de dados entre duas aplicações que se comunicam (FREIER et al, 2011). O TLS é composto por duas camadas: a *TLS Record Protocol* e o *TLS Handshake Protocol* (DIERKS; RESCORLA, 2008). O *TLS Record Protocol* fornece conexão segura com duas propriedades básicas: A conexão privada e a conexão confiável.

Conexão privada: A criptografia simétrica é utilizada para cifrar os dados. As chaves para essa criptografia simétrica são geradas exclusivamente para cada conexão, e são

negociadas em segredo por outro protocolo como o protocolo *TLS Handshake Protocol*. O *TLS Record Protocol* também pode ser utilizado sem criptografia.

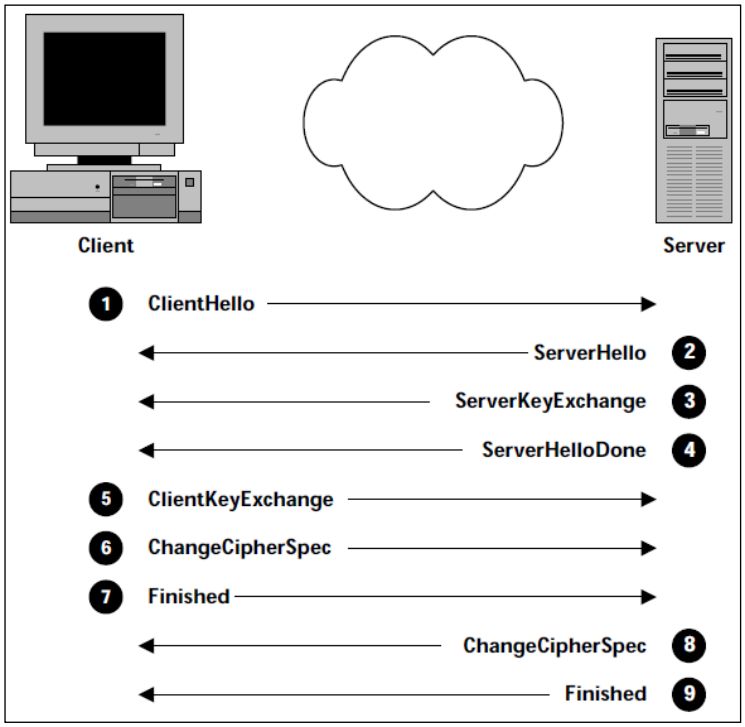
Conexão confiável: Transmite as mensagens e também inclui uma verificação de integridade usando um MAC com chave e funções *hash* como SHA e MD5, que são utilizadas em conjunto com a verificação MAC. O *TLS Record Protocol* pode operar sem um MAC, mas geralmente é usado quando outro protocolo o utiliza como um transporte para negociar os parâmetros de segurança.

O *TLS Handshake Protocol* permite a autenticação do servidor e do cliente. O TLS tem a função de negociar o algoritmo de cifragem e as chaves de criptografia antes que o protocolo de aplicação comece a receber ou transmitir dados. Este protocolo fornece conexão segura com três propriedades básicas que são a identidade, a negociação de um segredo compartilhado ser segura e a negociação segura.

Identidade: A identidade do par pode ser autenticada usando criptografia assimétrica, ou de chave pública. Essa autenticação pode ser opcional, mas é geralmente necessária para pelo menos um dos pares.

A negociação de um segredo compartilhado é segura: O segredo negociado não está disponível para terceiros, nem para uma conexão autenticada, mesmo por um atacante que se colocar no meio da transmissão.

Figura 2.7 – Mensagens SSL para estabelecer conexão cifrada



Fonte: Thomas (2000, p. 40).

A negociação é segura. Nenhum atacante pode alterar a comunicação/negociação sem ser detectado pelas partes da comunicação.

Para estabelecer a conexão segura entre o cliente e o servidor é necessária a troca de mensagens do *TLS Handshake Protocol*. A Figura 2.7 mostra a troca de mensagens necessárias e a Tabela 2.2, apresentada em seguida resume os passos mostrados na figura.

Tabela 2.2 – Passos para o protocolo SSL estabelecer a conexão cifrada

Passos	Ação
1	O cliente envia a mensagem *ClientHello* propondo opções SSL.
2	O servidor responde com a mensagem *ServerHello* selecionando as opções SSL.
3	O servidor envia a informação da sua chave pública na mensagem *ServerKeyExchange*.
4	O servidor conclui a sua parte da negociação com a mensagem *ServerHello-Done*.
5	O cliente envia a informação da chave de sessão (cifrada com a chave pública do servidor) na mensagem *ClientKeyExchange*.
6	O cliente envia a mensagem *ChangeCipherSpec* para ativar as opções negociadas para todas as mensagens futuras que serão enviadas.
7	O cliente envia a mensagem *Finished* para permitir que o servidor verifique as opções recentemente ativadas.
8	O servidor envia a mensagem *ChangeCipherSpec* para ativar as opções negociadas para todas as mensagens futuras que serão enviadas.
9	O servidor envia a mensagem *Finished* para permitir que o cliente verifique as opções recentemente ativadas.
Após estes nove passos, começa a transmissão dos dados da aplicação.	

Fonte: Thomas (2000, p. 41).

Uma vantagem do TLS é que ele é independente do protocolo da camada de aplicação, ou seja, esses protocolos de nível superior usam TLS de forma transparente. Entretanto, o padrão TLS não especifica como esses protocolos vão adicionar segurança com TLS. As decisões sobre como iniciar o TLS e como interpretar os certificados de autenticação trocados são deixadas a cargo da implementação dos protocolos que são executados em cima do TLS (DIERKS; RESCORLA, 2008).

Portanto, TLS/SSL é um protocolo que utiliza os dois tipos de criptografia, assimétrica e simétrica. Inicialmente, utiliza a criptografia assimétrica para poder realizar a troca de chaves de forma segura e assim continua com a comunicação de dados com criptografia

simétrica. Seu funcionamento fica entre a camada de aplicação e a camada de transporte, desta forma, o TLS/SSL pode ser aplicado em diversos protocolos de aplicação (FREIER et al. 2011).

2.5 Trabalhos relacionados

De acordo com as pesquisas feitas na literatura, foi encontrado um estudo relacionado no qual foi feita uma análise do desempenho do SRTP, quando ele é utilizado para proteger as conversas de VoIP (ALEXANDER et al. 2009). O experimento foi realizado com softphone Snom executado no sistema operacional Windows XP SP2, e softphone Twinkle, sobre o sistema operacional GNU/Linux Ubuntu 8.04 kernel 2.6.24-16. Foi testado o SRTP com transformações prédefinidas em codificação AES no *counter mode*, chave de 128 bits e HMAC-SHA1 com marca de autenticação de 32 bits, assim como chave de 192 bits e 256 bits, e marca de autenticação de 80 bits.

Nesse trabalho, foram realizadas várias medidas, e os resultados mostraram que o processamento de autenticação é mais caro do que o processo de cifragem em sí, independente do tamanho da chave. Foi encontrado também um aumento de pacote, sendo assim, o atraso é mínimo. Como conclusão, foi observado que o rendimento do SRTP é bom, a sobrecarga é insignificante e não tem efeito observável na qualidade do VoIP.

3 PROJETO PRÁTICO

Analisar o protocolo RTMP da Adobe, e suas principais variações RTMPT, RTMPE e RTMPS, utilizando o servidor Red5. Para tanto, foram transmitidos 3 vídeos de configurações diferentes.

Será realizada uma análise do desempenho que o acréscimo de segurança implica em uma aplicação quando é utilizada uma transmissão segura, isso é, quando se utilizam os mecanismos extras que proporcionam segurança à transmissão. Neste capítulo serão mostradas as ferramentas que foram utilizadas e como elas funcionam.

3.1 Conceito

O Adobe Flash Player desenvolvido incialmente pela Macromedia, mas que agora pertence à *Adobe Systems*, é um software reprodutor de multimídia amplamente utilizado nos navegadores web. Geralmente está embutido nesses navegadores, quando não, está disponível como *plugin*. Inicialmente criado para animações, mas com suporte para imagens e vídeos, o Flash Player é um software que executa unicamente arquivos de leitura. Esses arquivos podem ter seu conteúdo exibido, porém, não podem ser modificados.

Tecnicamente, é uma máquina virtual utilizada para executar arquivos Shockwave Flash (SWF), que é um formato de arquivos executáveis de aplicações web. Tem como característica suportar conteúdo multimídia, além de ser relativamente leve. Contém suporte para a linguagem *ActionScript* (AS) da Adobe e sua capacidade pode ser comparada com a linguagem JavaScript.

Para transmissão de vídeo, áudio e dados pela Internet, a Adobe criou um protocolo específico, o RTMP. O RTMP é um protocolo simples desenvolvido para atuar sobre qualquer protocolo da camada de transporte, mas que geralmente trafega sobre o protocolo TCP.

3.2 RTMP

O Adobe's *Real Time Messaging Protocol* (RTMP) é um protocolo destinado a transportar fluxos paralelos de vídeo, áudio, mensagens de dados, assim como informações de

tempo associadas. O protocolo é específico para o Flash Player (PARMAR; THORNBURGH, 2012).

O RTMP é um protocolo simples, seus pacotes são enviados através de uma conexão TCP estabelecida entre o cliente e o servidor, que por padrão utiliza a porta 1935. Para serializar (processo de transmitir um objeto por uma conexão de rede) e desserializar objetos, enquanto eles são enviados pela rede, o RTMP faz uso de um formato binário compacto chamado *Action Message Format* (AMF).

O AMF é um formato binário utilizado para serializar gráficos de objetos, como o ActionScript (programação orientada a objetos própria da Adobe e utilizada no RTMP), ou enviar mensagens entre um cliente Flash e um servidor. O AMF e o RTMP em conjunto, são utilizados em duas funções: estabelecer conexão e para o controle que realiza a entrega de *streaming* de mídia. Para o *streaming* de mídia, os dados do AMF são encapsulados no cabeçalho o qual define as características da mensagem, como o tipo ou o comprimento. As mensagens são utilizadas unicamente no nível da aplicação, quando prontas elas são fragmentadas, e então convertidas em *chunk*, de tamanho variável para proporcionar maior flexibilidade, e enviadas pela rede.

O protocolo RTMP tem múltiplas variações além da forma padrão, sendo que duas dessas variações tem a função de conferir segurança ao protocolo. As duas variações que oferecem segurança são: RTMPE e RTMPS (TOWERS; GREEN, 2010).

O RTMPE é um mecanismo básico de criptografia próprio da Adobe que criptografa o protocolo RTMP pelo Flash Media Server em tempo de execução. O RTMPE utiliza as primitivas padrão de criptografia que consistem na troca de chaves Diffie-Hellman, e o *hash* de autenticação HMAC-SHA256 [Help 2014]. Enquanto os dados são transmitidos, o RTMPE gera um par de chaves RC4, uma chave tem a função de cifrar os dados enviados pelo servidor, e a outra chave tem a função de cifrar os dados enviados para o servidor. O RC4 não é considerado um dos melhores algoritmos simétricos de criptografia, contudo, existe o RTMPS que confere maior segurança para o RTMP (HELP, 2014).

O RTMPS é um protocolo originado quando a criptografia de certificados SSL é utilizada. O uso da criptografia SSL requer a utilização da versão do Flash Player 8 ou superior (TOWERS; GREEN, 2010). O RTMPS utiliza HTTPS e normalmente se comunica pela porta 443.

Existe também o RTMPT, uma variação do RTMP, mas que não envolve o uso de criptografia. Diferentemente das mencionadas acima, o RTMPT tem como objetivo poder transitar mesmo quando conexões diretas no socket RTMP não são permitidas, seja pela

existência dos *firewalls* ou servidores de algumas organizações. Basicamente, o RTMPT funciona como um tunelamento, onde os dados RTMP são encapsulados e trocados via HTTP, e as mensagens do cliente são dirigidas para a porta 80 no servidor. A sessão encapsulada pode conter pacotes RTMP simples, RTMPS, ou RTMPE (HELP, 2014).

3.3 Red5

Criado em 2005, o Red5 Media Server, atualmente na versão 1.0.1 (lançada em 2013) (Gregoire, 2014), oferece uma poderosa solução de *streaming* de vídeo multiusuário para o Adobe Flash Player. Com base em Java e alguns dos mais poderosos *frameworks* de código aberto, o Red5 se destaca como uma solução sólida.

O Red5 inclui suporte para as novas APIs multiusuário, incluindo NetConnection, NetStream e SharedObject proporcionando implementações *servlet* poderosas com RTMP. Além do suporte para o protocolo RTMP, o servidor tem um container Tomcat Servlet embutido para aplicações web Java Enterprise Edition (JEE).

3.4 Processo de implementação

Em pesquisa feita buscando um servidor com suporte para RTMP foram encontrados os dois mais populares, o Wowza e o Red5. Ambos são escritos em Java, porém apenas o Red5 é gratuito e, por este motivo, foi escolhida a sua utilização.

Inicialmente foi pensado criar todo o ambiente de implementação dentro do computador, e fazer a análise localmente. Nessa etapa foi utilizado *Linux Containers* (LXC) para instalar dentro o servidor Red5. Dento da LXC seria criado um certificado autoassinado e instalado, dessa forma seria possível utilizar o protocolo RTMP sobre SSL (RTMPS) e poderia ser feita a análise desse protocolo assim como as outras variações principais RTMPE e RTMPT. No entanto, foi encontrada uma grande dificuldade com o uso do certificado autoassinado dentro do Red5. Algumas pesquisas bibliográficas foram realizadas com o intuito de encontrar uma solução na literatura para o impasse, porém, a mesma dificuldade foi relatada pelas referências pesquisadas. Através de comentário, um dos desenvolvedores do Red5 recomendou fortemente a utilização de certificados reais, justificando a existência de uma grande complexidade no uso de certificados autoassinado.

