DEGREE PROJECT, IN PROGRAMME , SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Privacy Ensuring SRTP for Cloud Conferencing

Maria Haider

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Acknowledgements

I will always be grateful to Swedish Institute for giving me the opportunity to complete my master degree from KTH, Royal Institute of Technology by awarding me with a full scholarship.

I would like to thank my supervisor, John Mattsson at security research, Ericsson, Stockholm for presenting me with this opportunity to work on this interesting project. He has been a great mentor by always guiding me to the right direction. His helpful feedback and suggestions inspired me to work harder and find the desired solutions.

I would also like to thank my supervisor from KTH, Markus Hidell and my examiner at KTH, Peter Sjödin for helping me to complete the thesis by providing useful guidelines and feedback, especially on the analytic part of the work.

Abstract

Multimedia conferences held using services provided by clouds owned by third party companies are becoming increasingly popular. While using such services, end users will want to keep their audio/video data private when they pass through the servers situated in the cloud. Application of SRTP (Secure Real-time Transport Protocol) in such use cases fail to provide the desired privacy because it leads to sharing the master keys for encryption and authentication of the media content with the semi trusted media servers of the cloud. As a solution, modifications of SRTP are proposed in this thesis with the result of redesigning the security mechanisms of RTP header extensions and RTCP packets by separating the cryptographic contexts and keying materials for protecting end-to-end sensitive data. A couple of design choices for key management through DTLS-SRTP for Cloud conferencingare also proposed. Moreover, analysis of existing solutions for modifying SRTP packets for cloud conferences have also been carried out in this project. The solutions are found by studying related protocols, understating the problems and analyzing current solutions if there were any. The proposed solutions show different alternatives to solve a specific problem and their tradeoffs in terms of complexity and compatibility with current standards.

Keywords

Network Security, Semi trusted cloud conference Server, SRTP, RTP, DTLS-SRTP

Table of Contents

List of Figures

List of Abbreviations

AEAD -Authenticated Encryption with Associated Data
AES-CM - Advanced Encryption Standard in Counter Mode
AES-GCM - Advanced Encryption Standard in Galois/Counter Mode
CNAME – Canonical Name
CSRC – Contributing Source Identifier
DTLS – Datagram Transport Layer Security
EEIV – End-to-End IV
EKT – Encrypted Key Transport
HTTP – Hyper Text Transfer Protocol
IANA – Internet Assigned Numbers Authority
IETF – The Internet Engineering Taskforce
ISN – Initial Sequence Number
IV – Initialization Vector
KMF – Key Management Function
MCU – Multiple Control Unit
MIKEY – Multimedia Internet Keying
MKI – Master Key Identifier
MSS – Media Switching Server
PT – Payload Type
PUV – Packet Unique Value
ROC – Roll Over Counter
RTCP – RTP Control Protocol
RTP - Real-Time Transport Protocol
SDES - Session Description Protocol Security Descriptions
SNI – Server Name Indication
SPI – Security Parameter Index
SRTCP – Secure RTP Control Protocol
SRTP - Secure Real-Time Transport Protocol
SSS – SRTP Source
SSRC – Synchronizing Source Identifier
TCP – Transmission Control Protocol
TLS – Transport Layer Security
UDP – User Datagram Protocol
URI – Uniform Resource Identifier
WebRTC – Web Real-Time Communication

1 Introduction

It is becoming a popular trend to host conferences between members of enterprises using services provided by cloud operators. The popularity is due to the cost effective property of such solution. Instead of using proprietary heavy weight hardware, virtualization of computing resources in the cloud that offer broader scope of scalability comes forward as a smart choice to be used by business or nonprofit organizations in hosting live conferences. But with this advantage, the risk of sharing corporate information with untrusted service providers also rises. This calls for solution to maintain the security of information flowing through the servers in the cloud. This thesis investigates various possible solutions and proposes the ones that best conforms to current standards. The work was performed for Ericsson which objective was to understand the communication protocols, analyze the existing proposals, their limitations and propose protocol modifications related to audio/video conferences held using third party cloud based services.

1.1 Background

Videoconference or multimedia conference is a means for communication through which people from multiple locations can interact using audio, video or text based conversations in real time [15]. In a video conferencing system, participants typically uses video cameras and microphones to capture audio and video data. With the use of audio/video compression techniques, these data are encoded and transmitted using digital networks from one location to another. On the receivers' sides, these packets of data are decoded and converted to audio and video information to be displayed on the monitors and be heard from the microphones of the recipients. The compression and decompression techniques are usually performed by hardware or software systems [22]. The transmission of real time audio and video data are governed by standardized communication protocols. The presence of internet and the advancement of computing processors' powers have made the use of videoconferences in business and commercial organizations a medium for internal or external communication and transaction. Use of videoconferences remove the cost of travel and overcome the barrier of geographical distances in holding business meetings within and between organizations. Participants not only can speak and view each other but also share files, graphical presentations and operate computers from remote locations [23].

Presently multimedia conferences are held with conference servers belonging to third party cloud-based service providers. To understand what it means, it is important to understand what cloud computing is and how it is used in multimedia conferences.

Cloud computing is availability of computing resources to be used by general users, developers or companies as a service in the form of software, hardware, developing platforms or infrastructure where the actual physical resources are situated in remote data centers. In most cases, these resources are owned by third party entities. The users employ computing power of these resources without having to manage, maintain or pay for their physical existence. Though

they may have to pay for using these resources from third party owners. This phenomenon can be termed as using virtualized resources. The advantages of using such services is that they are available on demand, same resources can be used by multiple users in different times, capabilities and size of such resources have dynamic nature as they can be relocated or scaled up or down per demand [24] [27].

In typical media conferences with multiple participants, the media packets are sent from the audio/video sources through a centralized media server. This server receives the packets from one source and delivers them to intended recipient(s). The enterprises holding the multimedia conferences build their own devices to facilitate the media switching. Owning and maintaining such system is a costly affair. In this scenario, cloud computing presents itself as an economic alternative. Virtualized servers can be set up in cloud and used as media forwarders for the conference instead.

The participant devices connect to the cloud servers to exchange their audio/video data. Their lower processing power offers a low cost incurring option where the processing of the media packets are to be left for the end devices to perform. They do not have to mix audio or transcode or recompose video, thus making it easy to be deployed on simple computing hardware hosted in clouds. Cloud based conference servers also offer more scalability in terms of increasing the size of the conference by letting multiple participants join in ongoing sessions [4]. Consequentially, corporations are leaning in to replace the costly proprietary hardware solutions with lightweight and cost effective virtual servers provided by third party cloud providers.

The communication protocols involved with real time audio/video exchange for multimedia conferences that are investigated in this thesis are RTP, RTCP, SRTP and DTLS-SRTP. RTP (Real-time Transport Protocol) provides functions to manage transmission of real time multimedia data between end-to-end devices in a conference [2]. RTCP provides means of controlling and monitoring RTP packets [2]. SRTP (Secure Real-time Transport Protocol) has been designed to protect confidentiality, message integrity, authentication and replay protection of both RTP and RTCP packets [1]. A secure transport layer channel is needed to send the encryption and authentication keys used in SRTP from one device to another in a cloud conference. The protocol that is used for establishing such channel is DTLS (Datagram Transport Layer Security) and the special extension of it is known as DTLS-SRTP [3]. This extension and the related protocol is considered as a key management protocol for SRTP.

RTP and RTCP packets exchanged during a videoconference are encrypted and authenticated using cryptographic keys following the specification of SRTP. SRTP is designed for topologies in which all the devices taking part share these keys. The relationship is explained in Figure 1 below.

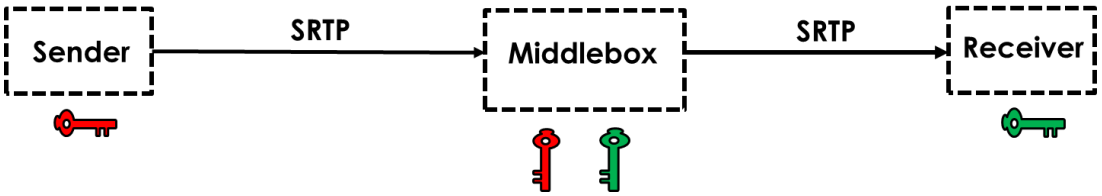


Figure 1. Typical SRTP use case

In this thesis, the use case that has been considered replaces the enterprise owned middleboxes with centralized servers hosted in cloud provided by third party organizations which is shown in Figure 2. These cloud based conference servers are lightweight solutions that do not perform any manipulation on the media but just forward them. Therefore, the servers in the cloud do not directly need to have access to media encryption keys. This particular use case has been labelled as cloud conferencing throughout this thesis. But upon application of SRTP in cloud conferencing, along with the end devices belonging to the participants in the conference, the virualized servers in the cloud also end up sharing the encryption keys.

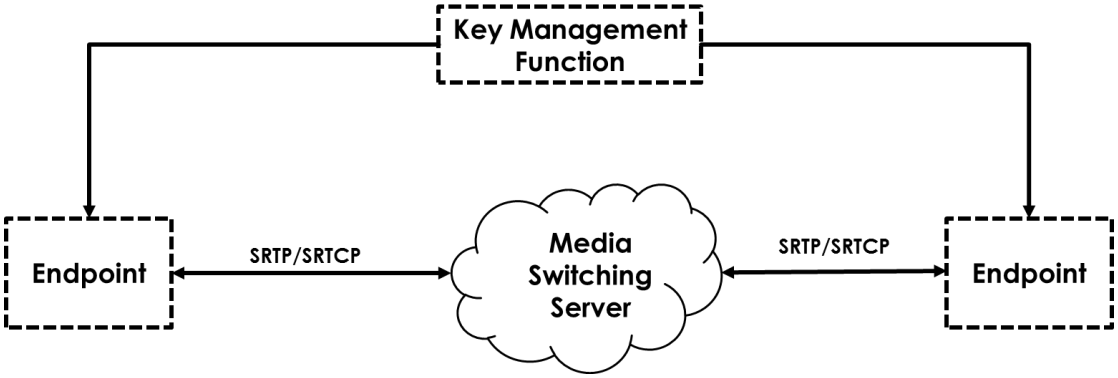


Figure 2. Cloud conferencing

In traditional video conferences, the middleboxes are trusted devices owned by conference organizers to share the SRTP keys with. In case of cloud conferencing, the organizations use forwarding services from third party cloud providers. The information flowing through these servers are neither their properties nor intended for them to have access to. But it cannot be guaranteed that the servers in the cloud will not try to decrypt the media packets and access the contents if they have access to the media encryption keys. [6] In traditional conferencing, the media forwarding is done within private networks behind firewalls. Whereas in cloud environments, in the absence of close monitoring of media owners, it is much more likely to have adversaries gain access to some open ports of cloud resources and consequentially have accesses to the cryptographic keys that are used to secure the conference audio/video data [7]. This way, the attackers are able to fake its identity to join in the conference or simply decode and view or listen to live conversations that may carry sensitive corporate information [4]. Therefore, sharing media encryption keys with cloud servers can lead to breach of privacy of data and risk of exposing it to insecure

public networks. The openness and easy provision of scalability in a cloud structure can cause absence of fixed infrastructure and security boundaries. [28]

Therefore, the accepted security model for a cloud conference is the one whose participants trust the cloud resources to forward their media to intended recipients but not enough to make their conversations visible to the cloud operators. To realize the model, it becomes mandatory not to share the encryption keys with the servers in the cloud. Therefore, the solution is to modify SRTP to introduce two sets of keys. This is how the media encryption will be separated from the source authentication keys used by the servers in the cloud.

SRTCP packets are used to control and monitor media flow in cloud conferences. RTP header extensions are used to carry specific application related information in the conference, the detail of which are given later in this thesis. It is also important to figure out how any data carried in these two formats that may need end-to-end protection in a cloud conference is going to be handled.

With cloud conferencing, two unique SRTP master keys and protection profiles are introduced separately for end-to-end and hop-by-hop operations. Therefore, along with the modification of SRTP, modification of existing key management protocols are also needed to facilitate the security features of conferences held in cloud. Among such protocols, this thesis chooses to design modification for DTLS-SRTP mechanism to make it suitable for the double key exchange for cloud conferencing.