Para contornar a dificuldade pelo Red5 na utilização de certificado autiassinado, houve a necessidade de adquirir um certificado emitido por uma *Certificate authority* (CA). Para uma CA emitir um certificado digital é necessário cumprir com uma serie de requisitos. São requisitos básicos: Ter um registro atualizado que será consultado pela CA, e ter o *Certificate Signing Request* (CSR).

CSR é um arquivo de pedido da assinatura. O CSR é um bloco de texto cifrado gerado no servidor que contém informações como o domínio, cidade, país e chave pública, que serão incluídas no certificado.

Os passos para obter um certificado são:

Registro: Para se adquirir um certificado é necessário possuir um domínio particular, já que a CA deve assegurar que você possui o nome de domínio do qual está recebendo o certificado e que você está autorizado a solicitar o certificado. Isso é feito basicamente conferindo o registro WHOIS, dessa forma os dados do registro e os dados informados no CSR devem coincidir.

CSR: A solicitação de assinatura de certificado, ou CSR é um pedaço de texto criptografado que deve ser gerado no servidor antes de solicitar o certificado SSL. A CA irá utilizar as informações contidas no CSR (nome da organização, nome do domínio, localização e chave pública) e conferir elas no WHOIS, para gerar o certificado SSL.

Enviar CSR: Depois de ter o domínio e empresa validados, é necessário enviar o CSR para a CA, a CA então envia o certificado SSL, o qual pode ser instalado no servidor (SHOPPER^{TM}, 2014).

4 METODOLOGIA

A fim de avaliar o desempenho quando são acrescentadas algumas características em protocolos de aplicações de tempo real, foi dada ênfase teórica em dois protocolos, RTP e RTMP. Pela existência do trabalho relacionado citado anteriormente sobre a análise do RTP, e pelo amplo uso na web do protocolo RTMP, a ênfase prática deste trabalho de graduação será sobre o protocolo da Adobe, o RTMP. A seguir, será mostrado o processo executado para a realização do projeto prático e dos testes efetuados.

4.1 Ambiente

O ambiente foi criado seguindo os critérios descritos na seção 3.4. Primeiro era necessária a utilização de um servidor, para tal, foi utilizado um computador Sony com processador Intel® Core™ i3-2330M CPU com frequência de 2,20GHz, 4 núcleos (*cores*) lógicos, 2 núcleos físicos, memória RAM de 4GB e Sistema Operacional GNU/Linux Ubuntu 14.04 LTS de 64-bit.

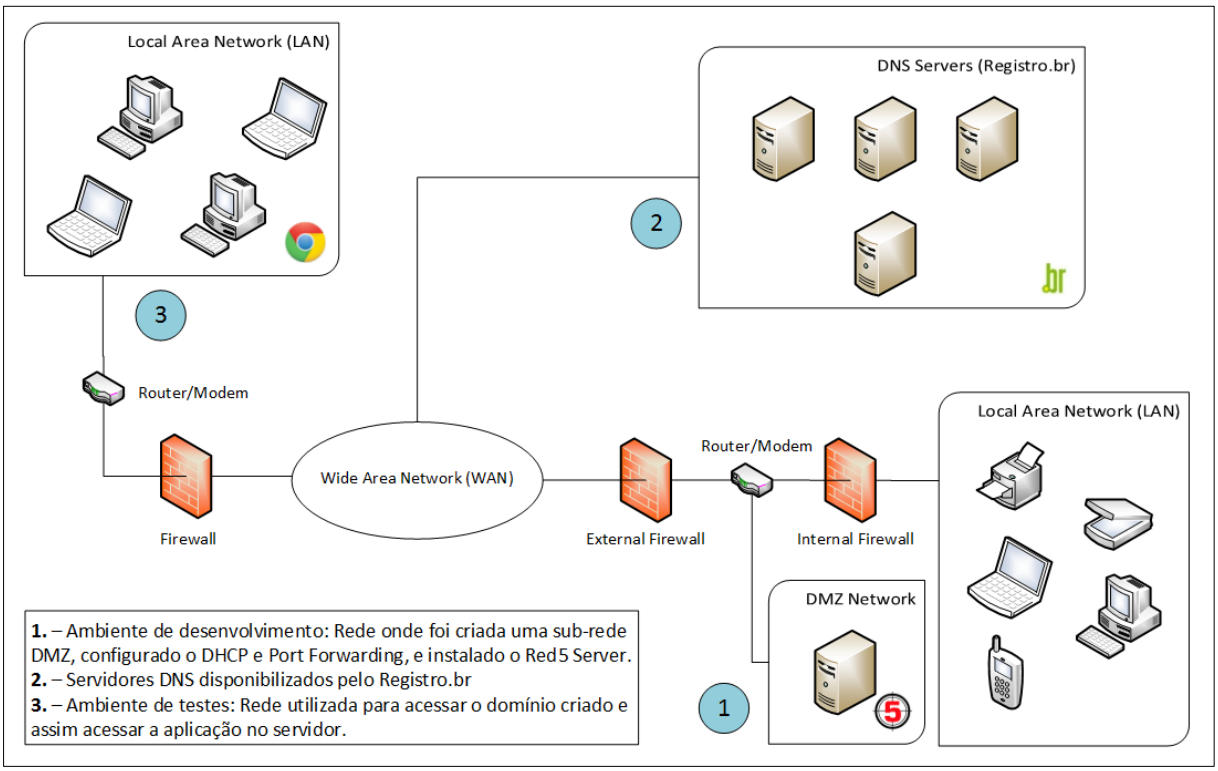
Posteriormente, foi criado um domínio da categoria *.com.br,* esse foi solicitado para o Núcleo de Informação e Coordenação do Ponto BR (nic.br), pelo site do Registro.br. O Registro.br cuida desde 1995, de nomes de domínios, da administração e publicação do *Domain Name System* (DNS) para o domínio *.br*, além de cuidar dos serviços de distribuição e manutenção de endereços Internet. Para um domínio ser vinculado ao IP do servidor e para que, dessa maneira, ele possa ser encontrado na Internet, é necessário o uso de DNS. O Registro.br disponibiliza o uso de seus servidores DNS gratuitamente, portanto, foram utilizados esses servidores DNS, fazendo que o nosso domínio criado apontasse para o IP externo do servidor, nesse caso é o IP do roteador da rede onde o servidor se encontra.

Para o servidor (que é um computador pessoal dentro de uma rede local) ser acessível externamente, foram necessárias algumas configurações no roteador local, pois se utilizou a rede do provedor de serviços (*Internet Service Provider* – ISP). Primeiro foi necessário alterar o DHCP do roteador, para atribuir sempre o mesmo IP para o servidor, assim mesmo que o servidor fosse desligado, ou reiniciado, sempre utilizaria o mesmo IP. Esse IP estático seria útil para poder alterar o encaminhamento de portas. A aplicação do servidor Red5 utiliza como padrão a porta 5080, portanto, no roteador foi criado o encaminhamento da porta 80 para a porta 5080, dessa maneira, a aplicação pode ser acessada externamente pela porta 80.

Após foi criada uma *demilitared zone* (DMZ) apontando para o IP estático configurado anteriormente. DMZ é uma sub-rede física, ou lógica, que contém e expõe serviços de fronteira externa. Sua função é manter todos os serviços que possuem acesso externo e assim possibilita o acesso externo em algumas portas diferentes da porta 80, as quais são necessárias para a comunicação dos protocolos da aplicação do Red5 Server. A versão do Red5 Server instalada no servidor foi a versão 1.0.1, versão mais atualizada.

Foi gerada uma chave pública de 2048 bits cifrada com o algoritmo RSA. A partir dessa chave foi criado o CSR, e para esse procedimento, foi utilizada a ferramenta *keytool*. Esse CSR foi submetido na CA Thawte. A CA gerou o certificado SSL, então foi configurado o Red5 Server para ser possível o envio do protocolo RTMP sobre uma conexão SSL com esse certificado. O certificado recebido utiliza como algoritmo de criptografia assimétrica o RSA, com chaves pública e privada de 2048 bits e como algoritmo de assinatura digital o SHA-1. A Figura 4.1 ilustra e explica o ambiente de desenvolvimento.

Figura 4.1 – Ambiente de desenvolvimento



Fonte: Hércules, L. A. L. (2014).

Vale ressaltar que cada uma das três partes destacadas na figura 4.1 encontram-se localizadas em posições geograficamente distintas e distantes, pertencendo a *Internet service provider* (ISP) diferentes. Portanto, o ambiente de desenvolvimento é um ambiente real.

4.2 Cenário de testes

Os cenários de testes foram realizados em um computador Samsung com processador Intel® Core™ i3-2328M, CPU com frequência de 2,22GHz, 2 núcleos físicos, 4 núcleos lógicos, memória RAM de 4GB, dos quais 3,71 GB são utilizáveis e Sistema Operacional Windows 7 64-bit.

Tabela 4.1 – Configurações dos vídeos de teste

	Vídeo 1	Vídeo 2	Vídeo 3
Qualidade	Standard Definition	Standard Definition	High Definition (1080p)
Tamanho	6,5 MB (6544259 bytes)	3,1 MB (3137481 bytes)	65,6 MB (65554746 bytes)
Formato	flv	Flv	mp4
Codec	On 2 VP6/Flash	On 2 VP6/Flash	H.264
Duração do vídeo	2 min. 32 seg.	1 min. 8 seg.	4 min. 7 seg.
Duração do teste	2 min. 32 seg.	1 min. 8 seg.	20 seg.

Fonte: Hércules, L. A. L. (2014).

Duas transmissões ao vivo nunca são idênticas, elas geram uma quantidade diferente de dados enviados em cada transmissão e, portanto, a comparação dessas transmissões não forneceria um resultado factível e confiável. Para contornar essa adversidade no momento da realização dos testes, foram utilizados e comparados três vídeos diferentes. Por se tratarem de vídeos, e não de transmissões ao vivo, tornou-se possível a comparação dos parâmetros desejados. Cada um dos vídeos foi transmitido pelo protocolo RTMP e suas três principais variações, RTMPT, RTMPE e RTMPS. Foi criada uma página, na qual podem ser reproduzidos os três vídeos diferentes, e cada vídeo pode ser reproduzido em um dos quatro protocolos mencionados anteriormente. A Tabela 4.1 mostra as configurações desses três vídeos.

Como o Flash Média Player é voltado para reproduzir vídeos em navegadores web, os testes foram realizados no navegador Google Chrome Version 39.0.2171.71 m. Além do Google Chrome, foram utilizados mais dois softwares para auxiliar a coleta dos dados, o Wireshark versão 1.12.2 e o Process Lasso versão 7.2.

4.3 Métricas

Com base nas questões mencionadas neste capítulo, foram definidas quatro métricas para avaliar o impacto que uma transmissão com criptografia causa no cliente. Essas métricas são explicadas a seguir.

CPU: A *Central Processing Unit* (CPU) é o cérebro do computador. É o elemento de maior importância dentro do computador, é responsável pelo processamento de todos os tipos de dados e pela apresentação do resultado do processamento, por tanto, se faz importante saber qual é o impacto nesse processamento.

Memória: A memória utilizada é uma métrica muito importante porque é nela que são carregados os programas que estão em execução, por tanto, a utilização dela influencia diretamente no desempenho de uma aplicação.

Delta: É basicamente a diferença entre a chegada do pacote atual e a chegada do pacote anterior. O delta é importante, pois tem relação com a taxa de pacotes recebidos, por exemplo, um tempo delta elevado significa que os pacotes foram recebidos a taxas baixas.

Handshake: O *handshake* é o processo pelo qual duas máquinas se reconhecem, e podem começar com a comunicação. É interessante comparar o tempo de *handshake* de cada um dos protocolos avaliados, já que o *handshake* influencia diretamente no tempo de inicio da transmissão dos dados.

Para a avaliação dos resultados referentes ao uso da CPU e memória, foi utilizado o programa Process Lasso, que permite visualizar a média de uso da CPU e a quantidade de memória utilizada por cada processo. Já para a análise do Delta e do *Handshake*, o programa utilizado foi o Wireshark, um programa que analisa o tráfego de rede gerado por dispositivos de comunicação. Por exemplo, o tráfego gerado pela placa de rede de um computador.

Para uma melhor análise das métricas definidas nesta seção foram realizados cálculos referentes ao desvio padrão e significância estatística. Os testes foram realizados para cada métrica. Também foi realizado um teste da significância estatística para todos os vídeos e todas as métricas, dessa forma, pode ser observado se o impacto é realmente relevante.

O teste de significância ou teste de hipótese, é um método de inferência estatística que utiliza dados de um estudo científico. Corresponde a uma regra decisória que permite rejeitar ou aceitar uma hipótese estatística com base nos resultados de uma amostra (PETERNELLI, 2004). Para isto, inicialmente foi necessário calcular o Z_c da seguinte forma:

$$Z_c = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

Onde $\bar{x}$ é a média, μ é a média esperada da população, s é o desvio padrão da amostra e n é o tamanho da amostra. Sendo que, para o cálculo do desvio padrão da amostra foi utilizada a fórmula do desvio padrão amostral da seguinte forma:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Onde $\bar{x}$ é a média, n é o tamanho da amostra e x_i é cada um dos n elementos. Em seguida foi consultada a tabela da curva normal para o Z_α correspondente. Para esse caso, onde foram utilizados dois desvios padrão, procuramos por 95%, o que corresponde a $Z_\alpha = 1{,}96$. Dessa forma, se $-Z_\alpha < Z_c < Z_\alpha$ podemos dizer que o resultado não é significativo (GIACOMELLI, 2014).

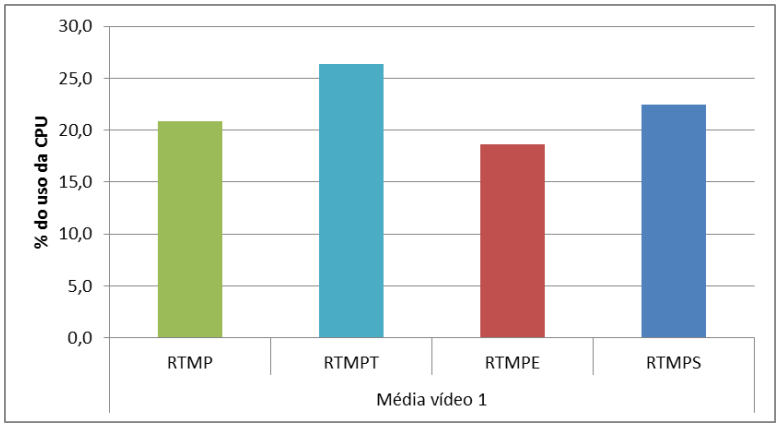
5 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos na avaliação das métricas descritas na Seção 4.3. Para a geração dos resultados, cada um dos três vídeos foi reproduzido cinco vezes para cada protocolo (RTMP, RTMPT, RTMPE, RTMPS), nesta ordem. Os resultados dos gráficos apresentados na Seção 5.1 correspondem à média para cada métrica.

5.1 Desempenho

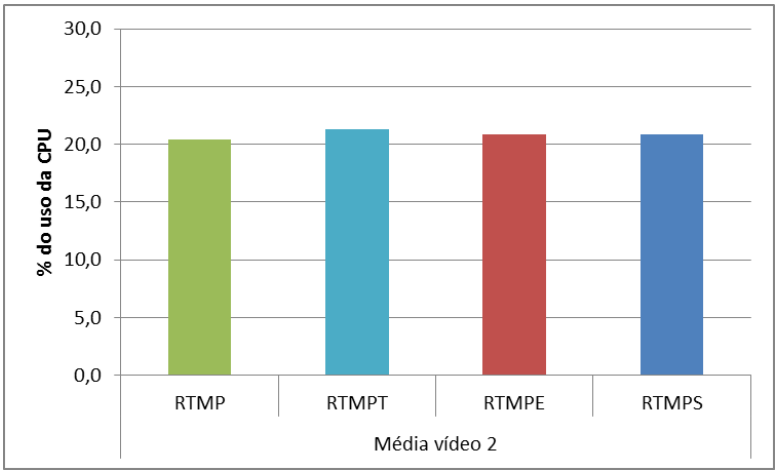
Primeiramente foi analisado o uso da CPU, a medida considerada foi a porcentagem do CPU que a aplicação utilizava reproduzindo cada vídeo pelo navegador. Como foi mencionado anteriormente, cada vídeo foi reproduzido cinco vezes em cada protocolo. As características dos vídeos como tamanho, duração do vídeo ou tempo do teste estão apresentadas na Tabela 4.1. Logo em seguida, são mostradas três figuras. A Figura 5.1 mostra os resultados da reprodução do vídeo 1, a Figura 5.2 mostra os resultados do vídeo 2 e a Figura 5.3 mostra os resultados do vídeo 3.

Figura 5.1 – Uso da CPU na reprodução do vídeo 1



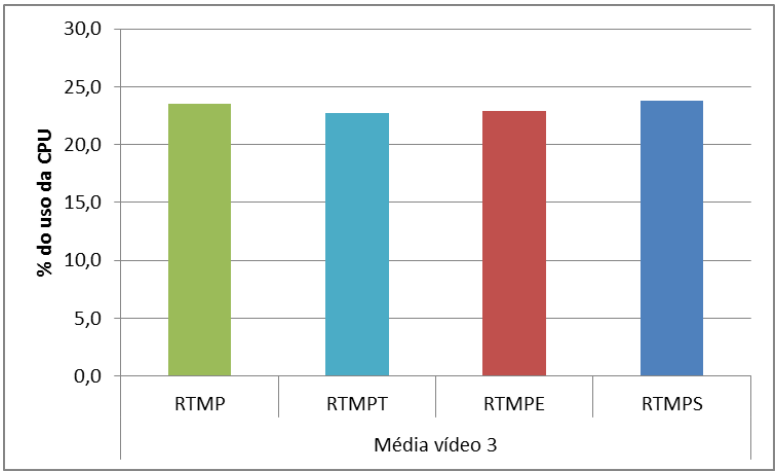
Fonte: Hércules, L. A. L. (2014).

Figura 5.2 – Uso da CPU na reprodução do vídeo 2



Fonte: Hércules, L. A. L. (2014).

Figura 5.3 – Uso da CPU na reprodução do vídeo 3



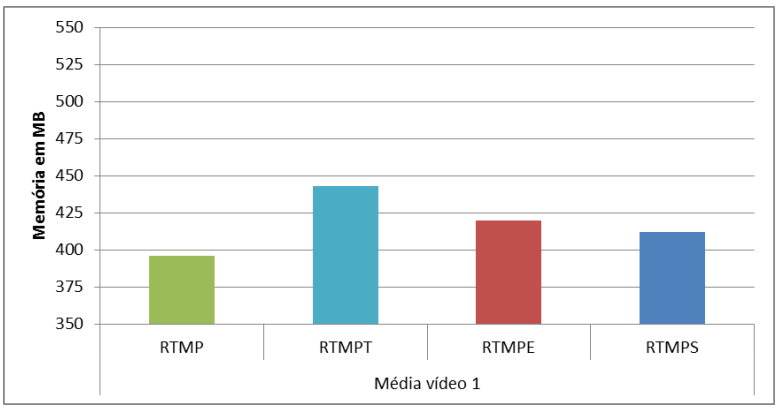
Fonte: Hércules, L. A. L. (2014).

As figuras acima apresentam no eixo X os protocolos nos qual os vídeos foram reproduzidos, e no eixo Y a porcentagem do uso do CPU. Percebe-se que o protocolo RTMPT foi o protocolo que mais utilizou CPU, quando comparado com os protocolos RTMPS e RTMPE, que são os protocolos que proporcionam segurança.

Depois de analisar o uso da CPU, foi analisado o uso de memória. Esse teste foi realizado da mesma maneira que o teste da CPU, executando cada vídeo individualmente no Google Chrome por cinco vezes. Os testes foram realizados cinco vezes para cada protocolo e posteriormente foi calculada a média. Nas figuras a seguir são apresentados os resultados obtidos nos três vídeos. A Figura 5.4 representa o uso de memória do vídeo 1, a Figura 5.5

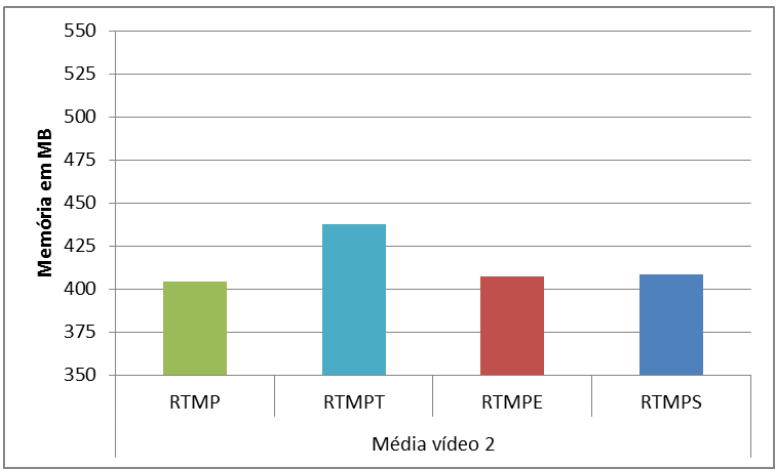
representa o resultado do uso de memória do vídeo 2 e a Figura 5.6 mostra o resultado para a execução do vídeo 3.

Figura 5.4 – Consumo de memória na reprodução do vídeo 1



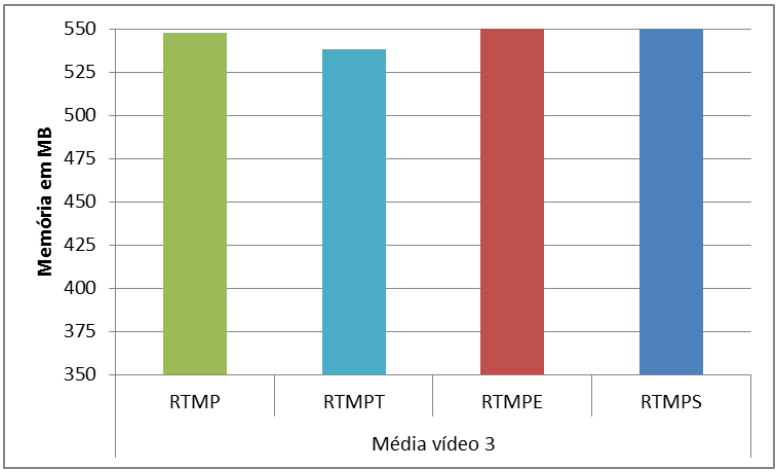
Fonte: Leiva (2004).

Figura 5.5 – Consumo de memória na reprodução do vídeo 2



Fonte: Leiva (2004).

Figura 5.6 – Consumo de memória na reprodução do vídeo 3

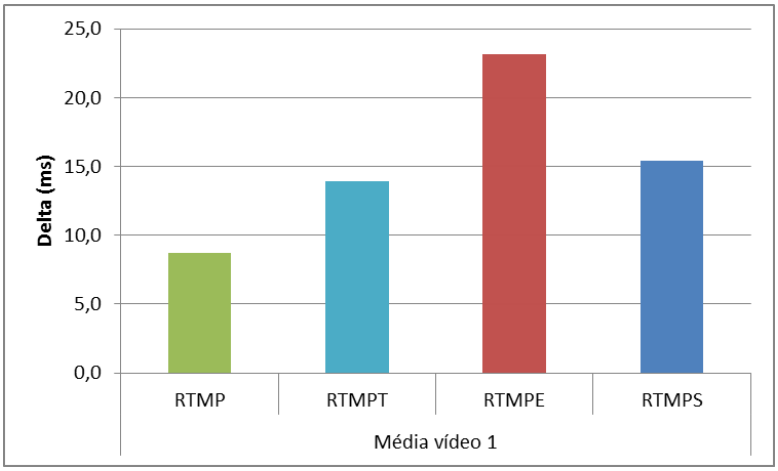


Fonte: Leiva (2004).

As três figuras acima apresentam no eixo X os protocolos e no eixo Y a memória que foi utilizada pela aplicação na reprodução de cada vídeo em Megabytes. Percebe-se que o protocolo RTMP consome menos memória quando comparado aos outros três protocolos. Também se pode observar o grande aumento no uso da memória para o vídeo 3. Apesar dos testes com o vídeo 3 serem realizados unicamente nos primeiros 20 segundos, o grande consumo de memoria é explicado pela qualidade do vídeo, já que ele é em alta definição e utiliza uma quantidade maior de dados por quadro, se comparado com os Standar Definition.

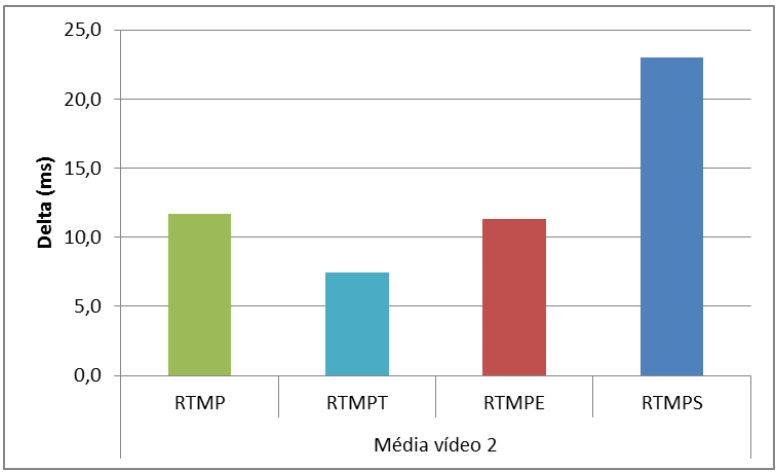
Para a análise do delta, foi utilizado o *wireshark* e foi realizado da mesma maneira que os testes anteriores, executando cada vídeo em cada protocolo cinco vezes. Foi calculada a média dos resultados que são mostrados a seguir. Na Figura 5.7 é apresentado o resultado para o vídeo 1, na Figura 5.8 é apresentado o resultado do vídeo 2 e na Figura 5.9 é mostrado o resultado para o vídeo 3.

Figura 5.7 – Delta na reprodução do vídeo 1



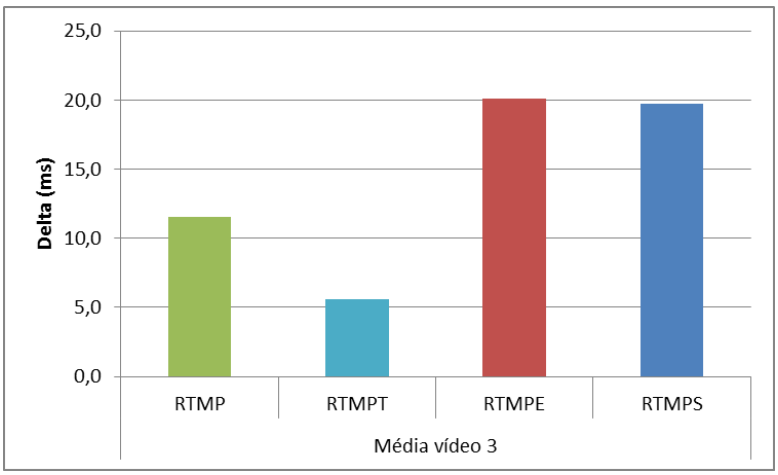
Fonte: Hércules, L. A. L. (2014).

Figura 5.8 – Delta na reprodução do vídeo 2



Fonte: Hércules, L. A. L. (2014).

Figura 5.9 – Delta na reprodução do vídeo 3

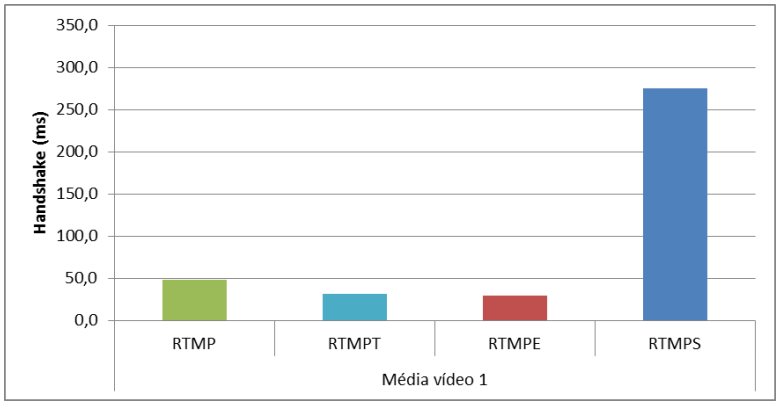


Fonte: Hércules, L. A. L. (2014).