In conclusion, the thesis focuses on a broader SRTP solution for cloud conferencing based on end-to-end privacy.

1.2 Problem

The structure of the cloud conferencing demands for security solution which cannot be fulfilled with the current SRTP architecture. Since the server in the cloud belongs to a third party service provider which is responsible for forwarding media to the participants in a conference, the media sources belonging to the endpoints do not want to risk their personal or corporate information to be inspected by any other entity except for the receivers whom they are meant to be shared with. To achieve that, the double layer of security that is needed cannot be established without any meaningful modification of SRTP. The separation of cryptographic context, keying materials, the need of protection of end-to-end sensitive data belonging to RTP header fields, RTP header extensions and RTCP packets in the event of passing through a semi trusted server cannot be maintained by following the present SRTP supported security solutions. Moreover, existing key management protocols do not support the exchange of end-to-end security context and other identity parameters exclusively needed by cloud conferencing.

1.3 Purpose

The basic SRTP profile is currently unable to fulfil the needs to provide maximum security for the information generated by the participants in a cloud

conference. The use case investigated by this thesis is different from the ones SRTP is built for. Also the current key management protocols used by SRTP to negotiate the keying materials is not sufficient to cater the needs of cloud conferences. The purpose of this thesis is to research the different ways and suggestions of modifications to SRTP and DTLS-SRTP and propose efficient design choices to make it applicable for this scenario. It is also the purpose of the research to find the trade-offs and the feasibility of implementing such modifications in terms of complexity, scalability, assurance of smooth transmission of the media along with possible security implications.

1.4 Goal

The objective of this master thesis is to study the problems and the current proposals to suggest various ways to solve the problems and list their respective pros and cons. It is expected to deliver proposals of SRTP protocol modifications and architecture, along with updated requirements. To be specific, the target is to propose design choices for securing RTCP and RTP header extensions along with suitable modifications of DTLS-SRTP. The description of the alternative choices and their detailed analysis of compatibility with the requirements are also part of the goal.

1.4.1 Benefits and Ethics

Nowadays the enterprises and companies are interested to pursue a cost effective solution for holding conferences between members of their enterprises or others'. This is why media conferences hosted by third party cloud based servers are preferred. Commercial services are also marketed based on such architecture. The deliverables from this thesis are expected to strengthen the security of such services and help preserve the confidentiality of both corporate and personal information. It is not only beneficial for the companies who will buy the service from the cloud operators, it also frees the seller of the service from owning any third party data and risking any legal complicacies rising from the claim of using such data.

To conduct this research for Ericsson, it is obligatory to follow the code of business ethics documented by the company and the rules to protect its confidential and proprietary information.

It is always difficult to monitor security breaches in real time communication channels such as the use case discussed in this thesis employs. The laws associated with misuse and ownership of data varies from one country to another which comes across as an ethical dilemma for all parties involved. Another interesting ethical issue is the side effects of having a medium of communication such as multimedia conference too secure. People with harmful intentions can take advantage of the secure medium and fulfil their purposes by hiding behind the protected services from the eyes of law.

If thinking from the perspective of sustainability, online video conferencing with the use of cloud services help make better utilization of resources available by an organization. Such services provide the opportunity to hold urgent meetings between employees placed in different locations. This optimizes the productivity by making sure that they do not have to leave their current projects

to attend a short meeting. This also minimizes travelling required by the participants to attend a conference which profits the environment by helping reduce carbon emissions. This is a twofold benefit for both the organization and the environment in terms of saving or protecting resources for future valuable uses.

1.5 Methodology

The primary method to be followed for this particular thesis is analysis and theoretical research. Analysis of the related protocols, existing solutions and conducting research on finding various solutions and study of their feasibilities are integral part of this research. Based on the study, final proposals of design choices are made. Finally these design choices become part of the modified SRTP. The process is explained in elaboration below.

The first step is to identify the trust model for cloud conferencing. It decides which entities taking part in the conference are trusted and which are not. This information is vital for redesigning SRTP. Based on the level of trust, SRTP grants which entities have access to cryptographic keys, associated context and private information carried in RTP packets.

The second step is to analyze the existing proposals on modifications of SRTP to determine their drawbacks and find out if they carry any useful ideas upon which the research of this thesis can build up.

This thesis proposes design choices for three components of SRTP applied in cloud conference. These are SRTCP, secure RTP header extension and DTLS-SRTP. For each of these parts, firstly, the security requirements expected to be delivered by these formats are identified based on the trust model of cloud conferencing. Then the current design of each format is studied extensively to determine how to fit the modifications for the new requirements replacing the old format. The roles of different fields and the application of cryptographic operations on different parts of the related packets are studied for SRTCP and secure RTP header extension formats. For DTLS-SRTP, each message, the content of the message and steps are studied to find out how the secure channels are established between the conference entities.

The next step is to propose alternative design choices of the required fields and steps related to three formats. These choices are going to facilitate the transition from a single key architecture to a double-key security model needed for cloud conferencing.

Finally, the most suitable design choice is selected for each format based on simplicity of the design, compatibility with the current specification and supply of maximum protection.

1.6 Delimitations

The thesis does not discuss the use of the protocols for external and prior signalling of setting up the audio/video sessions between the various entities of the cloud conference. Inclusion of studying of such protocols could provide a broader picture of the scenario examined in this thesis. On the other hand, it

would have shifted from the focus of the project which solely concentrates on the security of the information flowing through third party server in the cloud.

One of the requirements mentioned in IETF draft 'Requirements for Private Media in a Switched Conferencing Environment' includes cloud conference's compatibility with WebRTC security architecture [8], [4] This has not been studied in this particular thesis for time limitations. Study of WebRTC could add another choice for transferring end-to-end keying material apart from DTLS-SRTP.

Implementation of the proposed protocols were skipped both for time limitation and not being relevant to Ericsson's expected outcome from the work.

1.7 Outline (Disposition)

This thesis paper is organized into 3 more chapters.

Chapter 2 provides the background of the thesis broken into three sub chapters. The first sub chapter explains what cloud conference is, provides description of each of the entities that make up the structure of it and its security requirements. The second sub chapter contains general study of existing specification of each of the related protocol. The third one contains research and analysis of the current solutions.

Chapter 3 describes the proposed solutions. The chapter is divided between three protocols for which the solutions have been proposed. For each of the protocol, the existing design, the necessity of a new design, analysis of different choices and description of the final design have been discussed.

Chapter 4 holds the conclusion and influence of the work carried out in this thesis.

2 Background

To understand the proposed solutions, it is important to know the details of the use case studied in this thesis. It is also important to understand the related protocols and why their current specifications do not support the needed security features of the use case studied. These topics and the analysis of the existing solutions are presented below.

2.1 Cloud conference

The term – cloud conference is used when live audio or video conferences are held between multiple participants where the media is forwarded through servers situated in the cloud. The protocol that is used to transmit the media is RTP. Security of the RTP packets are provided by SRTP. Control and monitoring packets are sent using the protocol, RTCP.

Application of SRTP in a cloud conference, its trust model and the security solution which are discussed briefly in Section 1.1 are explained in detail with diagrams under this section.

2.1.1 Structure

The basic structure of a cloud conference involves more than one endpoints, a key management function and a Media Switching Server. Figure 3 shows the discussed entities belonging to a conference held in cloud. Figure 3 and Figure 2 from Section 1.1 are same figures.

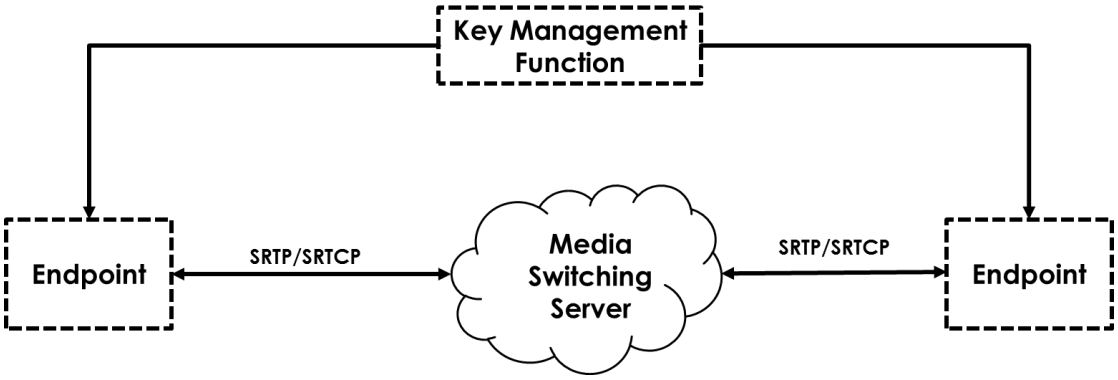


Figure 3. The use case for cloud conferencing

In an actual deployment, there may be more than one Media Switching Server. This structure does not include the interfaces shared between the signaling entities and the conference devices which are used for setting up the multimedia sessions. Apprehending the functions and properties of these three types of entity is sufficient to discuss the security needs and understand the design choices proposed in this thesis.

The properties of each of these entities are discussed below:

- **Endpoint**: Endpoints are end devices that belong to the participants.

The audio/video sources belonging to the Endpoints are capable of encoding and decoding the media themselves. Each Endpoint has to support a specific set of media formats and codecs which will be used by the other Endpoints in a particular conference.

- **Key Management Function (KMF):** It belongs to the enterprise holding the conference or trusted by the participants of the conference to deliver the end-to-end master keys to the Endpoints in a secure way. This way, the negotiation for keying materials becomes scalable without the necessity of each Endpoint directly addressing the other ones for the purpose. KMF also verifies the identities of the participants who will join the conference before delivering the keys.
- **Media Switching Server (MSS):** Instead of a privately owned middlebox by the organization, a Media Switching Server (MSS) in the cloud is going to forward the packets to intended recipients. It will be owned by a third party host.

  Unlike the middlebox, it will not mix, encode/decode or perform any manipulation on the media payloads, it may forward the packets based on audio level indication, meaning it will only forward the media coming from the active speakers. The switching server must have access to some parts of the RTP headers to make the forwarding decisions [4]. It may need to modify some of the RTP header fields of the received RTP packet for transmission purpose [7], the detail of which will be explained in the later part of the thesis. The server in the cloud is also responsible for generating conference control information (RTCP packets) [9]. In the case of end-to-end key management, the server can act as the middle device for relaying the keying information from Endpoint to the KMF. Some of these characteristics of the Switching Server varies with the implementation choices discussed in the thesis later.

2.1.2 Application of SRTP

In SRTP, one master key is shared between two entities in an RTP media session to exchange SRTP packets. In case of a multi-party conference, SRTP/SRTCP packets received by the centralized servers/middlebox are decrypted with the key shared with the senders. Next, the middlebox processes the payloads and re-encrypts them with a key shared with the receiving devices and forwards the resulting SRTP/SRTCP packets to those receivers. This scenario of SRTP master key sharing is depicted in Figure 4. Figure 4 and Figure 1 from Section 1.1 are same figures.

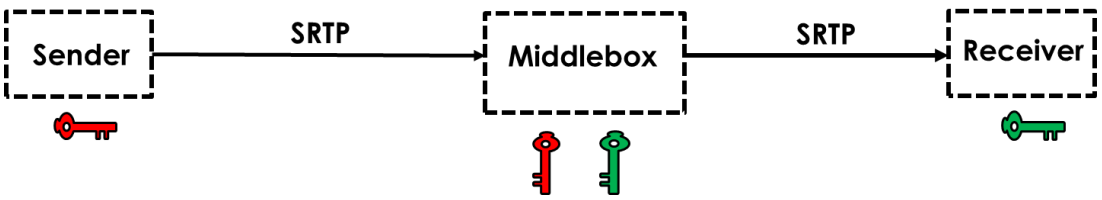


Figure 4. Key sharing for SRTP

Here, the middlebox is owned by the organizer or by the participants in the conversation, thus it is trusted to possess the key to decrypt RTP/RTCP payloads. It provides the hardware to modify the contents it receives, it is capable of encoding or mixing the content of the media (if there were more than one sources). All the entities belonging here are trusted, so all of them are entitled to hold the master keys.