O valor delta para cada vídeo está apresentado nas figuras acima e mostram no eixo X os quatro protocolos testados e no eixo Y o tempo delta, em milissegundos. Percebe-se que ocorreram diferenças para cada vídeo, mas quando a média é calculada, os resultados dos protocolos RTMP e RTMPT se mostram muito próximos. Já os protocolos RTMPE e RTMPS se mostram próximos entre eles, porém distantes do RTMP e do RTMPT.

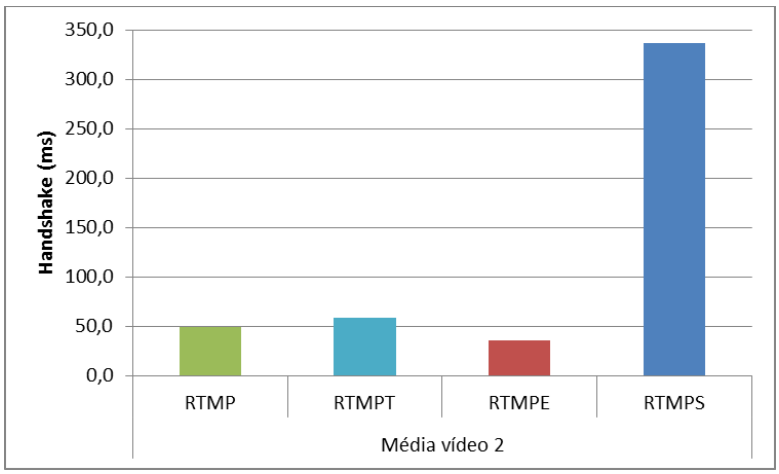
Por fim, foi avaliado o tempo de cada protocolo para executar o *handshake*. A execução dos testes para o *handshake* segue o mesmo formato dos testes anteriores. Os resultados são mostrados nas figuras a seguir. A Figura 5.10 mostra o resultado do handshake do vídeo 1, a Figura 5.11 mostra o resultado do vídeo 2 e na Figura 5.12 é apresentado o resultado do teste no vídeo 3.

Figura 5.10 – Handshake do vídeo 1



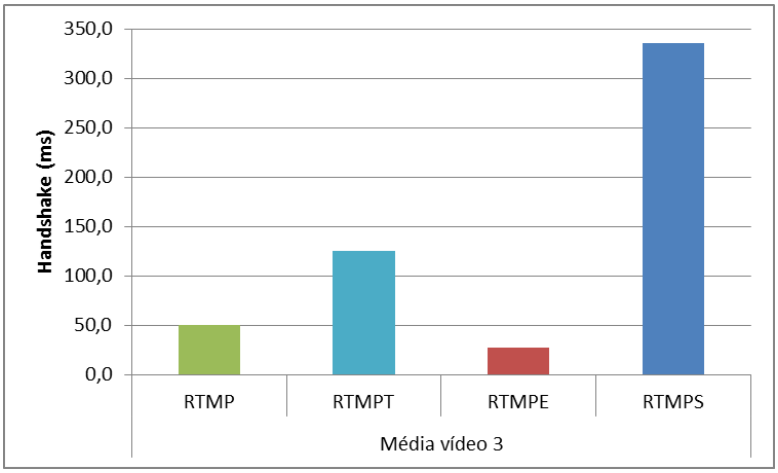
Fonte: Hércules, L. A. L. (2014).

Figura 5.11 – Handshake do vídeo 2



Fonte: Hércules, L. A. L. (2014).

Figura 5.12 – Handshake do vídeo 3



Fonte: Hércules, L. A. L. (2014).

Nas figuras que representam os resultados do *handshake*, o eixo X representa os quatro protocolos testados e o eixo Y o tempo que o *handshake* demorou, em milissegundos. Percebe-se um amplo aumento no processo de *handshake* no protocolo RTMPS, e isto se deve ao número de mensagens trocadas nesse processo pelo protocolo TLS/SSL. Na seguinte seção serão discutidos com mais detalhes os resultados obtidos.

5.2 Discussão dos resultados

Na análise dos resultados do uso do CPU, foi possível perceber que os quatro protocolos utilizam praticamente a mesma quantidade dessa métrica, existindo apenas, uma leve diferença entre o RTMPT e os outros protocolos.

Tabela 5.1 – Significância estatística dos resultados do uso de CPU

	RTMPT	RTMPE	RTMPS
Média	23,46 %	20,79 %	22,37 %
Desvio Padrão	2,37	1,88	3,42
Tamanho N	15	15	15
Média RTMP	21,59 %	21,59 %	21,59 %
Z_c	3,06	-1,65	0,88
Significância	significativo	nula	nula

Fonte: Hércules, L. A. L. (2014).

A Tabela 5.1 mostra a significância do uso do CPU. Em dois casos, pode-se perceber que a significância é nula. Isto quer dizer que, o processo de criptografia próprio da Adobe no RTMPE e de criptografia por certificado SSL no RTMPS não causa nenhum impacto considerável no uso da CPU. Já no caso do RTMPT, vemos que a significância não é nula, portanto o processo de tunelamento do RTMPT é significativamente mais custoso para o uso do CPU. Significa afirmar que o esforço da CPU para processar os protocolos RTMP, RTMPE e RTMPS é basicamente o mesmo.

Com relação ao uso da memória, foi visualizado um amplo aumento para todos os protocolos quando foi reproduzido o vídeo 3, porém, o aumento se deu nos quatro protocolos. Tal aumento pode ser explicado pela qualidade HD do vídeo 3, com o qual é necessária uma quantidade muito maior de dados para o vídeo ser exibido. Mas, realizando a comparação entre os protocolos, e não entre os vídeos, percebe-se que no total existe um aumento nas três variações do protocolo RTMP, entretanto, o teste de significância estatística mostrou que essa variação pode ser desconsiderada. Na Tabela 5.2 são apresentados os resultados da significância para a utilização de memória.

Tabela 5.2 – Significância estatística dos resultados do uso da memória

	RTMPT	RTMPE	RTMPS
Média	472,98 MB	459,36 MB	456,99 MB
Desvio Padrão	48,48	71,73	82,60
Tamanho N	15	15	15
Média RTMP	449,25 MB	449,25 MB	449,25 MB
Z_c	1,90	0,55	0,36
Significância	nula	nula	nula

Fonte: Hércules, L. A. L. (2014).

Para os valores encontrados para o delta foi observada uma variação entre os vídeos, mas em geral pode-se observar que houve um acréscimo nos dois protocolos que oferecem segurança para o RTMP. No teste de significância para o delta, os resultados apontaram para uma diferença significativa de valores para os protocolos RTMPE e RTMPS quando comparados com o RTMP. A Tabela 5.3 nos mostra os cálculos para a significância estatística do delta.

Tabela 5.3 – Significância estatística dos resultados do delta

	RTMPT	RTMPE	RTMPS
Média	9,01 ms	18,20 ms	19,40 ms
Desvio Padrão	5,01	7,56	7,35
Tamanho N	15	15	15
Média RTMP	10,66 ms	10,66 ms	10,66 ms
Z_c	-1,28	3,86	4,61
Significância	nula	significativo	significativo

Fonte: Hércules, L. A. L. (2014).

Por fim, os resultados da análise do tempo do *handshake* mostraram uma grande variação entre os protocolos RTMP, RTMPT e RTMPE comparados com o RTMPS. Aparentemente, os protocolos RTMP, RTMPT e RTMPE consomem o mesmo tempo para realizar o *handshake*, porém, depois de realizadas as análises da significância estatística foi possível concluir que: Tanto RTMPS, como o RTMPT e RTMPE, tem uma variação de tempo considerável para realizar o *handshake*. Na Tabela 5.4 é mostrado o calculo da significância estatística.

Tabela 5.4 – Significância estatística dos resultados do tempo de handshake

	RTMPT	RTMPE	RTMPS
Média	71,94 ms	30,76 ms	315,59 ms
Desvio Padrão	43,37	7,39	60,59
Tamanho N	15	15	15
Média RTMP	48,78 ms	48,78 ms	48,78 ms
Z_c	2,07	-9,45	17,06
Significância	significativo	significativo	significativo

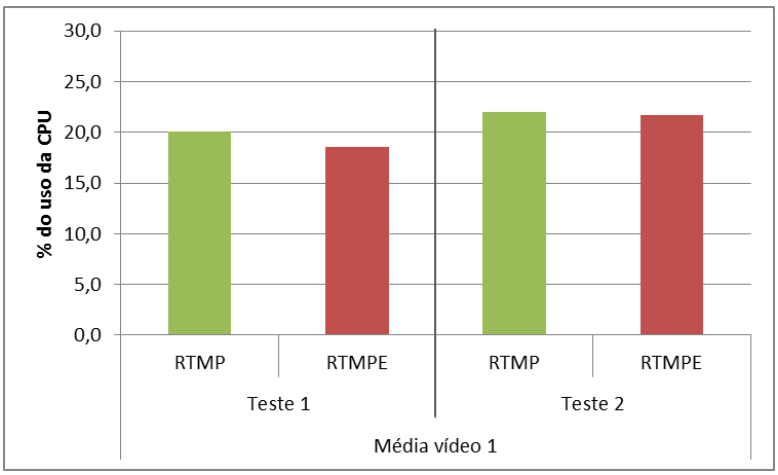
Fonte: Hércules, L. A. L. (2014).

A tabela acima mostra que, para o uso dos protocolos RTMPT, RTMPE e RTMPS, o tempo de *handshake* deve ser levado em consideração. A grande diferença entre o RTMPS e os outros protocolos, é devida ao processo de *handshake* do SSL necessitar transmitir mais mensagens para que se possa iniciar a transmissão de dados. Esse processo é explicado na Figura 2.7 e Tabela 2.2.

Depois de realizar a análise surgiu uma incerteza nos resultados. Pois o RTMPE utilizava menos CPU que o RTMP, porém, o resultado esperado era que o uso da CPU do RTMPE fosse maior que o uso da CPU do RTMP. Para corroborar o resultado obtido anteriormente foi realizada uma nova análise. Para tanto, foi utilizado um novo cliente, um

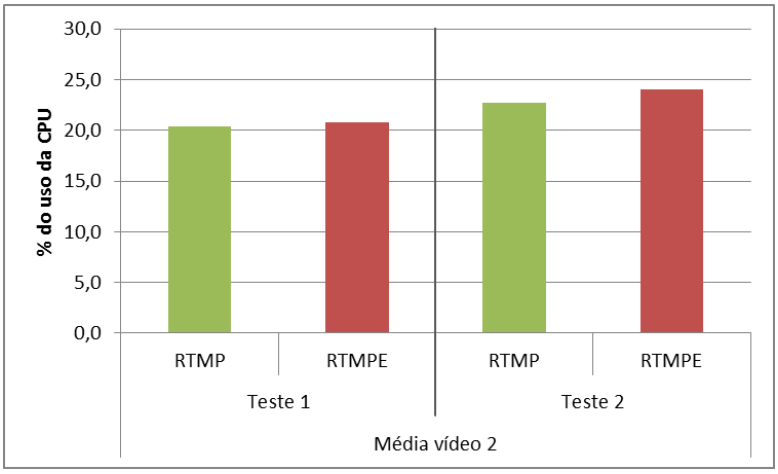
computador Spacebr com processador AMD Athlon™ X4 630, CPU com frequência 2,8 GHz, 4 núcleos físicos, 4 núcleos lógicos, memória RAM de 4GB, dos quais 3,25 GB são utilizáveis. Os novos testes foram realizados no navegador Google Chrome Version 39.0.2171.71 m, e foram realizados unicamente para o uso do CPU para os protocolos RTMP e RTMPE. Os resultados são apresentados nas três figuras a seguir. A Figura 5.13 mostra os resultados da reprodução do vídeo 1 referente ao primeiro teste realizado (Teste 1) e do novo teste realizado (Teste 2). A Figura 5.14 apresenta os resultados do vídeo 2 da mesma forma, mostrando o resultado referente ao primeiro teste (Teste 1) e o resultado novo (Teste 2). E a Figura 5.15 mostra os resultados do vídeo 3.

Figura 5.13 – Comparação entre teste 1 e teste 2 para o vídeo 1



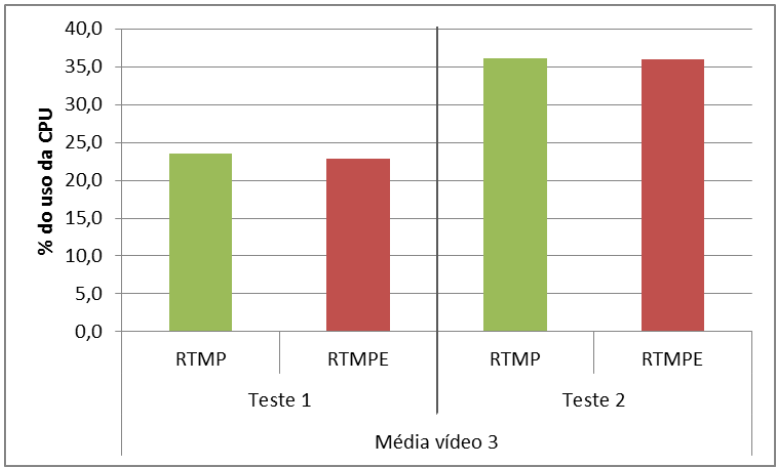
Fonte: Hércules, L. A. L. (2014).

Figura 5.14 – Comparação entre teste 1 e teste 2 para o vídeo 2



Fonte: Hércules, L. A. L. (2014).