If SRTP is desired to be implemented in the use case for cloud conferencing shown in Figure 3, then it is necessary for the Endpoints to share their master keys with the virualized servers. The consequences of which give birth to below security threats:

- Third party servers gain access to media keys.
- To have risk of exposure to vulnerable networks surrounding unprotected cloud resources.
- Corporate and personal information in SRTP/SRTCP packets might get inspected by unauthorized entities.
- Confidentiality is lost for data that are only intended to be shared with trusted endpoints in -
    - RTP Payloads
    - RTP header fields
    - RTP header extensions
    - RTCP Packet

2.1.3 Trust Model

To determine the security requirements of the use case described above, it is important to categorize the trustworthiness of each of the basic entity belonging to a cloud based conference. From the perspective of ensuring privacy of personal or corporate information forwarded through the servers in the cloud, the entities can be categorized as shown in the trust model in Figure 5. The trust model for the use case investigated in this thesis has been decided by consulting two particular IETF drafts which are [4] and [6].

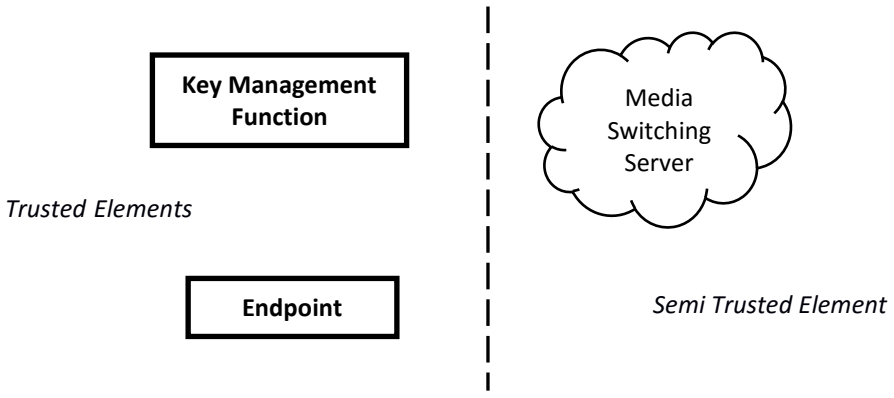


Figure 5. Trust model of cloud conference architecture

In accordance with Section 6.1 of [4], the Endpoints are trusted devices which are permitted to decrypt the SRTP payloads. The Key Management Function is also a trusted device as it is responsible for handing out the master keys to the Endpoints using which the SRTP/SRTCP payloads are protected end-to-end. KMF also has to make sure that the delivery is done in a way that the Media Switching Server has no access to them.

In case of cloud conferencing, the MSS does not have to perform any processing on the media payloads. This allows the Endpoints not to share the media encryption keys with the server in the cloud which security is not possible for the organization holding the conference to monitor [4]. Here, the cloud operator is not completely untrustworthy, since it is being used to deliver the media packets and control the conference which allows it to access and in some cases modify certain RTP header fields to make the forwarding decisions. Only high trust is not assigned to any of the servers belonging to the cloud when it comes to protecting the media or any personal or corporate information carried in RTP or RTCP packets. This is why the Media Switching Server in this use case is categorized as semi trusted. This deduction is in line with the discussion presented in Section 3.3 of [6].

With the above discussion of the trust model, it can be concluded that only the entities which are allowed to possess media encryption keys or keys that are used to provide end-to-end protection of information are considered trusted in this use case. Though it is not possible to guarantee that the trusted elements in this use case are never going to be compromised.

2.1.4 Solution

It is clear that the structure of the cloud conferencing demands for security solution which cannot be fulfilled with the current SRTP specification. As a solution, a meaningful modification of SRTP is needed with the separation of cryptographic contexts and keying materials used for end-to-end and hop-by-hop operations. The desired solution of key sharing is presented in Figure 6.

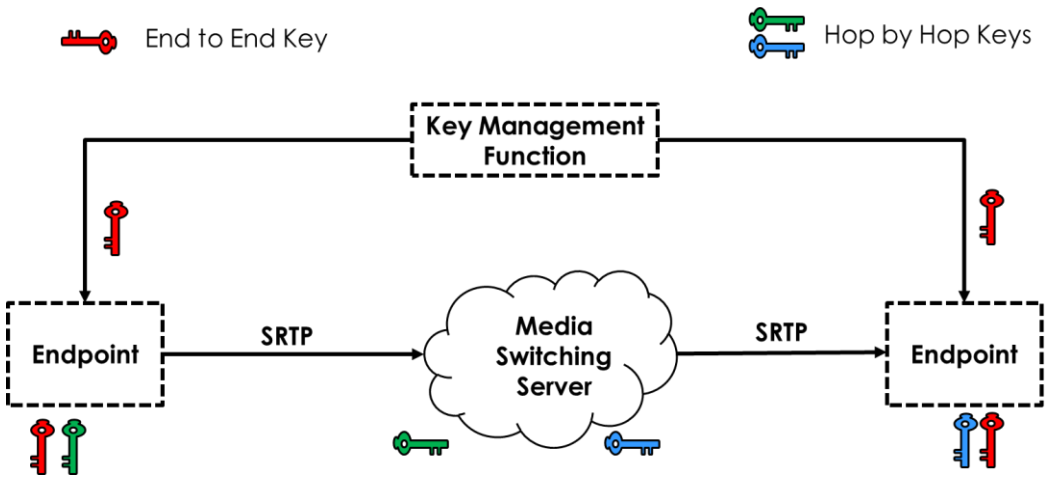


Figure 6. Separation of cryptographic contexts in modified SRTP

Instead of one master key, the solution proposes two. One is an end-to-end master key and another one is a hop-by-hop key. These two keys will have no association with each other. KMF is responsible for providing the end-to-end master keys to each of the Endpoints. The Endpoint derives its individual session key from the end-to-end one and encrypts the end-to-end protected part of SRTP packet with it. These keys are never shared with the Media Switching Server in the cloud. The Endpoint and the server share separate hop-by-hop master keys for hop-by-hop protection of the packets. Since the payload is protected with the end-to-end key, the server will not be able to read the content of it since it does not have access to that key. Only the intended recipient of the content, the receiving Endpoint has access to it and will be able to decrypt the media and play it.

2.1.5 Key Management

According to current SRTP, a specific cryptographic state information is maintained by sender and receiver in a SRTP stream. This is called cryptographic context [1]. This includes SRTP master key, cryptographic transform method and other necessary cryptographic information. The session keys are derived from the master key and they remain unique for each media stream. The negotiation of cryptographic contexts, delivery and derivation of encryption keys for a particular media session are managed by an external key management protocol. Example of such protocols are DTLS-SRTP, MIKEY etc. How DTLS-SRTP manages the key negotiation is described in Section 2.2.2.

The same SRTP master key for a particular media stream can be used to derive session keys for optional encryption of SRTCP packets and RTP header extensions belonging to the same session.

In the case of Cloud Conferencing, two unique master keys are introduced for each RTP stream. They are hop-by-hop and end-to-end master keys. The hop-by-hop key management is done as it is described in current key management protocols. These protocols are in need of modification in the event of the introduction of end-to-end keys which is discussed in later chapters.

2.1.6 Requirements

To design a detailed solution to meet the security needs for the mentioned cloud conference architecture and modify SRTP accordingly with minimum changes, the exact requirements of the solution need to be identified. Some of these requirements are from the IETF internet draft, 'Requirements for Private Media in a Switched Conferencing Environment' and some of these are based on the research performed for this thesis [4].

- End-to-end confidentiality and source authentication of media between the Endpoints must be ensured. On the other hand, the Media Switching Server should be allowed to change particular RTP header fields or header extensions to facilitate correct forwarding of the packets. [4]

- The solutions must support an architecture where the Endpoints do not need to address the other ones directly but are able to do that via a

centralized server, thus reducing resource consumption on part of the participants. [4]

- The solutions has to be supportive of cloud based environment where the conference instances are established in virtualized servers. [4]

- The end-to-end keying materials should never be accessible by the conference servers in the cloud or other unauthorized entities. [4]

- If the end-to-end security solution requires the use of certain RTP header fields namely SSRC (Synchronization source), Sequence Number and ROC (Rollover Counter), then these values need to be end-to-end authenticated too. [4]

Additional requirements proposed by this thesis are:

- Privacy and end-to-end authentication of information carried in RTCP packets and RTP header extensions must be provided if the information is considered in need of end-to-end protection.

- The solutions must support RTP topologies for multi-party conference mentioned in [10], particularly those which require the alteration of SSRC (Synchronization source) field of RTP headers.

- The server should be able to generate its own control packets (RTCP packets).

- The other SRTP security features should be present in the new solutions which are hop-by-hop replay protection and authentication of SRTP/SRTCP packets and optional encryption of RTP header extensions and SRTCP packets between the hops.

- With cloud conferencing, two unique SRTP master keys and protection profiles are introduced separately for end-to-end and hop-by-hop operations. Existing key management protocols must be able to support the negotiation of end-to-end security context.

2.2 Related Protocols

The protocols that have been studied in detail for designing the solutions for cloud conferencing are Secure Real-time Transport Protocol (SRTP), Encrypted Key Transport (EKT) and Datagram Transport Layer Security (DTLS) extension for SRTP. It is important to know the current specifications of these protocols, how they can be used or in need of modification for the proposals made in this thesis.

2.2.1 SRTP

The Secure Real-time Protocol [1] is defined as a profile of RTP [2] which is an extension to RTP Audio/Video Profile [11]. It provides confidentiality for RTP

and RTCP payloads. It also provides message authentication and replay protection to RTP and RTCP packets.

2.2.1.1 SRTP Packet Format

Figure 7 shows the format of RTP packet after it has been processed with SRTP security features. The figure is redrawn matching the one from Section 3.1 of [1].

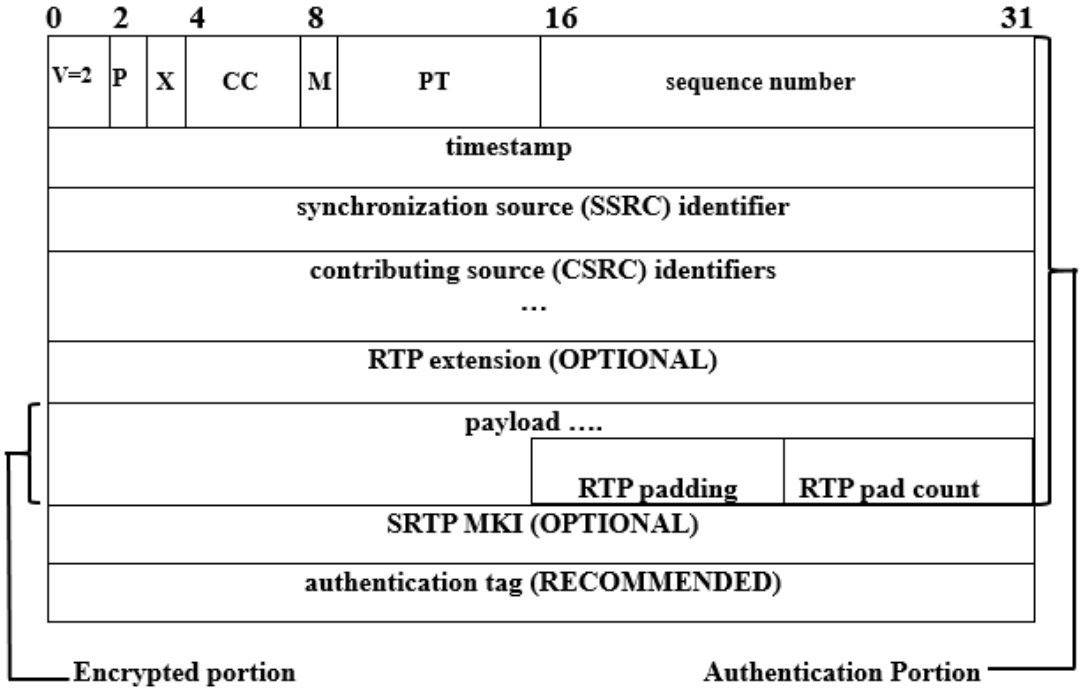


Figure 7. SRTP packet format

SSRC (synchronization source) is a random 32 bit value which identifies the source generating this particular SRTP packet. Sequence number is a 16 bit value which is incremented by one with each of the packet sent in a particular RTP stream [2].