Figura 5.15 – Comparação entre teste 1 e teste 2 para o vídeo 3



Fonte: Hércules, L. A. L. (2014).

Também foi realizado o teste de significância para o teste 2, e os resultados apresentam equivalência com os resultados de significância do teste 1. Sendo assim, não houve significância estatística na utilização do uso de CPU para os protocolos RTMP e RTMPE.

6 CONCLUSÃO

De posse dos resultados das análises teórica e prática sobre os protocolos RTP e RTMP, pode-se ter um melhor entendimento do processo de videoconferência e do funcionamento dos seus principais protocolos. Também foi possível observar qual o impacto no desempenho quando são acrescentados alguns mecanismos de segurança.

De acordo com os resultados obtidos neste trabalhado, e também no trabalho relacionado já discutido, nota-se que o acréscimo de segurança impõe custos ao processamento. Os protocolos SRTP e RTMPE apresentaram um desempenho desejável, em tempo, processamento, memória e taxa de dados quando comparados com o RTP e RTMP respectivamente. Já o RTMPS apresentou resultados satisfatórios em algumas características como o uso do CPU e uso de memória, porém na taxa de transmissão e no handshake não se encontrou um desempenho tão bom. Esses achados não tão aceitáveis podem influenciar na escolha pela implementação da segurança em videoconferência.

Durante o processo de execução deste trabalho foi possível perceber que existem várias soluções de segurança gratuitas e também pagas, ambas, com a mesma funcionalmente, finalidade e com o mesmo poder de segurança. Portanto, a escolha pela melhor solução dependerá do esforço que se está disposto a realizar para garantir a segurança da comunicação e também do tipo de aplicação.

O impacto no desempenho pode ser desconsiderado no protocolo SRTP, isto se deve ao fato de que o SRTP é específico para fornecer segurança ao RTP. Por esta razão, o desempenho encontrado foi satisfatório. Porém, para casos onde não existe especificidade entre os protocolos, como o RTMPS que utiliza certificados SSL, os resultados de desempenho podem não ser satisfatórios, mas ainda sim são aceitáveis e vantajosos.

Vale salientar que ainda são necessários mais estudos que avaliem não só as métricas estudadas neste trabalho, mas também medidas como uso da CPU no servidor, quantidade de conexões simultâneas e *Round Trip Time* (RTT) para confirmar os resultados obtidos no presente estudo. Mas o que se pode concluir é que a utilização de segurança em transmissões em tempo real se mostrou uma saída pertinente para quem coloca a segurança como prioridade.

REFERÊNCIAS

ALEXANDER, L.; WIJESINHA, A.; KARNE, R. **An Evaluation of Secure Real-time Transport Protocol (SRTP) Performance for VoIP**. Towson, MD, USA. Third International Conference on Network and System Security, 2009.

ANDREASEN F, BAUGHER M, WING D. **Session Description Protocol (SDP) Security Descriptions for Media Streams**. IETF RFC 4568, 2006.

BAUGHER, M.; MCGREW, D.; NASLUND, M.; CARRARA, E.; NORRMAN, K. **The Secure Real-time Transport Protocol (SRTP)**. IETF RFC 3711, 2004.

BEGEN, A.; FRIEDRICH, E. **Multicast Acquisition Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)**. IETF RFC 6332. 2011.

DIERKS, T.; RESCORLA, E. **The Transport Layer Security (TLS) Protocol Version 1.2**. IETF RFC 4568, 2008.

EASTLAKE, D.; CROCKER, S.; SCHILLER, J. **Randomness Recommendations for Security**. IETF RFC 1750, 1994.

FIRESTONE, S.; RAMALINGAM, T.; FRY, S. **Voice and Video Conferencing Fundamentals**. Cisco Systems. Security Design in Conferencing. USA: Cisco Press, 2007. p. 223-327.

FREIER, A.; KARLTON, P.; KOCHER, P. **The Secure Sockets Layer (SSL) Protocol Version 3.0**. IETF RFC 6101, 2011. p. 1-40.

GIACOMELLI, M. A. **Probabilidade e Estatística**. Instituto de Matemática, Departamento de Estatística. UFRGS, 2014. p. 93-107.

GORALSKI W. **The Illustrated Network**. USA: Morgan Kaufmann Publisher, 2009, p-558-604.

HELP ADOBE. **Developing FLASH LITE 4 Applications**. Adobe Systems Incorporated and its licensors, 2011. p. 43-49.

KOHNO, T.; PALACIO, A.; BLACK, J. **Building Secure Cryptographic Transforms, or How to Encrypt and MAC**. Agosto, 2003. p. 1-8.

KRAWCZYK, H.; BELLARE, M.; CANETTI. R. **HMAC: Keyed-Hashing for Message Authentication**. IETF RFC 3711, 1997.

KUROSE, J. F.; ROSS, K. W. **Computer Networking: A top-down approach**. 6th. ed. USA: Addison-Wesley Publishing Company, 2012.

LOSHIN P. **Simple Steps to Data Encryption**. 1th. ed. USA: Syngress, 2013.

MARK R. **Symmetric-key cryptography**. Disponível em <http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/symmetric-key.html>. Acesso em: 01 dez. 2014.

PARMAR, H.; THORNBURGH, M. **Adobe´s Real Time Messaging Protocol**. Adobe Systems Incorporated, 2012.

PETERNELLI, L. A. **Testes de significância**. INF 162 Prof. Luiz A. Peternelli Capítulo 6, 2004.

PUANGPRONPITAG S, KASABAI P, PANSA D. **An Enhancement of the SDP Security Description (SDES) for Key Protection**. Faculty of Informatics, Mahasarakham University. Mahasarakham, Thailand, 2012

ROCHA R. **Detecção em Tempo-Real de Ataques de Negação de Serviços na Rede de Origem Usando um Classificador Bayesiano Simples**. 2012. p. 1-17.

ROSENBERG, J.; SCHULZRINNE, H.; CAMARILLO, G.; JOHNSTON, A.; PETERSON, J.; SPARKS, R.; HANDLEY, M.; SCHOOLER, E. **SIP: Session Initiation Protocol**. IETF RFC 3261. 2002. p. 1-20.

SHOPPER™. **SSL Shopper**. Disponível em: <https://www.sslshopper.com>. Acesso em: 01 dez. 2014.

SYSTEMS CISCO. **Cisco TelePresence Secure Comunications and Signaling**. Cisco TelePresence Security Protocol Details. Cisco, 2009.

THOMAS S. A. **SSL & TLS Essentials**. USA: John Wiley & Sons, Inc, 2000.

TOWERS, K.; GREEN, T. **Video content protection measures enabled by Adobe Flash Media Interactive Server 3.5**. Adobe Systems Incorporated. Enable basic video protection with Flash Media Interactive Server, 2010.

WARD, M. **Secure SIP protects VoIP traffic**. Disponível em <http://www.networkworld.com/article/2311252/tech-primers/secure-sip-protects-voip-traffic.html>. Acesso em: 01 dez. 2014.

ANEXO A: TABELAS REFERENTES AOS RESULTADOS DOS TESTES

Tabela 1 - Uso da CPU na reprodução do vídeo 1 (Teste 1)

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 1	15,93	24,28	18,94	23,06
	22,15	26,14	19,32	21,71
	21,80	26,48	18,13	25,90
	23,20	27,19	18,07	15,92
	21,12	27,75	18,69	25,85
Média	20,84	26,37	18,63	22,49

Tabela 2 - Uso da CPU na reprodução do vídeo 2 (Teste 1)

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 2	22,03	22,15	21,03	20,54
	20,77	21,60	20,95	23,86
	18,44	19,85	21,01	23,85
	20,71	21,79	20,22	20,40
	20,29	21,33	21,00	15,43
Média	20,45	21,34	20,84	20,82

Tabela 3 - Uso da CPU na reprodução do vídeo 3 (Teste 1)

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 3	23,18	22,77	23,66	23,37
	23,90	21,90	22,06	21,32
	23,08	22,79	22,76	28,15
	23,59	23,01	22,31	22,91
	23,75	23,00	23,67	23,26
Média	23,50	22,69	22,89	23,80

Tabela 4 - Consumo de memória na reprodução do vídeo 1

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 1	336,43	441,56	400,44	354,70
	407,80	443,39	419,46	414,69
	412,84	441,90	423,19	446,87
	420,38	445,34	432,07	441,12
	401,83	443,88	422,50	402,94
Média	395,86	443,21	419,53	412,07

Tabela 5 - Consumo de memória na reprodução do vídeo 2

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 2	421,41	430,51	328,34	422,99
	430,86	441,75	431,44	436,77
	304,76	443,13	431,93	300,94
	429,40	443,69	421,20	444,47
	433,48	428,24	423,96	436,17
Média	403,98	437,47	407,37	408,27

Tabela 6 - Consumo de memória na reprodução do vídeo 3

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 3	549,24	550,44	549,35	521,47
	546,80	540,52	553,82	609,40
	547,22	549,65	550,68	585,61
	548,54	524,76	551,04	550,10
	547,81	525,96	550,94	486,55
Média	547,92	538,27	551,17	550,63

Tabela 7 - Delta na reprodução do vídeo 1

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 1	9,405	12,131	28,575	4,747
	9,453	10,089	18,186	20,047
	9,199	9,344	29,632	9,507
	6,140	19,365	21,814	23,808
	9,508	18,900	17,576	19,012
Média	8,741	13,966	23,157	15,424

Tabela 8 - Delta na reprodução do vídeo 2

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 2	11,746	10,479	10,932	28,583
	11,519	9,305	11,918	22,542
	12,766	8,031	11,619	30,556
	10,969	3,378	11,288	16,257
	11,374	6,116	10,997	17,187
Média	11,675	7,462	11,351	23,025

Tabela 9 - Delta na reprodução do vídeo 3

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 3	11,346	10,123	20,381	29,551
	11,292	5,170	12,827	11,716
	11,437	4,260	33,624	21,970
	11,926	2,280	21,087	14,962
	11,841	6,178	12,482	20,603
Média	11,568	5,602	20,080	19,760

Tabela 10 - Handshake do vídeo 1

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 1	53,413	21,000	41,266	331,260
	44,645	23,562	27,114	333,406
	67,996	69,123	32,594	341,178
	47,057	23,784	23,445	258,118
	25,068	20,367	23,140	110,530
Média	47,636	31,567	29,512	274,898

Tabela 11 - Handshake do vídeo 2

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 2	49,453	52,977	32,630	342,296
	45,005	77,925	42,373	336,417
	48,900	59,322	31,890	323,207
	51,124	50,310	38,650	351,669
	48,780	52,373	33,188	328,632
Média	48,652	58,581	35,746	336,444

Tabela 12 - Handshake do vídeo 3

	RTMP	RTMPT	RTMPE	RTMPS
Vídeo 3	53,835	118,946	33,205	332,989
	51,557	130,369	27,380	338,317
	49,824	144,499	34,663	328,358
	47,785	114,495	24,895	345,338
	47,269	119,984	14,937	332,107
Média	50,054	125,659	27,016	335,422

Tabela 13 - Uso da CPU na reprodução do vídeo 1 (Teste 2)

	RTMP	RTMPE
Vídeo 1	20,48	22,00
	22,84	22,23
	22,54	21,54
	22,49	21,53
	21,69	21,21
Média	22,00	21,70

Tabela 14 - Uso da CPU na reprodução do vídeo 2 (Teste 2)

	RTMP	RTMPE
Vídeo 2	24,03	23,62
	24,58	23,07
	23,59	24,50
	16,05	24,82
	25,22	24,39
Média	22,69	24,08

Tabela 15 - Uso da CPU na reprodução do vídeo 3 (Teste 2)

	RTMP	RTMPE
Vídeo 3	36,99	36,62
	35,94	35,52
	36,75	35,96
	35,51	35,67
	35,33	35,94
Média	36,10	35,94

ANEXO B: TABELA REFERENTE À CURVA NORMAL Z

Área sob a curva normal padronizada compreendida entre os valores 0 e Z

Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.0000	0.0040	0.0080	0.0120	0.0160	0.0199	0.0239	0.0279	0.0319	0.0359
0.1	0.0398	0.0438	0.0478	0.0517	0.0557	0.0596	0.0636	0.0675	0.0714	0.0753
0.2	0.0793	0.0832	0.0871	0.0910	0.0948	0.0987	0.1026	0.1064	0.1103	0.1141
0.3	0.1179	0.1217	0.1255	0.1293	0.1331	0.1368	0.1406	0.1443	0.1480	0.1517
0.4	0.1554	0.1591	0.1628	0.1664	0.1700	0.1736	0.1772	0.1808	0.1844	0.1879
0.5	0.1915	0.1950	0.1985	0.2019	0.2054	0.2088	0.2123	0.2157	0.2190	0.2224
0.6	0.2257	0.2291	0.2324	0.2357	0.2389	0.2422	0.2454	0.2486	0.2517	0.2549
0.7	0.2580	0.2611	0.2642	0.2673	0.2704	0.2734	0.2764	0.2794	0.2823	0.2852
0.8	0.2881	0.2910	0.2939	0.2967	0.2995	0.3023	0.3051	0.3078	0.3106	0.3133
0.9	0.3159	0.3186	0.3212	0.3238	0.3264	0.3289	0.3315	0.3340	0.3365	0.3389
1.0	0.3413	0.3438	0.3461	0.3485	0.3508	0.3531	0.3554	0.3577	0.3599	0.3621
1.1	0.3643	0.3665	0.3686	0.3708	0.3729	0.3749	0.3770	0.3790	0.3810	0.3830
1.2	0.3849	0.3869	0.3888	0.3907	0.3925	0.3944	0.3962	0.3980	0.3997	0.4015
1.3	0.4032	0.4049	0.4066	0.4082	0.4099	0.4115	0.4131	0.4147	0.4162	0.4177
1.4	0.4192	0.4207	0.4222	0.4236	0.4251	0.4265	0.4279	0.4292	0.4306	0.4319
1.5	0.4332	0.4345	0.4357	0.4370	0.4382	0.4394	0.4406	0.4418	0.4429	0.4441
1.6	0.4452	0.4463	0.4474	0.4484	0.4495	0.4505	0.4515	0.4525	0.4535	0.4545
1.7	0.4554	0.4564	0.4573	0.4582	0.4591	0.4599	0.4608	0.4616	0.4625	0.4633
1.8	0.4641	0.4649	0.4656	0.4664	0.4671	0.4678	0.4686	0.4693	0.4699	0.4706
1.9	0.4713	0.4719	0.4726	0.4732	0.4738	0.4744	0.4750	0.4756	0.4761	0.4767
2.0	0.4772	0.4778	0.4783	0.4788	0.4793	0.4798	0.4803	0.4808	0.4812	0.4817
2.1	0.4821	0.4826	0.4830	0.4834	0.4838	0.4842	0.4846	0.4850	0.4854	0.4857
2.2	0.4861	0.4864	0.4868	0.4871	0.4875	0.4878	0.4881	0.4884	0.4887	0.4890
2.3	0.4893	0.4896	0.4898	0.4901	0.4904	0.4906	0.4909	0.4911	0.4913	0.4916
2.4	0.4918	0.4920	0.4922	0.4925	0.4927	0.4929	0.4931	0.4932	0.4934	0.4936
2.5	0.4938	0.4940	0.4941	0.4943	0.4945	0.4946	0.4948	0.4949	0.4951	0.4952
2.6	0.4953	0.4955	0.4956	0.4957	0.4959	0.4960	0.4961	0.4962	0.4963	0.4964
2.7	0.4965	0.4966	0.4967	0.4968	0.4969	0.4970	0.4971	0.4972	0.4973	0.4974
2.8	0.4974	0.4975	0.4976	0.4977	0.4977	0.4978	0.4979	0.4979	0.4980	0.4981
2.9	0.4981	0.4982	0.4982	0.4983	0.4984	0.4984	0.4985	0.4985	0.4986	0.4986
3.0	0.4987	0.4987	0.4987	0.4988	0.4988	0.4989	0.4989	0.4989	0.4990	0.4990
3.1	0.4990	0.4991	0.4991	0.4991	0.4992	0.4992	0.4992	0.4992	0.4993	0.4993
3.2	0.4993	0.4993	0.4994	0.4994	0.4994	0.4994	0.4994	0.4995	0.4995	0.4995
3.3	0.4995	0.4995	0.4995	0.4996	0.4996	0.4996	0.4996	0.4996	0.4996	0.4997
3.4	0.4997	0.4997	0.4997	0.4997	0.4997	0.4997	0.4997	0.4997	0.4997	0.4998
3.5	0.4998	0.4998	0.4998	0.4998	0.4998	0.4998	0.4998	0.4998	0.4998	0.4998
3.6	0.4998	0.4998	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999
3.7	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999
3.8	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999	0.4999
3.9	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000

ANEXO C: TRABALHO DE GRADUAÇÃO 1

Impacto no Desempenho em Aplicações de Tempo Real Utilizando Criptografia

Luis A. L. Hercules[1], Valter Roesler[1], Raul F. Weber[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brasil

{lalhercules,roesler,weber}@inf.ufrgs.br

Abstract. *Currently, security is a major concern for the development of applications in computer science, increasing its importance when it involves sensitive or relevant information to the user. In data transmission in real time such as voice and video streaming applications, a good performance increase safety becomes paramount. This article is a study of the Secure Real-time Transport Protocol (SRTP) which provides security to the Real-time Transport Protocol (RTP), and a security mechanism to Adobe's Real Time Messaging Protocol (RTMP) as these will be required to implement security in a real-time application. This study will serve as a theoretical foundation for the realization of Graduate Work 2, in which is discussed the impact of the performance in real time applications.*

Resumo. *Atualmente, segurança é uma das principais preocupações para o desenvolvimento de aplicações na área de informática, elevando sua importância quando envolve informações sigilosas ou relevantes para o usuário. Em aplicações de transmissão de dados em tempo real como transmissão de voz e transmissão de vídeo, um bom desempenho do acréscimo da segurança se torna fundamental. O presente artigo é um estudo sobre o Secure Real-time Transport protocol (SRTP) que fornece segurança ao Real-time Transport Protocol (RTP), e um mecanismo de segurança para o Adobe´s Real Time Messaging Protocol (RTMP) pois estes serão necessários para a implementação de segurança em uma aplicação de tempo real. Este estudo servirá de embasamento teórico para a realização do Trabalho de Graduação 2, no qual será discutido o impacto do desempenho em aplicações de tempo real.*

1. Introdução

Dentre as características desejadas em qualquer tipo de transmissão de dados encontram-se a integridade e a confidencialidade dessas informações. Dependendo do contexto em que essa transmissão se encontra, o impacto negativo no desempenho se torna um problema para o uso dessas aplicações, com isso, se faz necessário encontrar mecanismos que consigam englobar segurança e desempenho favorável. Em aplicações de transmissão de dados em tempo real, como transmissão de voz e transmissão de vídeo, o impacto no desempenho é fundamental.

Aplicações de transmissão em tempo real são cada vez mais utilizadas em redes e ambientes públicos, dessa forma, estão cada vez mais vulneráveis a quaisquer tipos de

ataque. Portanto, medidas de segurança devem ser aplicadas para proteger as informações contidas nesse tipo de transmissão que trafega por essas redes. Soluções em criptografia conferem a esse tipo de transmissão a privacidade desejada.

O *Real-time Transport Protocol* (RTP) define o formato padrão do pacote para transmitir áudio e vídeo através de uma rede IP [Baugher et al. 2004]. O RTP é altamente utilizado em sistemas de comunicação e entretenimento que envolvem streaming de mídia, como telefonia, aplicações em videoconferência e serviços de televisão. O RTP é o encarregado de carregar a mídia em si, enquanto que em conjunto, o *Real-time Transport Control Protocol* (RTCP) fornece as informações de controle e estatísticas do fluxo RTP fora da banda [Begen et al. 2011]. Para tanto, alguns protocolos utilizam o *Session Initiation Protocol* (SIP) para dar inicio à transmissão, entre esses está incluso o RTP [Rosenberg et al. 2002].

O *Secure Real-time Transport Protocol* (SRTP) define o perfil do RTP que é destinado a fornecer criptografia, integridade, autenticação de mensagens e proteção a ataques de replay ao trafego de dados RTP e também ao trafego de controle para o RTP, o RTCP, tanto em aplicações unicast como em aplicações multicast [Baugher et al. 2004].

O presente estudo abordará brevemente o *Adobe's Real Time Messaging Protocol* (RTMP), elucidando como são aplicados mecanismos de segurança nesse protocolo. Os motivos pelos quais o RTMP foi escolhido dentre tantos outros existentes encontram-se na sua compatibilidade com o *Flash Player* e na futura implementação de segurança que será apresentada no Trabalho de Graduação 2 (TG2).

Existem diversos tipos de abordagem para implementar uma aplicação de videoconferência, os quais envolveriam diferentes métodos de comunicação e protocolos. O estudo teórico neste artigo abordará principalmente os protocolos citados anteriormente, e alguns outros que estão envolvidos na transmissão de vídeo e áudio dentro do Mconf. O Mconf é um sistema de multiconferência open source, o qual tem apoio da Universidade Federal do Rio Grande do Sul (UFRGS) e da Rede Nacional de Ensino e Pesquisa (RNP) [Mconf 2014]. Atualmente, o Mconf não conta com criptografia ou métodos de segurança, ou seja, se ocorrer interceptação da comunicação, as informações contidas poderão ser capturadas com facilidade. Tendo em vista os motivos relatados acima, o TG2 terá como um dos focos a implementação de segurança no Mconf, avaliando também o impacto que esse acréscimo de segurança causará no desempenho desse sistema.

Portanto, o objetivo do presente estudo é explicar os principais mecanismos de criptografia, expondo as principais características do funcionamento dos protocolos envolvidos em sistemas de multiconferência, e também mostrar como a criptografia pode ser aplicada nestes protocolos. O TG2 pretende investigar, implementar e também avaliar o impacto no processamento, tempo extra, memória utilizada e a largura de banda, que este tipo de segurança acrescenta em uma aplicação de tempo real.

O restante do artigo está organizado da seguinte maneira, na Seção 2 será apresentada uma explicação sobre criptografia e os principais protocolos envolvidos em sistemas de multiconferência. Na Seção 3 será mostrada a proposta para o TG2, bem como sua organização e a metodologia que será adotada, juntamente com o cronograma

de atividades a serem realizadas. E por fim, a Seção 4 é reservada para as conclusões parciais do estudo.

2. Conceitos Básicos

Antes de se fazer qualquer análise acerca da criptografia em videoconferência é necessário que se tenha uma compreensão básica sobre a criptografia.

2.1. Criptografia

A criptografia de dados consiste em permitir que um emissor e um receptor possam garantir a confidencialidade de dados. Para isso, existem dois tipos de criptografia, criptografia simétrica (algoritmo de chave única) e criptografia assimétrica (criptografia de chave pública).

A criptografia simétrica consiste em: o emissor cifrar o texto com uma chave secreta, pré compartilhada e transmitir o texto cifrado. Quando o receptor recebe o texto cifrado, o texto é descifrado junto com a chave secreta [Firestone et al. 2007]. Desta forma, se durante a transmissão os dados transmitidos forem interceptados por terceiros, não poderão ser interpretados. A Figura 1 mostra este procedimento.

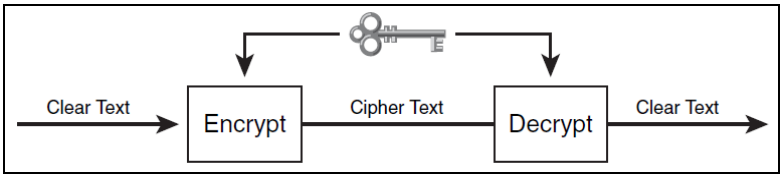


Figura 1. Criptografia simétrica

Ao contrário do que ocorre com a criptografia simétrica, em que o remetente e o destinatário usam a mesma chave. A criptografia de chave pública (assimétrica) utiliza duas chaves. Nesta abordagem, cada terminal cria uma chave pública e uma chave privada. Cada terminal guarda a chave privada, mas faz com que a chave pública seja disponível [Firestone et al. 2007].

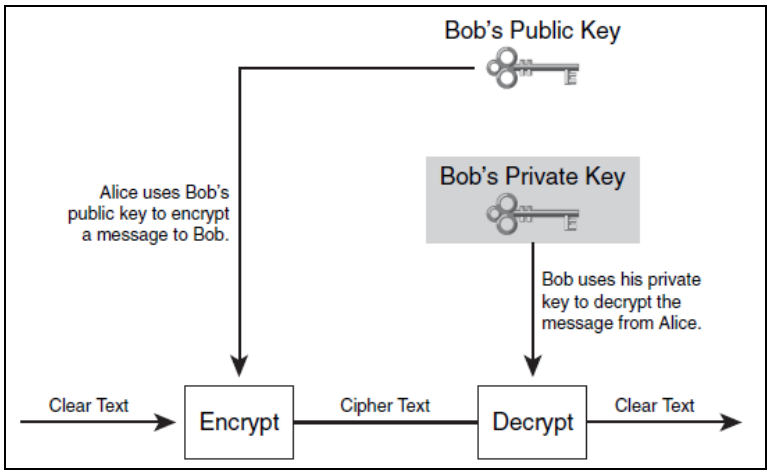


Figura 2. Criptografia assimétrica

A criptografia assimétrica consiste em: o emissor cifrar o texto simples com sua chave privada, e logo após transmite o texto cifrado. Quando o receptor recebe o texto

cifrado, ele é descifrado com a chave pública do emissor [Firestone et al. 2007]. A Figura 2 ilustra este procedimento.

2.2. SRTP

O *Secure Real-time Transport Protocol* (SRTP) define um perfil do *Real-time Transport Protocol* (RTP), o qual fornece confidencialidade, integridade, autenticação de mensagens, proteção contra replay para o trafego RTP e também para seu protocolo de controle, o RTCP. O SRTP não só define um conjunto de transformações padrão de criptografia, mas também permite que futuramente possam ser introduzidas novas transformações a este protocolo [Baugher et al. 2004]. Uma transformação de criptografia consiste em capturar o *payload* (carga útil) de uma aplicação e alguns dados associados como entrada, e transformar esses dados da entrada, de modo a garantir a privacidade do *payload* e a integridade de ambos, o *payload* e os dados associados [Kohno et al. 2003]. Com o gerente de chaves apropriado o SRTP é seguro para aplicações unicast, assim como multicast [Baugher et al. 2004].

O SRTP pode alcançar alto *throughput* (vazão de dados) e baixa expansão de pacotes, SRTP prova que fornece uma proteção adequada para ambientes heterogêneos, ou seja, ambientes onde existem redes com e sem fio. Para alcançar essas características, as transformações padrão são descritas baseadas em uma cifra de fluxo aditivo para criptografia, uma função hash com chave para autenticação de mensagens, e um índice implícito para sequencia e sincronização baseado no número de sequência RTP. Para o SRTP e número de índice para *Secure* RTCP (SRTCP) [Baugher et al. 2004].

Dessa forma, o objetivo do SRTP é assegurar a confidencialidade do *payload*, dos pacotes RTP, RTCP e a integridade dos pacotes RTP, RTCP inteiros, em conjunto com a proteção contra pacotes repetidos. Esses serviços de segurança são facultativos e independentes uns dos outros, exceto para a proteção de integridade dos pacotes SRTCP que esses serviços de segurança são obrigatórios, já que mensagens com erros ou mensagens maliciosas no RTCP podem interferir negativamente no processamento do fluxo RTP [Baugher et al. 2004].

Além do que já foi mencionado, o SRTP possui alguns recursos adicionais para auxiliar no gerenciamento de chaves e assim proporcionar maior segurança. Por exemplo, uma única chave mestre a qual fornece proteção, favorecendo confidencialidade e integridade para os fluxos SRTP e SRTCP através de uma função de derivação de chaves, que fornece chaves de sessão de forma segura derivando da chave mestre. Além disso, a derivação de chaves pode ser configurada para que a chave de sessão seja atualizada periodicamente, limitando assim, a quantidade de dados transmitidos por uma chave fixa [Baugher et al. 2004].