The encrypted portion carries the media payload including padding if there is any. The authenticated portion carries the RTP header plus the encrypted portion of the RTP packet. Two new fields are added by SRTP to an RTP packet to facilitate the security features. These are MKI (Master Key Identifier) and Authentication Tag. MKI indicates the master key from which the session keys for encryption and authentication have been derived for this particular packet. Authentication Tag carries the keyed-hash of the message integrity data [1].

RTP header extension which is shown in Figure 7 as RTP extension is an optional part that carries application specific data with SRTP packet. According to SRTP specification, it is not encrypted but only authenticated.

2.2.1.2 SRTP security mechanisms

In SRTP, a single master key is used for protecting SRTP packets. Session keys for encryption and authentication are derived from this master key. The same master key may be used for protecting the SRTCP packets as well. But the session keys for SRTCP packets are derived separately [1].

There are per-defined cryptographic transforms mentioned in SRTP for use. These are Advanced Encryption Standard (AES) in Counter Mode (CM) and AES in f8-mode [1]. It is also possible to add new ones to this list of default transforms following the specification mentioned in SRTP. A new addition is an authenticated encryption transform which is AES in Galois/Counter Mode (GCM) [12]. It is an AEAD algorithm that encrypts a plaintext in Counter Mode and provides the authentication and integrity check of the resulted ciphertext using Galois Mode. It also provides the ability to authenticate some additional data without encryption.

Specially, mechanisms for encryption and authentication performed by AES-CM and AES-GCM are discussed in this thesis since these two transforms have been considered as default cryptographic transforms for the design of security solutions for cloud conferencing. The detailed description can be found from the respective RFCs mentioned in the references. In this report, only the parts that are important to understand the modification mentioned in the proposals are discussed.

- **Definition of IV (Initialization Vector)**
  Both the transforms use concatenation of session encryption key with an IV to generate the keystream blocks. The keystream blocks are Exclusiver-ORed with the plaintext to create the cipher blocks. The IV used by AES-CM is the Exclusiver-ORed product of SSRC of the RTP packet, the SRTP packet index and session encryption salt [1]. SRTP packet index is defined in [1] as:

  $$SRTP\ packet\ index = 2^{16} \times ROC\ + SEQ$$

  Here ROC is a 32 bit roll over counter that keeps record of how many times the 16 bit sequence number (SEQ) has rolled over to zero [1].

  IV generation of AES-GCM is shown in Figure 8. The figure is redrawn matching the one from [12].

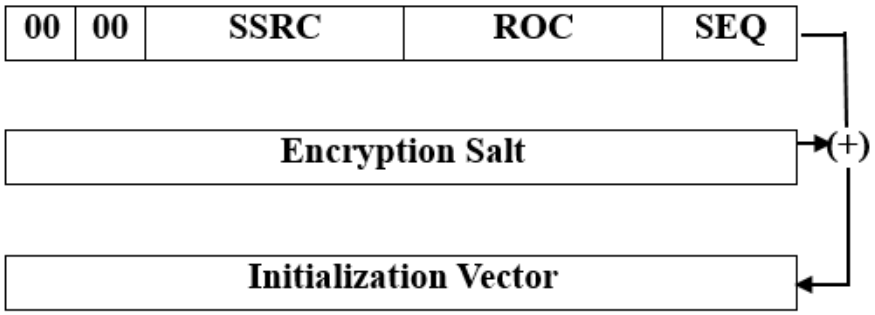


Figure 8. SRTP IV for AES-GCM

Firstly, for a particular SRTP packet, two octet of zeros, 32 bit SSRC of the packet, 48 bit ROC, and 16 bit sequence number of the same packet are added together. The result of this addition is Exclusiver-ORed with the 12 octet encryption salt to generate a unique 12 octet IV. The IV is used for encrypting the payload carried in that packet [12].

Inclusion of SSRC in calculating IV makes sure that the same SRTP master key can be used for multiple RTP streams belonging to the same session. This is why it is mandatory to keep the SSRC values unique for each of the sources in a RTP session [1].

To ensure the keystream generated for a particular packet is never repeated for other packets in the stream, packet index is included to calculate a unique IV for each packet. Keystream reuse can cause serious compromise of security. Therefore, to provide maximum security, the uniqueness of SSRC for each source in a session and a unique combination of ROC and sequence number for each packet in a stream must be maintained when a single counter mode key is used for protecting the entire RTP session [1].

- **Authenticated Encryption**
  Using AES-GCM, SRTP payload is both authenticated and encrypted at the same time. All the fields from RTP header are authenticated by adding them in Associated Data [12]. Figure 9 from [12] shows the classification of how an SRTP packet is secured using AES-GCM.

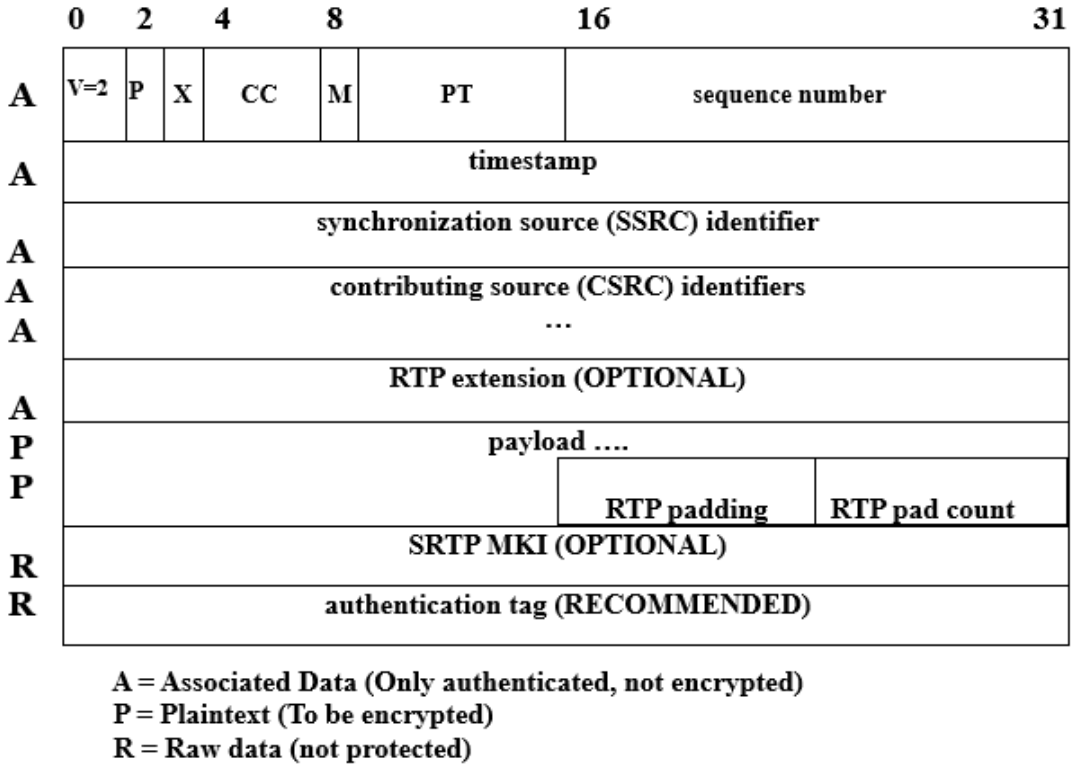


Figure 9. SRTP packet security provided by AES-GCM

- **Message Integrity**
  Integrity and authenticity of the RTP packet is confirmed checking the authentication tag carried in SRTP packet. Using pre-defined SRTP hash algorithm, HMAC-SHA1, the authentication tag is produced. The tag is the hashed output of concatenation of the 'authenticated portion' (shown in Figure 9) and ROC [1]. If AES-GCM is used then authenticate tag becomes a redundant security feature and is avoided. Moreover, the session authentication key is not used when AES-GCM is chosen as part of the SRTP protection profile [12].

- **Replay Protection**
  As mentioned in Section 3.3.2 of [1], if a received SRTP packet passes authentication check, then the packet index of that particular packet is logged in the Replay List. This is how replay protection is maintained in SRTP.

Protection mechanism of SRTCP packets are part of SRTP specification too. The default security profiles for SRTCP are as same as those for SRTP. The security mechanisms are similar too. But there are slight variations which are mentioned in detail in Section 3.1.1.

2.2.2 DTLS-SRTP

Datagram Transport Layer Security (DTLS) provides channel security for establishing keys, negotiation of parameters and protection of application data. It is based on TLS (Transport Layer Security) [13]. Unlike TLS which provides the security of TCP channels, DTLS is designed to support security of unreliable

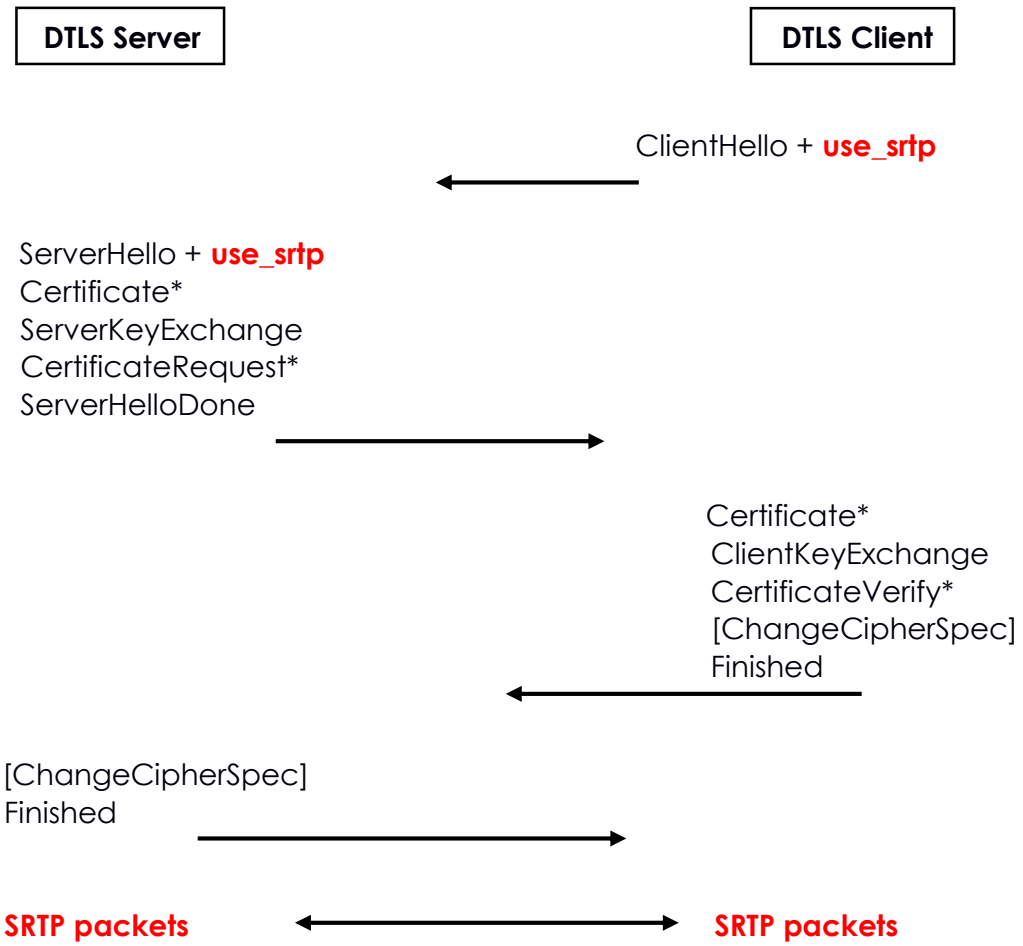
transport channels of UDP [13]. DTLS-SRTP introduces a particular extension to DTLS for establishing SRTP and SRTCP keying materials [3]. It is also used for negotiating algorithms and other parameters of cryptographic contexts belonging to individual RTP sources. Overall DTLS-SRTP takes the advantage of secure key management method of DTLS but uses the security features of SRTP to consider the secure RTP packets as a new data format for DTLS to transmit over its established media paths [3].

For an RTP media session between two entities, an out of band signalling is used to decide which one is DTLS server and which one is DTLS client. Generally, one pair of source and destination port is used for each RTP and RTCP flow. Each of this flow is bound to a single DTLS association resulting in a single SRTP context to be used in each direction. But it is also possible to multiplex both RTP and RTCP flows in the same UDP port. In this case, both SRTP and SRTCP packets are received over the same DTLS session [3].

To initiate a DTLS channel between a client and a server, a particular message known as 'hello message' is sent. Hello message contains various information including session ID and encryption algorithms supported by client or server. Server or client hellos are part of the handshake protocol of standard TLS. Handshake protocol is the first step of communication between a TLS client and server to decide on a protocol version, choose cryptographic algorithms, authenticate each other (optional) and generate shared secrets that are going to be used later to protect the exchange of application layer packets. [20]

2.2.2.1 Use of the extension

A particular extension called 'use-srtp' is introduced to be used by DTLS peers who want to exchange SRTP packets. This extension is added with the extended version of client hello which lets the server know that the client wants to negotiate SRTP security parameters to send and receive SRTP/SRTCP packets with the server. The data field of 'use-srtp' contains all the SRTP security profiles (algorithms and related parameters) that the client can support. If the server is willing to use SRTP will select the protection profile it can support and send the information with a server hello extended with the same extension – 'use-srtp' [3]. The exchange of DTLS handshake messages that take place when negotiating SRTP security contexts is shown below in Figure 10. The figure is redrawn matching the one from [3].



'*' indicated DTLS messages are not always sent, but for DTLS-SRTP client and server Certificates, CertificateRequest and CertificateVerify will be sent.

Figure 10. Handshake message flows of DTLS-SRTP

The structure of extended message for 'use-srtp' is shown below.

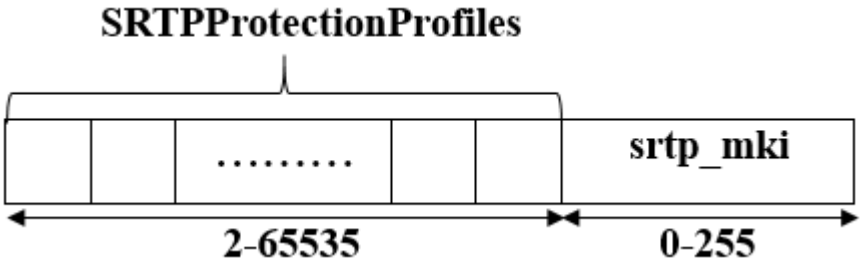


Figure 11. The message format of hello extension, 'use_srtp'. The size of the fields are in bytes.

The data field of the extension, "use_srtp" carries two separate data types. They are SRTPProtectionProfiles and srtp_mki. SRTPProtectionProfiles is the list that contains all the protection profiles that the client or server can support. Each profile has a particular number assigned to it. These number are entered as two bytes-value in the data stream. The use_srtp data contains the values of the chosen or supported profiles from the list. The pre-defined profiles and their values are mentioned in Section 4.1.2 of [3]. SRTPProtectionProfiles can never be an empty field. The srtp_mki holds the MKI value for indicating the SRTP master key that the DTLS peer is going to use. The byte size "0-255" indicates that the length of MKI can be between zero byte to 255 bytes. It is zero when no MKI is sent as this is an optional field for SRTP packets.

2.2.2.2 Key Derivation

For protecting SRTP packets, the DTLS keys for protecting application data are not used. Instead, DTLS keying material is used to derive SRTP master keys and salts. Next these SRTP master keys and salts are fed into the key derivation function mentioned in RFC 3711 and turned into session keys for each direction [1], [3]. The client session keys are used for encrypting and authenticating SRTP packets sent by the client and the session keys for server are used for the same purpose for SRTP packets sent by the server [3]. Both client and server will share these keys to decrypt the received packets. Same procedure is followed for deriving SRTCP keys by using its designated key derivation parameter mentioned in RFC 3711 [1].

An important thing to note about DTLS-SRTP is that all the RTP sources belonging to a particular DTLS peer uses the same SRTP master key. This is why it is important to keep the SSRC(s) of each of these sources unique to avoid the possibility of occurrence of same key stream being used by multiple RTP streams [3].

2.2.2.3 Transmission and Reception

When DTLS-SRTP is in use, meaning when the session is established exchanging the extension 'use-srtp', client's data from application layer which should be complete RTP or RTCP packets are never sent to DTLS stack but are

processed through SRTP stack. These packets are not considered as typical DTLS record data to be protected with DTLS protection mechanism, instead the implementation encrypts and authenticates the packets using the SRTP session keys derived from DTLS session following the specification of SRTP. Next, the packets are transmitted over the DTLS channel to the destination port. On the server side, they are detected to be RTP or RTCP packet by inspecting the first byte of the packet which should be between 128 and 191 (inclusive) [3]. Then these packets are authenticated and decrypted accordingly.

2.2.2.4 Use of DTLS-SRTP in Multi-party Conference

DTLS is suitable for point to point communication. For a conference with multiple participants, a point to point communication between each of the participants is not scalable. A central media forwarder or typical RTP mixer can be used to hold DTLS sessions with each of the participant to receive and forward media and control packets [3].

The existing DTLS-SRTP mechanism is not compatible with cloud based conferences where the media encryption keys need to be separated from the hop-by-hop keys. Moreover these media keys or end-to-end keys are distributed by a key management function which do not use the keys itself whereas in the current specification of DTLS-SRTP, the keys for both the peers are derived during the session and used in exchanging SRTP packets.

2.2.3 EKT

Encrypted Key Transport (EKT) protocol adds an extension to SRTP for secure transport of SRTP master key and current value of Rollover Counter (ROC) attached in a SRTP or SRTCP packet to each SRTP source (SSRC) during a live RTP session. The specification of this protocol has been proposed with an internet draft which work is still in progress [14].

2.2.3.1 Usefulness of EKT

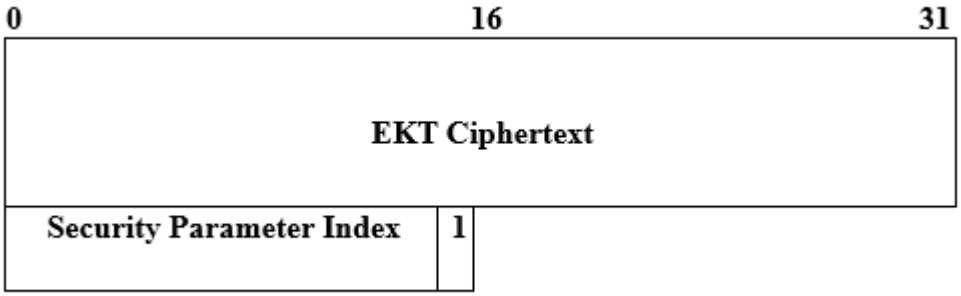
SRTP implementation requires strict coordination of SSRC values and timely distribution of ROC and SRTP keys between the participants. Usually through a separate key management protocol and signalling system, the participants are updated with these information. Use of EKT minimizes the overhead of such central control and complexities of maintaining coordination with multiple participants. As EKT facilitates the transport of SRTP master key, values of SSRC and current ROC with SRTP packet, it provides any RTP source to choose any SSRC value it likes. Use of EKT removes the restriction of keeping SSRC values unique within the session as it provides the advantage of using separate SRTP master keys for each stream. With EKT, the sources can change their master keys and transport them with packets they are encrypted with in a live RTP stream. Using EKT, a participant can start a new RTP source or replace an old one with any SSRC value and let the other sources know by attaching EKT field in SRTP packet. On the other hand, newly joined sources are easily informed of the current values without any need of external signalling [14].

2.2.3.2 EKT Field Formats

EKT provides a particular key, called an EKT key which encrypts the SRTP master key and sends the ciphertext with selected SRTCP or SRTP packets. For a particular RTP source, the format of the plaintext is concatenation of its SRTP master key with its SSRC value, current value of ROC and a new value called Initial Sequence Number (ISN). It indicates the sequence number of the initial RTP packet that has been protected using this SRTP master key. If the value of ISN is zero, then it indicates that the sequence number of the initial RTP packet that has been protected with this SRTP master key belongs to the last roll over of ROC [14].

The generated ciphertext and Security Parameter Index (SPI) is added as a separate field at the end of an SRTP or SRTCP packet. It is known as the Full EKT Field. SPI indicates a value for identifying the EKT key, EKT cipher to decrypt the full EKT field, key management parameters mentioned in Section 8.2 RFC 3711 , SSRC value carried in the packet and associated SRTP or SRTCP master salt [14], [1].

There is a second format of EKT field which is the Short EKT Field. It does not carry the EKT key or other parameters but a 'Reserved' field of 7 zero bits which are ignored on reception [14]. The formats are shown in Figures 12 and 13. The figures are redrawn matching the ones from [14].



$$\boldsymbol{EKT\_Ciphertext} = \boldsymbol{EKT\_Encrypt}\,(\boldsymbol{EKT\_key}, \boldsymbol{EKT\_Plaintext})$$
$$\boldsymbol{EKT_{Plaintext}} = \boldsymbol{SRTP\_Master\_key} \parallel \boldsymbol{SSRC} \parallel \boldsymbol{ROC} \parallel \boldsymbol{ISN}$$

Figure 12. Format of Full EKT Field

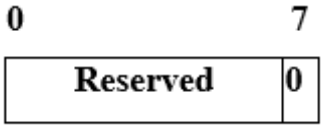


Figure 13. Format of Short EKT Field

As a final bit, '1' is added at the end of the full EKT field and '0' is added at the end of the short EKT field to distinguish between the two formats [14].

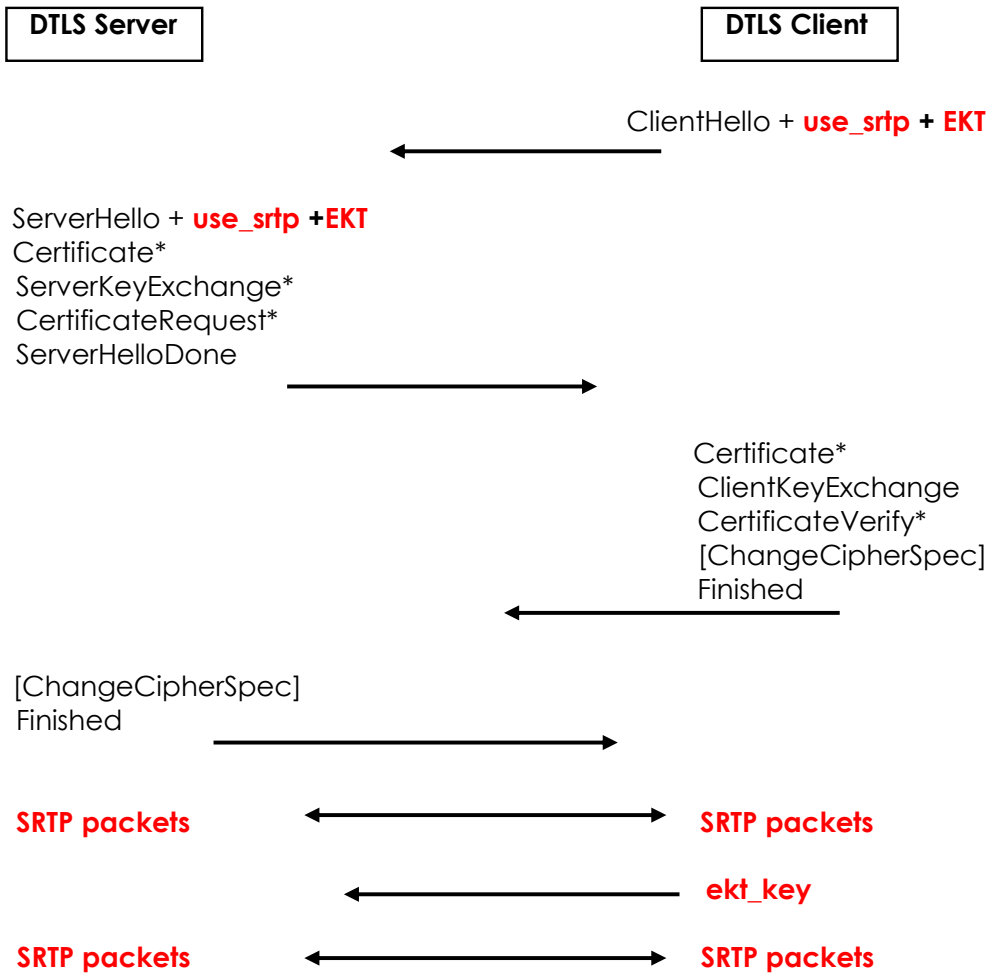
A full EKT field with SRTP or SRTCP packet is sent in the event of a new sender joining a session, changing of SRTP master key or changing of EKT key. Existing senders should also transmit full EKT fields in intervals and the frequency of it depends on how often new receivers are joining the session [14].