Conceitualmente, o SRTP funciona como uma implementação que reside entre a aplicação RTP e a camada de transporte. O SRTP intercepta os pacotes RTP e logo transmite os pacotes SRTP equivalentes para a camada de transporte no lado do envio. A seguir, o receptor intercepta os pacotes SRTP e passa os pacotes equivalentes RTP para a camada superior [Baugher et al. 2004]. O formato dos pacotes SRTP é mostrado na Figura 3 [Systems Cisco 2009], destacando a parte na qual é aplicada a criptografia e a parte na qual é aplicada a autenticação. Também é destacado os campos originais e os campos adicionais referentes ao SRTP.

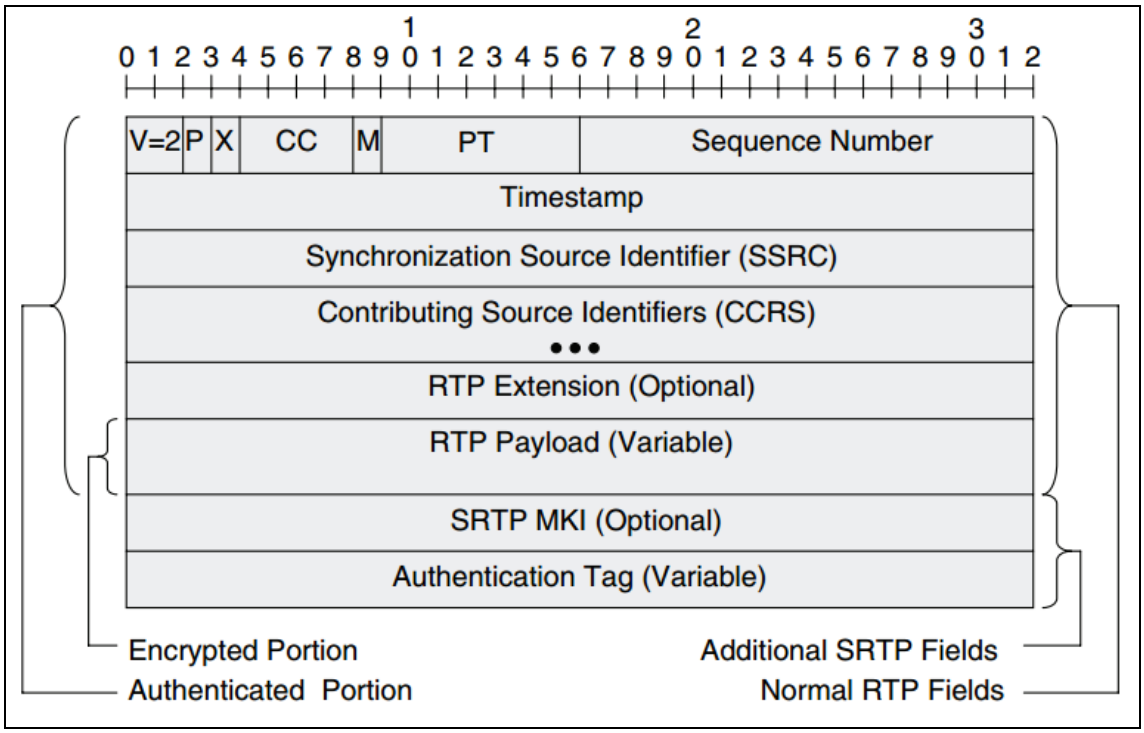


Figura 3. Formato de um pacote SRTP

A parte criptografada consiste na criptografia da carga útil dos pacotes RTP, incluindo padding, quando este estiver presente. A parte criptografada pode ser do mesmo tamanho do pacote RTP ou maior. As transformações padrão pré-definidas não fazem uso do padding, nesse caso o *payload* do RTP e do SRTP são idênticas. As transformações pré-definidas também garantem um baixo custo computacional, um tamanho de código pequeno e independência da aplicação com as camadas subjacentes [Baugher et al. 2004].

Dois campos do SRTP que não se encontram no RTP serão adicionados, o *Master Key Identifier* (MKI), um campo opcional de comprimento configurável utilizado para informar a chave mestre, a partir da qual foram obtidas as chaves de sessão, e *Authentication Tag*, um campo recomendado, utilizado para armazenar os dados de autenticação de mensagens do seu cabeçalho RTP e o *payload* referente a ele [Systems Cisco 2009].

O *Secure* RTCP (SRTCP) fornece os mesmos serviços de segurança para o RTCP da mesma forma como o SRTP faz com RTP. Autenticação de mensagens SRTCP é obrigatória e assim protege os campos RTCP e mantem os contadores de sequência dos pacotes [Baugher et al. 2004]. Para criar um pacote SRTCP são adicionados 3 campos ao pacote RTCP, o *Encryption Flag* com o índice SRTCP, o MKI referente ao SRTCP e o *Authentication Tag* . O pacote SRTCP é mostrado na Figura 4 [Systems Cisco 2009].

A simbologia E na Figura 4 representa o *Encryption Flag* de um bit, que indica estando o pacote criptografado ou não. O índice SRTCP de 31 bits se faz necessário e é incrementado a cada pacote SRTP enviado [Systems Cisco 2009]. O campo MKI do SRTCP é um campo opcional o qual indica a chave mestre, a partir das quais foram

obtidas as chaves de sessão para a criptografia e autenticação, e *Authentication Tag* que nos pacotes SRTCP é obrigatório e de comprimento configurável, o qual é utilizado para armazenar os dados de autenticação de mensagens para o seu cabeçalho RTCP e o seu *payload* [Baugher et al. 2004].

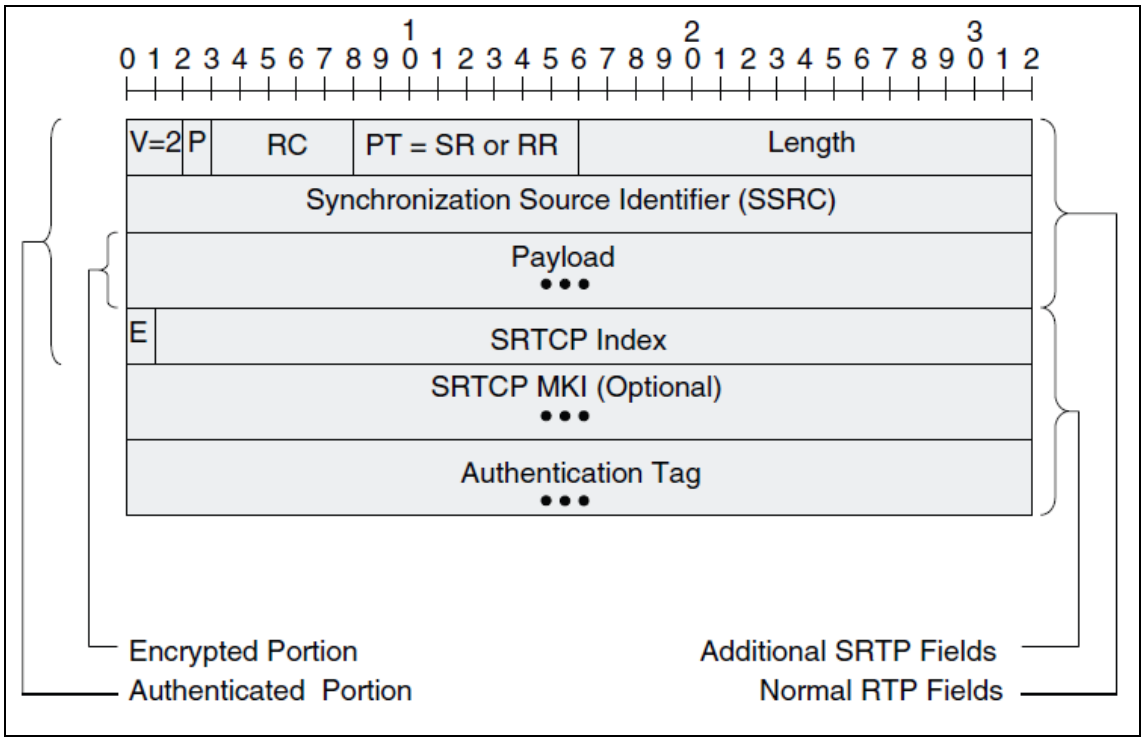


Figura 4. Exemplo do formato de um pacote SRTCP

Embora existam numerosos algoritmos para realizar a criptografia e autenticação de mensagens que possam ser utilizadas no SRTP, serão mostrados unicamente os algoritmos predefinidos. A cifra padrão é o *Advanced Encryption Standard* (AES) e para criptografia no SRTP são definidos dois modos de funcionamento do AES, o AES no *Counter Mode* e o AES no f8-mode que é uma variação do modo *output feedback mode* (OFB) [Baugher et al. 2004].

A transformação de criptografia definida no SRTP mapeia o índice e a chave secreta do pacote SRTP em um segmento *keystream* pseudoaleatório [Eastlake 1994]. Um *keystream* é definido por ser um fluxo de caracteres aleatórios que quando combinados com um texto, resultam em um texto cifrado. Cada segmento *keystream* é capaz de criptografar um pacote RTP. O processo de encriptação de um pacote consiste em gerar o segmento *keystream* correspondente ao pacote, e em seguida, produz bit a bit a parte encriptada do pacote SRTP. A descriptografia é feita da mesma maneira, porém trocando os papéis do texto simples e encriptado. A definição de como o *keystream* é gerado, dado o índice, depende da cifra e o seu modo de funcionamento [Baugher et al. 2004].

Os octetos de cada segmento *keystream* podem ser reservados para uso em um código de autenticação de mensagem. No caso em que o *keystream* utilizado para a encriptação começa imediatamente após o último octeto reservado. Os octetos reservados iniciais são chamados de "*keystream prefix*", e os octetos restantes são

chamados de "*keystream sufix*". O *keystream prefix* não devem ser utilizados para criptografia [Baugher et al. 2004]. O processo é mostrado na Figura 5.

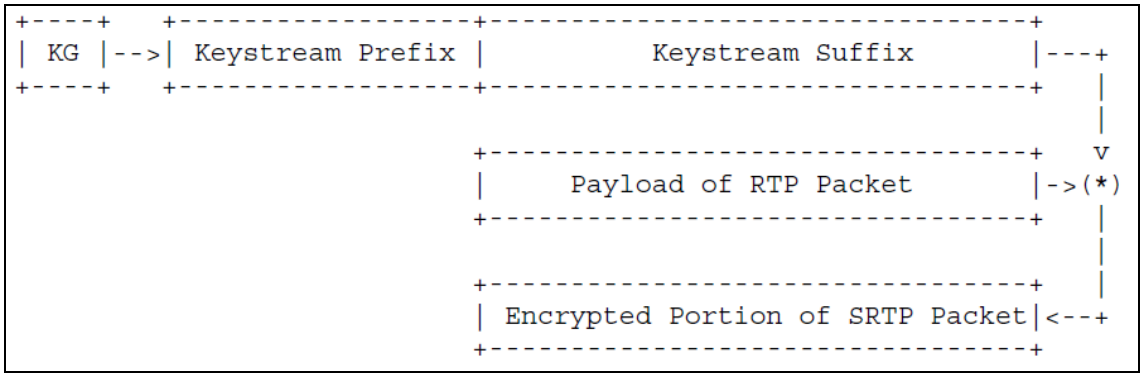


Figura 5. Processamento padrão de criptografia SRTP. Aqui KG indica o gerador keystream, e (*) indica a função binaria "ou exclusivo" (XOR).

Conceitualmente, o modo *counter* consiste em criptografar inteiros sucessivos com o objetivo de embaralhar o ponto de partida da sequência inteira. Cada pacote é codificado com um segmento *keystream* distinto e uma chave de sessão. Já no modo f8 a diferença básica com relação ao modo *counter* é que o modo f8 passa por um processo de realimentação, ele gera blocos *keystream*, e então é aplicado um XOR com o bloco de texto simples (não criptografado) para se obter o texto cifrado (criptografado). Com isso, essa *keystream* será utilizada como entrada junto com a chave de sessão para cifrar o próximo bloco. A Figura 6 ilustra as diferenças entre o *counter mode* e o OFB. Na figura, a saída do *block cipher encryption* se refere à *keystream* [Mark 2014].

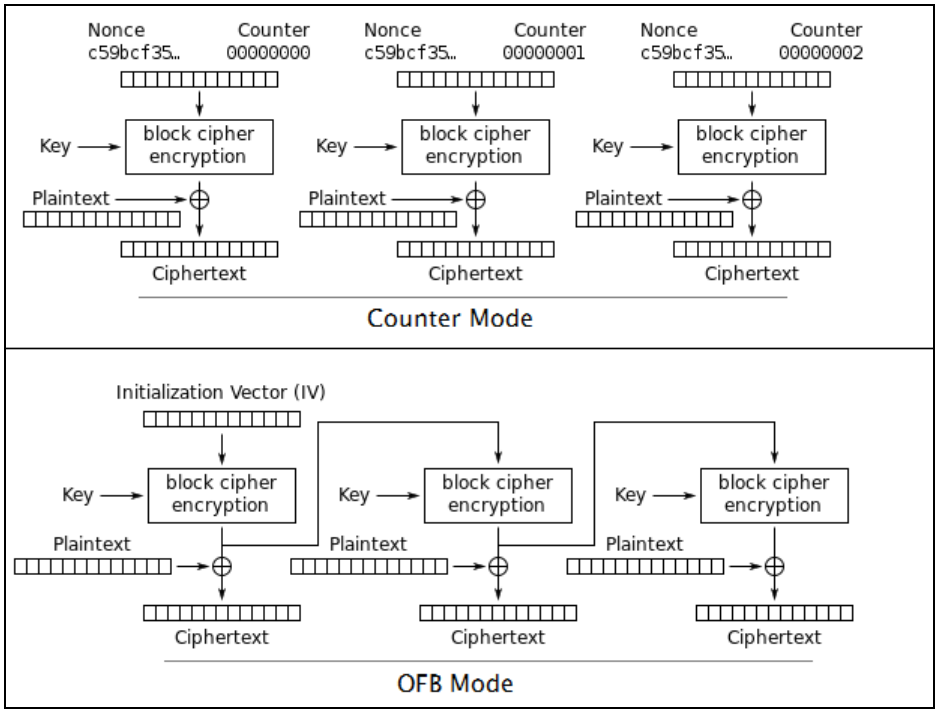


Figura 6. Diagrama do counter mode e OFB mode do AES.