2.2.3.3 Use of EKT with DTLS-SRTP

DTLS-SRTP causes all the sources generating from a particular SRTP entity to use the same SRTP master key. SSRC values of these sources have to remain unique to avoid keystream reuse. Use of EKT with DTLS-SRTP frees the entity from using the same DTLS generated key for all the sources and enables each source to use and transport a separate key with its SRTP stream. Also assignment of SSRC of the sources do not need to be tightly controlled if EKT is used in conjunction with DTLS-SRTP.

To use EKT with DTLS-SRTP, an extension called Key Transport has been defined in the IETF draft. The extension enables EKT keying material to be transported from one DTLS peer to another using the secure channel provided by DTLS. The draft includes a new TLS content type called EKT and a new TLS negotiated extension of the same name. The extension should only be used with the other extension 'use-srtp' from RFC 5764 [14], [3].

Before DTLS handshake takes place, through external signalling such as SDP (Session Description Protocol), the wish to use EKT with DTLS-SRTP is established mutually between the server and the client. Acknowledged with external signalling, client sends a ClientHello message with the extension 'ekt' and server replies the same with its ServerHello [14]. The handshake message flow for negotiating EKT is shown in Figure 14. The figure is redrawn matching the one from [14].



'*' indicated DTLS messages are not always sent, but for DTLS-SRTP client and server Certificates, CertificateRequest and CertificateVerify will be sent.

Figure 14. Handshake message flows of DTLS-SRTP with EKT

After the successful negotiation of EKT in DTLS-SRTP handshake, the client sends EKT keying material information using 'EKT' content type. The actual EKT key is sent with the 'ekt_key' message. The client and server may keep exchanging SRTP packets protected using DTLS-SRTP derived master keys though they have negotiated to use EKT key. It also may happen they start sending packets with full EKT fields right away [14].

2.2.3.4 Use of EKT with cloud conference

The use of double key in cloud conference makes the transport of end-to-end key with DTLS-SRTP incompatible. Moreover, this key needs to be sent by a central key management server to all the Endpoints using a secure channel. Whereas DTLS-SRTP is not suitable for multi-party conference, EKT is very useful to be used in conferences with multiple participants. In cloud conferencing architecture, EKT key can be used as a common key to be sent to all the participants taking part in the conference by the Key Management Function (KMF). After receiving the EKT key from KMF, each source in the

conference can encrypt its individual end-to-end SRTP master key using the EKT key. A source can send the encrypted result directly with SRTP/SRTCP packets protected by the same master key. This way, the Endpoints do not have to have multiple DTLS-SRTP sessions with each other to negotiate end-to-end SRTP security contexts and keying materials. The use of EKT makes the end-to-end key distribution scalable and secure.

2.2.3.5 Updates of EKT

In IETF 92 meeting, it has been presented that use of ISN in EKT is not compatible with SRTCP. In that case, MKI can replace the use of ISN to manage multiple keys per SSRC [14]. It can remain optional as it is presently in RFC 3711 [1]. In the recent draft, use of MKI is not allowed since it duplicates some of the function of EKT. This problem and other changes will be taken care of and updated in the next draft.

2.3 Related Work

Reflecting on the discussion of problems of applying SRTP in cloud based conference and its security requirements, a general solution of having a double layer of security has been considered. Two proposals of solution frameworks by Cisco and Ericsson have been made based on these requirements. Both of these frameworks are works in progress in the form of IETF internet drafts. In this thesis, details of these solutions are presented and analysed. Analyses are performed by finding out how compatible these frameworks are with current standardization, protocols and the requirements of cloud conferences discussed above. Additionally, the security features that have been found useful from the analysis have been used in the designs proposed in this thesis.

2.3.1 Proposal by Cisco

The solution framework proposed by Cisco Systems focuses on privacy of media carried in SRTP payloads and cryptographic separation of the keys used for media privacy and integrity by making them inaccessible for Media Switching Server [7]. This has been tried to achieve with minimal changes in current SRTP specification. The structure and trust model of the use case considered in this framework is similar to the ones discussed earlier in this thesis in Sections 2.1.1 and 2.1.3.

2.3.1.1 Security Features

The framework provides the following security features as mentioned in [7].

- SRTP payload is encrypted and authenticated with a media encryption key which is shared between the trusted Endpoints (receiver and sender of the payload). The conference server in the cloud does not have access to it.
- The Media Switching Server is not able to decrypt or authenticate the media but is capable of modifying certain RTP headers.
- The Media Switching Server is permitted to include, exclude, encrypt and decrypt RTP header extensions.

- The server is allowed to generate RTCP packets and encrypt, decrypt and authenticate all SRTCP packets in the conference.
- Replay protection of each SRTP and SRTCP packet is ensured by making authentication mandatory between each Endpoint and the conference server with a unique key shared only by them. These keys must not have any relation to the media encryption and authentication keys.

2.3.1.2 Packet Format

The modified packet format of SRTP proposed by this framework is shown in Figure 15. The figure is redrawn matching the one from [7].

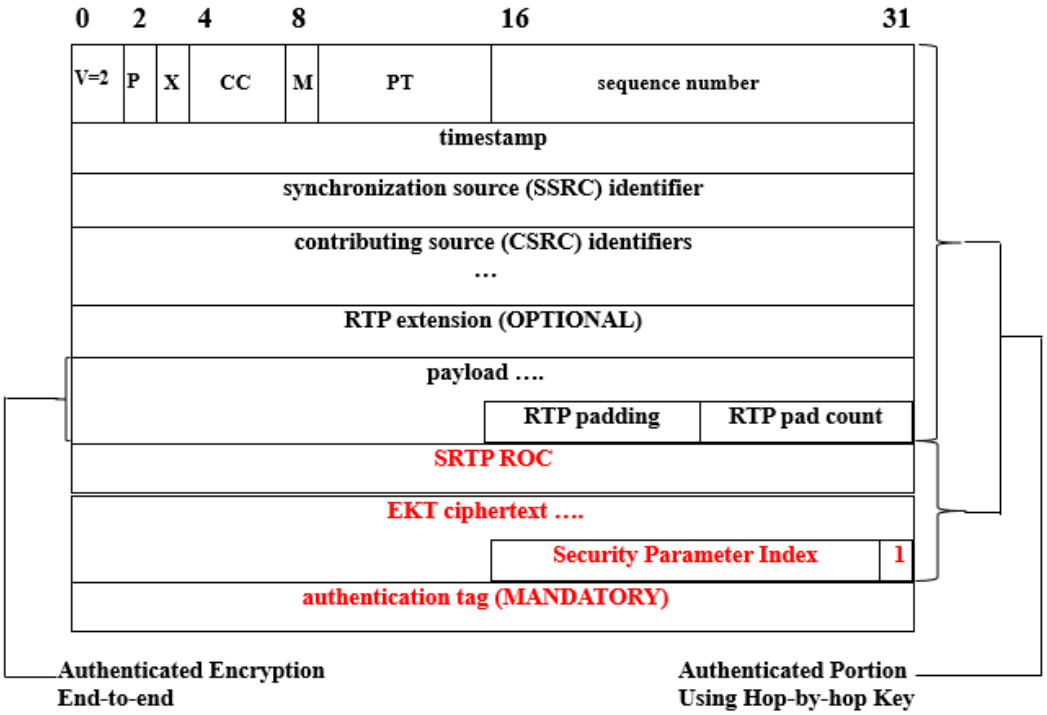


Figure 15. Modified SRTP packet proposed by Cisco

The RTP payload is encrypted and authenticated end-to-end with an authenticated encryption algorithm using an end-to-end SRTP master key. The entire packet up to the Authentication Tag is mandatorily authenticated between each hop using a hop-by-hop key [7]. This is why the attachment of the authentication tag has been made compulsory.

Encrypted with an EKT key, the end-to-end SRTP master key is transported with SRTP packet. The receiving Endpoint uses this master key to derive the encryption key with which this particular payload has been protected end-to-end. Figure 15 shows the 'Full EKT Field'. According to the framework, the full field will be transmitted every 100 milliseconds. 'Short EKT Field' is used in other times. The EKT key may be changed in the event of a participant leaving or joining a conference [7].

The framework proposes that ROC is sent as a plaintext with the packet. The motivation behind this modification, according to this framework, is to let the Endpoint know of the value of ROC of a sender when Media Switching Server performs selective forwarding based on audio level indication. If the sender has been silent for a long time before speaking again, it may happen that the ongoing sequence number of SRTP packets sent by this particular sender has been rolled over. Also according to the framework, the receiving server or Endpoint is able to generate a correct authentication tag by reading the value of ROC from the packet since this value is part of the calculation of an SRTP authentication tag as mentioned in Section 2.2.1.2[7].

2.3.1.3 End-to-end Operation

A SRTP master key generated by an Endpoint is used to derive the session encryption key according to key derivation mechanism mentioned in RFC 3711 [1]. This mechanism also produces an authentication key. But this key is ignored for end-to-end operation as only the session encryption key is used by an authenticated encryption algorithm for both encryption and authentication [7]. The session encryption key or the end-to-end media key is used to encrypt and authenticate RTP payload using the transforms defined in [12]. The procedure mentioned in this draft is followed for protecting the payload except for a change in the content of Associated Data shown in Figure 9. According to the framework proposed by Cisco [7], the content changes to:

Associated Data: version (V), padding Flag (P), sequence number, timestamp and SSRC

The above RTP header fields are authenticated end-to-end but not encrypted. The other header fields including RTP header extension (optional) remain free to be modified by the Switching Server. According to this framework, this allows the server to make needed changes in the header for smooth transition of SRTP packets, for example changing payload type (PT) value or access voice level indication data from RTP header extension etc. [7].

The end-to-end master key, responsible for protecting the payload, is securely transported with the same SRTP packet encrypted by the EKT key. The knowledge of this key is only shared with trusted Endpoints who are allowed to decrypt the payload.

2.3.1.4 Hop-by-hop Operation

The entire packet is authenticated with a unique key shared between each Endpoint and the Media Switching Server in the conference and an authentication tag is attached at the end of the packet. If there are cascaded servers, then they also share unique keys between each of them. Therefore, if the server makes any changes in RTP headers, the changes are authenticated at each hop. The authentication key used for this hop-by-hop authentication is derived from the hop-by-hop master key. The key derivation, the transforms used and the process of authentication follow the specification mentioned in RFC 3711 [1]. Other standardized transforms may also be used [7].

Encryption of RTP header extension is possible following RFC 6904 [16]. The Cisco framework specifies that an encryption key is derived from SRTP hop-by-hop master key if the extensions are confidential [7]. This allows the Media Switching Server to decrypt and re-encrypt them.

RTCP packets are mandatorily hop-by-hop authenticated. This allows the Media Switching Server to generate its own SRTCP packets or collect SRTCP packets from other Endpoints and create compound packets. This way, the server is able to generate conference control information and forward them in a secure way. These RTCP packets can be optionally encrypted if needed. The SRTCP session encryption keys are derived from the same SRTP hop-by-hop master key. Authentication and optional encryption mechanism and the cryptographic transforms used for these operations are followed from RFC 3711 [1]. Other standardized transforms may also be used [7].

The optional encryption key derivation procedure for RTP header extension and SRTCP also follow the specification of RFC 3711 [1], [7].

Mandatory hop-by-hop authentication of SRTCP and SRTP packets mentioned in this framework ensure replay protection of the packets at each hop [7].