Existe também a *NULL cipher* (cifra NULL), que é utilizada quando não é solicitada confidencialidade para RTP/RTCP. O *keystream* pode ser pensado como uma

sequencia de zeros "000 .. 0", ou seja, a criptografia terá como função simplesmente copiar a entrada de texto plano para a saída de texto cifrado [Baugher et al. 2004].

Os algoritmos de criptografia mencionados anteriormente protegem contra confidencialidade, mas não protegem a integridade, autenticação ou proteção de replay. Para obter essas características o SRTP faz uso de dois mecanismos, o primeiro é uma função Hash com chave secreta e código de autenticação de mensagem (MAC). O HMAC faz uso especificamente do HMAC-SHA1 na qual a saída HMAC serão os bits mais à esquerda, e o mecanismo requisitado para proteção de replay se faz através de um receptor, que tem como função manter os índices das mensagens previamente recebidas. Sendo assim o SRTP só processará as mensagens que apresentem um índice não recebido anteriormente, caso contrário, a mensagem será descartada [Baugher et al. 2004].

2.3. SIP

O *Session Initiation Protocol* (SIP) é um protocolo de iniciação de sessão utilizado em alguns sistemas de multiconferência. O SIP também é um protocolo de aplicação utilizado para estabelecer, modificar, e terminar sessões de comunicação multimídia entre um ou mais participantes [Rosenberg et al. 2002]. Entre estas sessões multimídia estão inclusas conferências multimídia, ensino à distância, telefonia via internet e aplicações similares. Para tal, o SIP utiliza como porta padrão a porta 5060 [Firestone et al. 2007].

O padrão SIP define um método para estabelecer uma conexão de SIP segura, isto é, o SIP faz uso do *Tansport Layer Security* (TLS) pela porta 5061. Neste caso, o endereço SIP utilizado habitualmente sip:URL é substituído pelo sips:URL. O TLS oferece a autenticação de um só lado ou autenticação mútua, e fornece a criptografia e integridade para o fluxo de dados em ambas as direções. A desvantagem da utilização do TLS é que seu mecanismo de segurança funciona salto por salto (hop-by-hop). Para a conexão fim a fim ser segura, todos os dispositivos ao longo do caminho devem confiar uns nos outros. As mensagens de sinalização SIP podem especificar o RTP seguro (SRTP) para a criptografia na transmissão de mídia [Firestone et al. 2007].

2.4. SDES

Session Description Protocol Security Descriptions (SDES) é uma forma segura de negociar as chaves para o SRTP [Andreasen et al. 2006]. As chaves são transmitidas como anexo do *Session Description Protocol* (SDP) de uma mensagem SIP, então um endpoint envia a proposta de chave para o outro endpoint, e em seguida, a outra parte aceita a chave e o fluxo criptografado de dados prossegue. Todas essas informações remetem afirmar que a camada de transporte SIP deve certificar-se de que ninguém pode ver o seu anexo, ou seja, de que esta deve ser uma conexão SIP protegida contra terceiros por meio de criptografia e autenticação, caso contrário, a chave poderia ser interceptada [Puangpronpitag et al. 2012]. A principal vantagem deste método encontra-se na sua simplicidade.

As descrições acima explicam o porquê da troca de chaves SDES funcionar unicamente através de um SIP Seguro usando TLS, um canal de comunicação seguro, que trabalha similarmente ao HTTPS. A Figura 7 ilustra o funcionamento do SDES e

explica a forma como depois de definida a chave começa a transmissão de dados através do SRTP.

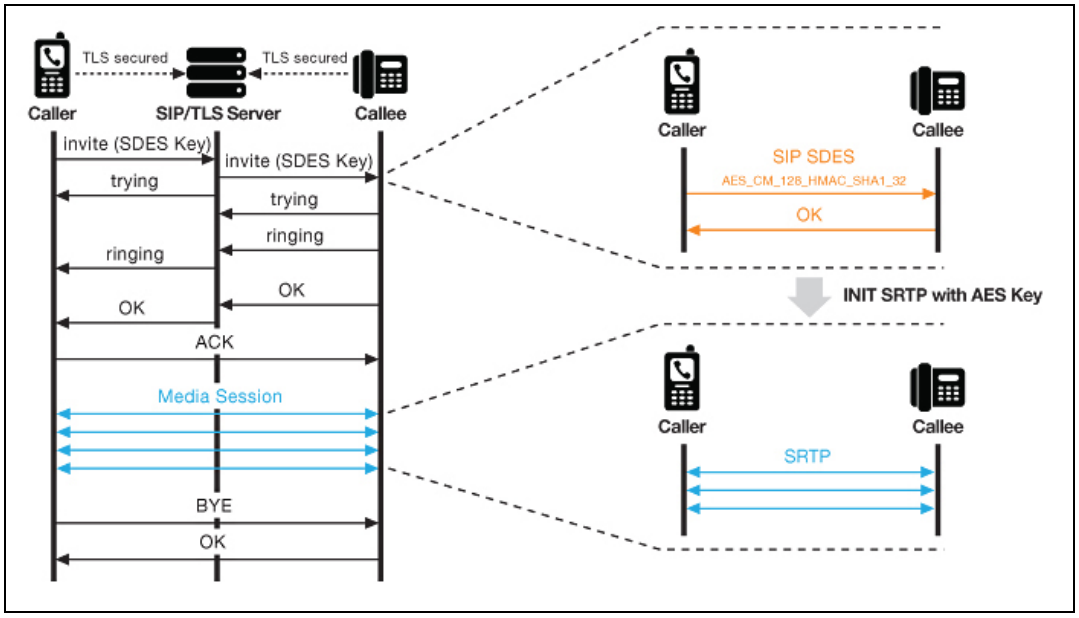


Figura 7. Negociação de chaves pelo protocolo SDES.

2.5. RTMP

Adobe's Real Time Messaging Protocol (RTMP) é um protocolo destinado a transportar fluxos paralelos de vídeo, áudio e mensagens de dados, assim como informações de tempo associadas, o protocolo é especifico para o Flash Player [Parmar et al. 2012].

O RTMP é um protocolo simples, seus pacotes são enviados através de uma conexão TCP estabelecida entre o cliente e o servidor, que por padrão utiliza a porta 1935. Para serializar (processo de transmitir um objeto por uma conexão de rede) e desserializar objetos, enquanto eles são enviados pela rede, o RTMP faz uso de um formato binário compacto chamado *Action Message Format* (AMF) [Parmar et al. 2012].

O AMF é um formato binário utilizado para serializar gráficos de objetos, como o *ActionScript* (programação orientada a objetos própria da Adobe e utilizada no RTMP), ou enviar mensagens entre um cliente Flash e um servidor. O AMF e o RTMP em conjunto, são utilizados em duas funções: estabelecer conexão e para o controle que realiza a entrega de streaming de mídia. Neste caso os dados do AMF são encapsulados no cabeçalho o qual define as características da mensagem, como o tipo ou o comprimento [Parmar et al. 2012].

Atualmente, o Mconf utiliza o RTMP para a transmissão de mídia quando a aplicação é utilizada pelo browser. Mas o protocolo RTMP tem múltiplas variações além da forma padrão, sendo que duas destas variações oferecem segurança ao protocolo. As variações são: RTMPE, RTMPS [Towers et al. 2010] e RTMPT [Help 2014].

O RTMPE é um mecanismo de criptografia próprio da Adobe. O *Flash Media Server* criptografa o conteúdo em tempo de execução. O RTMPE utiliza as primitivas

padrão de criptografia que consistem na troca de chaves Diffie-Hellman, e o Hash de autenticação HMACSHA256. Enquanto os dados são transmitidos, o RTMPE gera um par de chaves RC4, uma chave cifra os dados enviados pelo servidor, e a outra chave cifra os dados enviados para o servidor. O RC4 não é considerado um dos melhores algoritmos simétricos de criptografia, mas existe o RTMPS que confere maior segurança para o RTMP [Help 2014].

O *Flash Media Server* pode ser configurado para fazer uso de um certificado SSL que fornecerá criptografia de 128 bits. Quando a criptografia SSL é utilizada, o protocolo se torna o RTMPS [Towers et al. 2010]. O RTMPS utiliza HTTPS e normalmente se comunica pela porta 443. A criptografia SSL requer a utilização da versão do Flash Player 8 ou superior [Towers et al. 2010].

Secure Sockets Layer (SSL) é um protocolo que utiliza os dois tipos de criptografia, assimétrica e simétrica. Inicialmente, utiliza a criptografia assimétrica para poder realizar a troca de chaves de forma segura, assim continua com a comunicação de dados com criptografia simétrica. O SSL é um protocolo de segurança separado para segurança, e seu funcionamento fica entre a camada de aplicação e a camada de transporte, assim como o SRTP. Desta forma o SSL pode ser aplicado em diversos protocolos de aplicação [Freier et al. 2011].

O RTMPT é uma variação do RTMP, mas que não envolve o uso de criptografia. Diferentemente das mencionadas acima, o RTMPT tem como objetivo poder transitar mesmo quando conexões diretas no socket RTMP não são permitidas, pela existência dos firewalls e servidores de algumas organizações. Basicamente, o RTMP funciona como um tunelamento, onde os dados RTMP são encapsulados e trocados via HTTP, e as mensagens do cliente são dirigidas para a porta 80 no servidor. A sessão encapsulada pode conter pacotes RTMP simples, RTMPS, o RTMPE [Help 2014].

3. Objetivos e Organização do Trabalho

Para alcançar o objetivo pretendido, primeiramente será realizado um estudo teórico sobre os protocolos e métodos com os quais podemos alcançar a segurança desejada para o sistema. Segundo, fazer uma análise de como esses protocolos podem ser implementados. Posteriormente, será feita a implementação destes mecanismos de segurança em sistemas de multiconferência, com a finalidade de tornar o sistema seguro. E por último, será realizada uma etapa de testes a fim de garantir que o sistema tornou-se seguro e para poder mensurar o impacto que este acréscimo de segurança acarreta nesses sistemas.

Atualmente, a primeira etapa já foi concluída, a segunda e a terceira etapa estão sendo estudadas. Durante o andamento das seguintes etapas espera-se que se possa compreender e identificar melhor as dificultardes de implantar esses mecanismos de segurança.

A seguir, serão apresentadas as atividades a serem realizadas no TG2, a Tabela 1 mostra o provável cronograma de trabalho para o segundo semestre.

1. Compreensão da arquitetura de sistemas de multiconferência, bem como o funcionamento em detalhes.
2. Projeto de implantação dos mecanismos de segurança.

3. Implementação desses mecanismos de segurança em sistemas de multiconferência.
4. Avaliação de segurança e testes.
5. Coleta de resultados e realização da análise.
6. Redação do Trabalho de Graduação 2.
7. Apresentação do Trabalho de Graduação 2.

Tabela 1. Cronograma das atividades que serão desenvolvidas no TG2.

	Junho	*Julho*	*Agosto*	*Setembro*	*Outubro*	*Novembro*	*Dezembro*
1	X	X					
2	X	X					
3		X	X	X			
4			X	X	X		
5				X	X	X	
6					X	X	X
7							X

4. Conclusão

Através da realização deste trabalho foi possível compreender melhor como funciona a segurança e criptografia em transmissões de tempo real, como esses mecanismos podem ser adicionados às aplicações e como cada um destes protocolos emprega seus mecanismos para fornecer segurança. Por tanto, este trabalho irá possibilitar, de uma maneira mais clara e estruturada, a organização e elaboração do Trabalho de Graduação 2.

Referências

Andreasen F, Baugher M, Wing D. Session Description Protocol (SDP) Security Descriptions for Media Streams. IETF RFC 4568. 2006.

Baugher M. et al. The Secure Real-time Transport Protocol (SRTP). IETF RFC 3711. 2004.

Begen A, Friedrich E. Multicast Acquisition Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs). IETF RFC 6332. 2011.

Eastlake D et al. Randomness Recommendations for Security. IETF RFC 1750. 1994.

Firestone S, Ramalingam T, Fry S. Voice and Video Conferencing Fundamentals. Cisco Systems. Security Design in Conferencing, 257-327. 2007.

Freier A, Karlton P, Kocher P. The Secure Sockets Layer (SSL) Protocol Version 3.0. IETF RFC 6101. 2011.

Help Adobe. Developing FLASH LITE 4 Applications. http://help.adobe.com/en_US/flashlite/dev/4/flashlite_4_developing.pdf. Disponível em 08/06/2014.

Kohno T, Palacio A, Black J. Building Secure Cryptographic Transforms, or How to Encrypt and MAC. Agosto, 2003. 37 páginas.

Mark R. Symmetric-key cryptography. http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/symmetric-key.html. Disponível em 08/06/2014.

Mconf. http://mconf.org/m/. Disponível em 08/06/2014.

Parmar H, Thornburgh M. Adobe´s Real Time Messaging Protocol. Adobe Systems Incorporated. 2012

Puangpronpitag S, Kasabai P, Pansa D. An Enhancement of the SDP Security Description (SDES) for Key Protection. 2012

Rosenberg J. SIP: Session Initiation Protocol. IETF RFC 3261. 2002.

Systems Cisco. Cisco TelePresence Secure Comunications and Signaling. Cisco TelePresence Security Protocol Details. Cisco. 2009.

Towers K, Green T. Video content protection measures enabled by Adobe Flash Media Interactive Server 3.5. Adobe Systems Incorporated. Enable basic video protection with Flash Media Interactive Server, 7-10. 2010.