2.3.1.5 Key Management

A key management protocol is needed to deliver EKT keys from Key Management function (KMF) and negotiate end-to-end SRTP protection profile. The Cisco framework suggests the establishment of a tunnelled DTLS-SRTP session between two trusted entities – Endpoint and KMF – through the untrusted Media Switching Server. This framework categorises Media Switching Server as an untrusted entity. This tunnelled session is used to send EKT keys from KMF to each of the Endpoints in the conference. Media Switching Server's job, in this case, is just to forward these DTLS messages from Endpoint to the KMF [7].

Some form of external signalling is used to negotiate a participant identifier for each of the Endpoint with the KMF. KMF keeps track of the DTLS connection to a particular Endpoint with its participant identifier. How to negotiate end-to-end SRTP context is not clearly specified in this version of the Cisco draft but an extension to EKT or some other means have been suggested. The framework also calls for an extension to EKT to send the participant identifier. The draft suggested modification of EKT to send ROC as plaintext instead of sending it with the EKT ciphertext [7].

About the negotiation of hop-by-hop SRTP master key and security context, the framework suggests using the above tunnelled DTLS-SRTP to settle the security profile and some extension to make KMF deliver the unique hop-by-hop keys to each of the Endpoints and the server [7].

2.3.1.6 Analysis of the Framework

An analysis of the security features proposed in the solution framework by Cisco has been done based on their compatibility with current RTP and SRTP

standards and also the requirements of cloud based conference mentioned in Section 2.1.5. The analysis has been discussed below.

- **Large sequence number gaps**
  If selective forwarding based on audio level indication is implemented at the Media Switching Server, the server only forwards those SRTP packets that are coming from active speakers and discard the silent ones. According to this framework, the server is not allowed to change the sequence number of the packets. Therefore, at the receiver's end, it may seem that there has been a packet loss.

- **End-to-end protection of RTP header extensions and SRTCP packets**
  SRTCP packets and RTP header extensions may carry data that need end-to-end protection, but solution from Cisco does not consider any such possibility for the present or future needs.

- **No topology definition**
  The solution framework does not specify which RTP topologies [10], the framework is compatible with. There are some RTP topologies with implementation in the Switching Server that may require it to have its own synchronizing source identifier (SSRC) and replace the incoming packet's SSRC with its own. In another topology, the server needs to remap an incoming SSRC with an outgoing one for a particular source and destination RTP source pair. The framework does not support rewriting of SSRC, thus makes it incompatible with such implementations.

- **CS and CSRC are not end-to-end protected**
  According to RTP, if an Endpoint generates a payload with mixed audio or video data coming from different sources, it lists all the sources in CSRC (the contributing source identifier) field of the RTP header. The field CS (CSRC count) holds the number of sources mentioned in CSRC field [2]. In such case, both CSRC and CC need to be included in end-to-end Associated Data which is not done in the mentioned framework.

- **Re-keying EKT**
  According to the framework, EKT keys are changed when new participant joins or old participant leaves an ongoing conference. If the number of participants is quite high and the nature of the conference is very dynamic, then re-keying every so often can become too complicated in terms of complexity and bandwidth consumption. The future drafts of the framework should consider a solution to this problem.

- **Sending ROC**
  Sending ROC as a plaintext with SRTP packet is redundant since the Endpoint can be informed of the changed value by the transmission of 'Full EKT Field'. To coordinate ROC values between the RTP sources is one of the purposes of using EKT [14]. In case of the server, it does not have access to EKT ciphertext. But according to SRTP specification, the

server is meant to keep track of the sequence numbers of authenticated packets received from each of the participant (discarded or forwarded) to maintain a Replay List [1]. This is how it is always able to calculate correct value of ROC for a particular source. Therefore, changing the format for EKT proposed by the framework is also unnecessary.

- **Key management and DTLS-SRTP tunnelling**
  The key management specification for distributing end-to-end and hop-by-hop cryptographic materials are not clear in this draft. The details for the tunnelling protocol is not submitted. There is no clear indication how the participant identifier is exchanged during DTLS-SRTP-EKT session and by which entity. The negotiation of end-to-end security profile and also transfer of hop-by-hop keying material are not clearly mentioned. Since, this is a 'work-in-progress' draft, these limitations of the current design have been listed to be decided in future versions [7].

2.3.1.7 Updated Framework

During the documentation of this thesis report, another version of Cisco solution framework has been submitted [15]. Since this draft appeared after the completion of the tasks for the thesis, only important highlights and a short analysis of the changes are discussed below.

- **Insertion of MKI**
  To realize the change in EKT mentioned in Section 2.2.3.5 of replacing ISN with MKI, this version of framework adds two new fields in SRTP packet format. These are – 'MKI for End-to-End' and 'SRTP MKI (Optional) for Hop-By-Hop' [15]. Inclusion of the last field follows the original design of SRTP packet from RFC 3711 [1].

- **Change in Associated Data**
  The content of the associated data has been changed to include CSRC count (CC), marker (M) and contributing source identifier (CSRC) [15]. The inclusion of CC and CSRC fields reflects the analysis made in this thesis in the previous Section.

- **End-to-end IV (EEIV) in RTP header extension**
  There is a new proposal of sending a value called EEIV in RTP header extension in the event of any changes made to sequence number and SSRC by the switching server [15]. Because these values are used by the Endpoint to calculate the end-to-end IV. The format of EEIV according to [15] is:

$$EEIV = SSRC \parallel SEQ$$

  This allows the server to change SSRC if needed by certain RTP implementation and also sequence number. Server not being able to change these values was a limitation of the previous draft as the consequences of it is pointed out in the previous analysis. But this

inclusion of EEIV in RTP header extension does not receive any end-to-end protection since the server is able to access it. But any parameter related to end-to-end cipher calculation should also be end-to-end protected. Moreover, the inclusion of sequence number in EEIV may introduce two counts of 'sequence number' at the Endpoint which is a violation of existing design of RTP.

- **Hop-by-hop key management and DTLS-SRTP tunnelling**
  In this draft, the tunnelling protocol has been revealed as exchange of DTLS messages encapsulated in a second DTLS message between the KMF and Media Switching Server. It is also mentioned that the same DTLS-SRTP connection will be used by the KMF to deliver the hop-by-hop SRTP master key, salt and security context to the server in the cloud [15]. Whether the Endpoints receive their hop-by-hop keying material and security context in the same way is not mentioned.

2.3.2 Proposal by Ericsson

A proposal from Ericsson in IETF provides a solution for a use case similar to the one for this thesis. The use case by Ericsson offers solution for communication with a cloud based conference server or middlebox that handles instantaneous communication or store-and-forward communication for pre-encrypted media [6]. The trust model for this use case is exactly the same as the one mentioned in this thesis in Section 2.1.3. This solution also proposes separation of cryptographic contexts with end-to-end and hop-by-hop keys. The interesting difference between this solution and the solution proposed by Cisco is that the Ericsson solution provides features to make the confidentiality and authentication of media not depend on any of the fields from RTP header.

2.3.2.1 Security Features

The framework from [6] provides the following security features.

- Media protection is ensured which mechanism does not rely on RTP header fields.
- Media protection includes confidentiality and message authentication.
- Source authentication of end-to-end media is provided.
- The solution supports retransmission of cached protected media and random seek and playback between the media payloads.
- The entire SRTP packet is authenticated and replay protected between each hop. The end-to-end protected part can be protected with hop-by-hop encryption key if needed.
- RTCP packets are only authenticated and replay protected from hop to hop.
- The keying materials and security context for end-to-end and hop-by-hop operations are absolutely separated and independent from each other.

The solution from Ericsson contains detailed description of different features of the design proposed by it. Only the features and the details that are relevant to the tasks of this thesis are explained in the following Sections.

2.3.2.2 Packet Format

The modified packet format of SRTP proposed by this framework is shown in Figure 16. The figure is redrawn matching the one from [6].

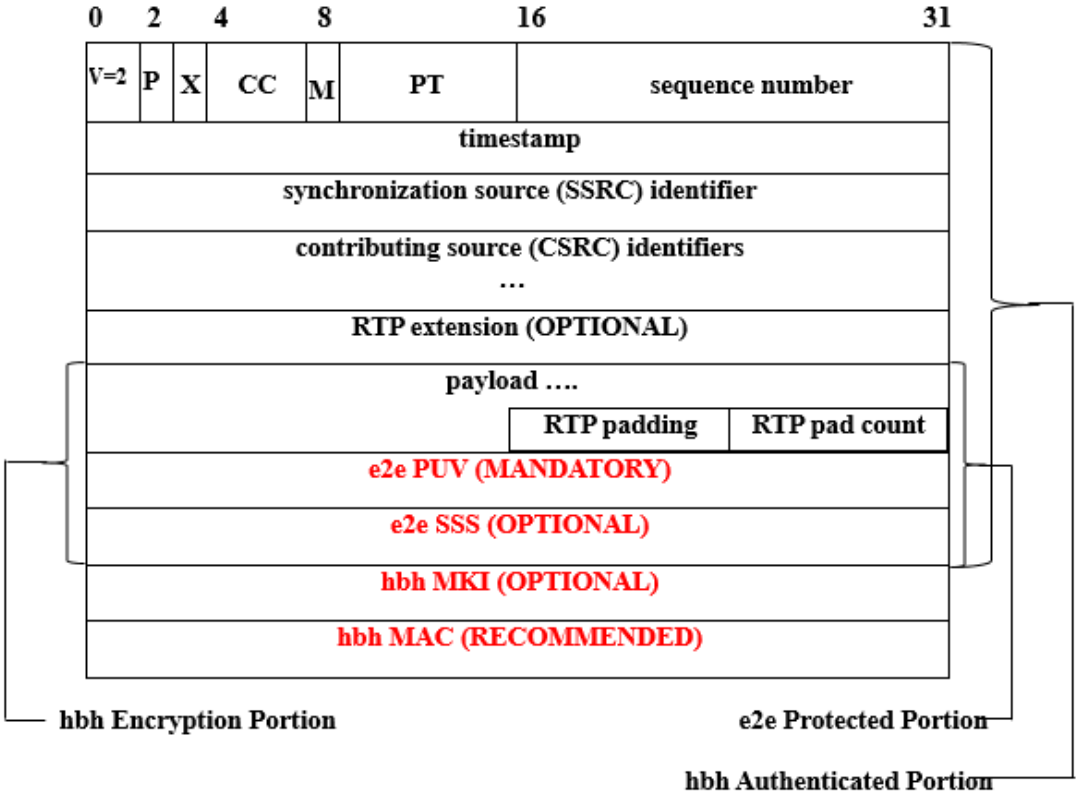


Figure 16. Modified SRTP packet proposed by Ericsson

The portion of RTP packet that requires end-to-end protection is clearly identified in the format above. It includes two new fields proposed by this solution which are Packet Unique Value (PUV) and SRTP source (SSS). End-to-end cryptographic transform is dependent on these values. RTP payload, padding and pad count are both authenticated and encrypted. PUV, SSS and padding flag (P) are only authenticated end-to-end. Description of PUV and SSS are given below.

- **PUV**
  Inclusion of packet unique value in the end-to-end part is mandatory. It is of configurable length. Its function is similar to 'sequence number' field of an RTP packet. It can be considered as a counter for the end-to-end protected part that starts with the value zero and is incremented by one for each packet. To prevent the possibility of generating same keystream for encrypting the payloads in a particular RTP stream, PUV(s) need to be kept unique [6].

- **SSS**
  SRTP source is an identifier of the RTP source which creates this particular end-to-end portion. Its function is similar to SSRC but different in a sense that it only identifies the source of the end-to-end part of the packet. Like SSRC, it also needs to be unique between the sources if the same end-to-end SRTP master key is used by all the sources in a session. Its use is optional and size is of configurable length [6].

The field 'Hbh MAC' functions exactly the same way authentication tag does for the existing SRTP packet from Figure 7. The definition of Hbh MKI is also same as the MKI field found in RFC 3711 [1].

The end-to-end protected part here is considered as hop-by-hop encrypted portion to match with the secure RTP packet format from RFC 3711 [1]. Therefore, the entire packet can be compared with a typical SRTP packet which is authenticated from the start of the packet up to 'hbh MKI' field between each hop according to RFC 3711 [1] , [6].

2.3.2.3 End-to-end operation

The end-to-end media protection is secured by end-to-end keying material based on end-to-end master key which is completely independent of the hop-by-hop master key. The derivation of end-to-end session key may follow the procedure mentioned in RFC3711 [1]. The default transform to be used for end-to-end protection is AES-GCM specified in [12]. The significant design choice for this framework is to enable the end-to-end media protection entirely be independent of RTP header. This has been made possible with the use of the fields PUV and SSS instead of sequence number and SSSRC respectively for calculation of end-to-end IV [6]. It changes the IV formation mentioned in [12] when AES-GCM is used for end-to-end cryptographic transform. The second change from that specification is the redefinition of the content of associated data. According to the framework proposed by Ericsson [6], the content changes to:

Associated Data: padding Flag (P), PUV and SSS

The processing of end-to-end protection starts with the derivation of session encryption key from the end-to-end SRTP master key. Next, the key is used with AES-GCM transform to encrypt and authenticate the payload and associated padding and pad count. Since, PUV and SSS are directly related to the process of end-to-end encryption, they are end-to-end authenticated as well by including them in the Associated Data [6]. Concatenation of all these fields in the mentioned order, produces the end-to-end protected part of a SRTP packet shown in Figure 16.

2.3.2.4 Hop-by-hop operation

Except for padding flag, all the other RTP header fields are considered as hop-by-hop values that can be changed by the switching server or the trusted Endpoint [6]. The session key for hop-by-hop protection is derived from the hop-by-hop master key following the procedure mentioned in RFC 3711 [1]. The default hop-by-hop encryption algorithm is NULL encryption and authentication is done using HMAC-SHA1 [6]. The processing of hob-by-hop operation is identical to the one described in SRTP specification [1].

SRTCP packets for the cloud conference are generated with hop-by-hop security as mentioned in the current specification. Replay protection remains the same as it is in RFC 3711 which is done at each hop with hop-by-hop authentication of SRTP packets and keeping track of corresponding sequence numbers [1], [6].

2.3.2.5 Key management

No specific key management protocol or procedure has been described in this solution to cater the modifications it has proposed. But the solution does mention that the separation and secrecy for the transport of end-to-end keying material should be provided by the key management protocol, whichever it is [6]. The solution also provides details on how to differentiate security context identification for end-to-end and hop-by-hop operations. These details are skipped since they are not directly related to the tasks of this thesis.

2.3.2.6 Analysis of the Framework

An analysis of the solution proposed by Ericsson has been conducted based on its application to cloud conference architecture. The analysis is presented below.

- **End-to-end protection of RTP header extension and SRTCP packets**
  This design [6], like the one from Cisco [7], also does not propose any solution to protect any data belonging to RTP header extension and SRTCP packets that may need end-to-end security. It does not even mention any protection mechanism needed for keeping RTP header extension confidential, hop-by-hop or end-to-end. In case of RTCP packets, the solution from Ericsson specifies them to have just hop-by-hop protection, since they are going to be used for conference monitoring and traffic controlling between the hops by Media Switching Servers. But it does mention that there are SRTCP application messages that may need end-to-end integrity protection [6].

- **End-to-end replay protection**
  Inclusion of PUV can be used as a counter for end-to-end protected parts and provide end-to-end replay protection. But the framework does not enforce this use. The reason mentioned in the proposal is that the server from the cloud is semi-trusted for proper deliveries of SRTP packets, thus not replay them [6]. On the other hand, if the server is not trusted

with authentication and confidentiality of the media, then replay protection can also be provided using PUV as an end-to-end counter [6].

3 Proposed Solutions

After analysing the related works and the solutions they propose, an assessment of the needs of a secure cloud conferencing is done. The assessment shows that an end-to-end solution for RTCP packets and RTP header extension have never been explored with any of the existing frameworks. But both RTCP packets and RTP header extensions are part of RTP structure and eventually going to be used for cloud conferencing. This why it is important to find out how to protect the payloads of these formats if they need to end-to-end security. Therefore, the end-to-end security design of SRTCP and secure RTP header extension are parts of an end-to-end SRTP solution for cloud conferencing.

Another important aspect that has been missing from the Ericsson and Cisco frameworks are one or more proposals of key management protocol that is compatible with the modifications of SRTP. The design choices presented in this thesis provided solution for these missing needs – designing SRTCP, RTP header extension and DTLS-SRTP specifically for a cloud conference in Sections 3.1, 3.2 and 3.3 respectively.

Many of these design choices use security features proposed by Cisco and Ericsson solutions. These features and the motivations for employing them in the proposals made by this thesis are explained below.

- **Use of EKT**
  Use of EKT makes end-to-end key distribution from the KMF to the Endpoints scalable and secure. The other options could have been the use of a group master key or different master key for each Endpoint. The second option is not very scalable and more resource consuming on KMF's part. If the first option is used and each Endpoint wants to have a unique master key derived from the group key then they have to rely on some other parameter to do so, for example their participant ID or SSRC. Among all these choices, EKT is the most secure and scalable way of master key negotiation. This also enables the Endpoints to generate individual end-to-end master key for the SRTP sessions. It already has its own protocol specification and offers usefulness in coordinating ROC and SSRC values between the participants as explained in Section 2.2.3.

- **Use of SSS as an optional field**
  The framework should be flexible in terms of rewriting SSRC by the switching server to make it compatible with certain RTP topologies. The solutions can be making SSS, an optional field or moving the original SSRC to CSRC by the server. In the second case, the server needs to be trusted for this action and the field cannot be end-to-end protected. But any field from RTP header directly related to end-to-end encryption should be end-to-end authenticated. Therefore using SSS as proposed by Ericsson presents itself as a better solution. It provides an option to keep the payload protection independent of using any fields from RTP header and prevents any possibility of being modified by third party. SSS is transported as part of the end-to-end protected protection and remains

integrated between one trusted Endpoint to another if SSRC needs to be modified by the server in the cloud.

- **Use of PUV**
  Using end-to-end PUV solves the large sequence number gap problem on hops which is described in Section 2.3.1.6. It enables the server to change outgoing sequence number. In consequence, the receiver is not faced with repair attempts or buffering issues thinking there is a packet loss. Moreover, keeping records of PUV(s) can provide end-to-end replay protection of media contents. This is why use of PUV has been integrated with modification of SRTCP proposed in this thesis.

- **Use of Tunneled DTLS-SRTP**
  KMF's negotiation of end-to-end security context with the Endpoints while placing the Media Switching Server in the middle has been mentioned in both the solutions (Ericsson & Cisco). This keeps KMF from opening its secure ports to multiple Endpoints in the conference. It enables the KMF to maintain limited key management sessions through the media servers to the other Endpoints. Therefore one of the task of this thesis was to explore the possibility of having such solution and finally design it in detail.

3.1 SRTCP for Cloud Conferencing

A new modified packet format for SRTCP has been proposed in this thesis. This new packet format is going to be used by the protocol, SRTCP for monitoring data transmission of SRTP packets among other functions for cloud conferencing.

3.1.1 Existing Design

The RTP control protocol (RTCP) is responsible for transmission of control packets to all participants in a multimedia session. The primary function of these control packets is to keep track of the quality of data transmission. The other functionalities involve sending session description items, application specific information etc. [2]. The security of RTCP is provided by the protocol Secure RTCP (SRTCP).

3.1.1.1 RTCP Packet Types

Based on the functionalities, the RTCP packets are classified into several packet types as mentioned in Section 6 of [2]. Which are:

- Sender Report (SR)
- Receiver Report (RR)
- Source Description Items (SDES)
- Goodbye Packet (BYE) and
- Application Defined Packet (APP)

3.1.1.2 SRTCP Packet Format

According to Section 6.1 of RFC 3550 [2], multiple RTCP packets are stacked upon each other to create a compound packet that can be sent over UDP as a single packet. According to RTP specification, each compound RTCP packet must carry at least two packets. The first packet has to be a Sender Report or a Receiver Report. It must also carry an SDES packet with a CNAME item. CNAME or canonical name is an alternative for SSRC [2].

The protocol that is followed for securing a RTCP packet is SRTCP. In Section 3.4 of [1], it has been specified how to secure a compound SRTCP packet with authentication and encryption using SRTCP master key. The format of a secure RTCP packet is shown below. The figure is adopted from Section 3.4 of [1].

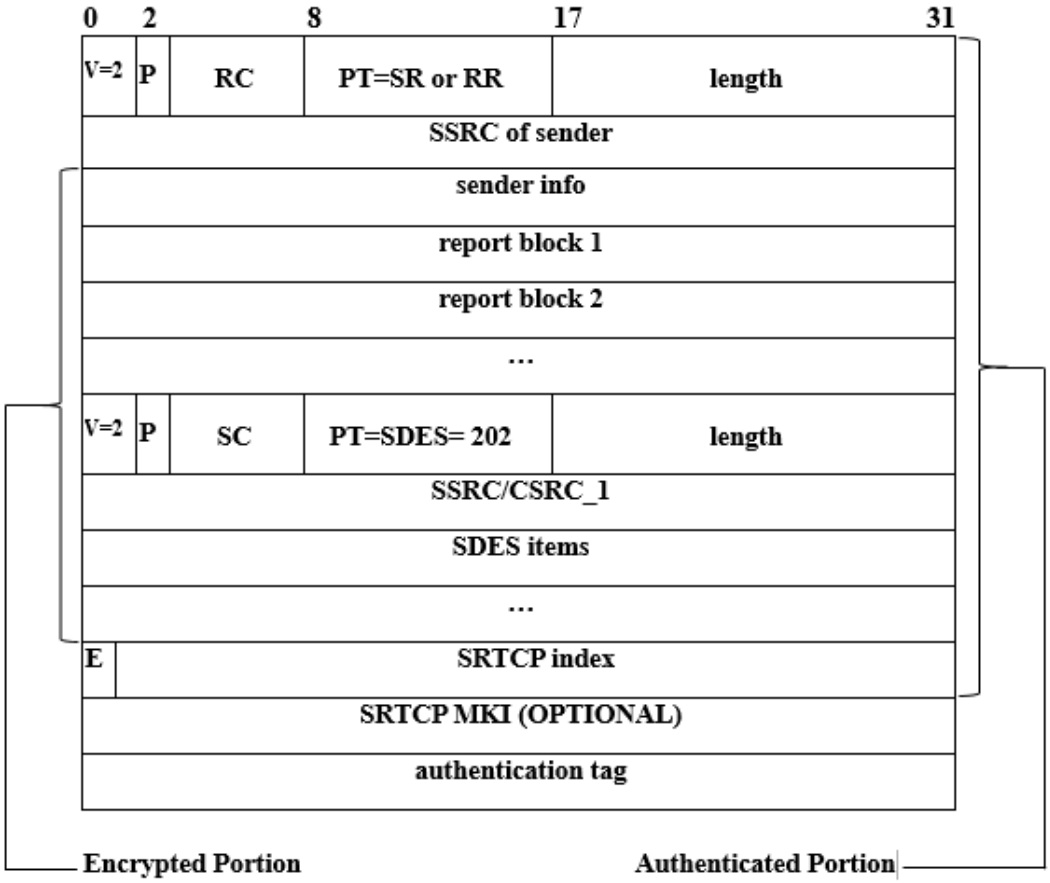


Figure 17. Secure RTCP compound packet

The portion starting after the top SSRC field up to SRTCP index is encrypted and the entire packet leaving MKI (master key indicator) field is authenticated. Four additional fields are included to a typical RTCP packet to accommodate its security features [1]. These fields are described below as they are mentioned in [1].

- The E-flag indicates whether the SRTCP packet is encrypted or not. If E equals “1” then the packet is encrypted. If E equals “0” then it is not.
- SRTCP index is a 31 bit value that is incremented by one with each new SRTCP packet sent in a particular stream.
- MKI has the same definition as the one in SRTP packet.
- SRTCP may or may not be encrypted but it always has to be authenticated. This is why presence of an authentication tag for each of the SRTCP packet is mandatory.

3.1.1.3 Use of SRTCP Index and SSRC

A single master key is used for deriving the keys for encryption and authentication of SRTCP packets. The pre-defined cryptographic transforms mentioned in RFC 3711 uses the field SRTCP index as one of the parameters for deriving these keys [1]. Moreover, the fields SSRC (the one on top of an SRTCP packet which identifies the source that generates the packet) and SRTCP index are used for calculating an Initialization Vector (IV) for the encryption mechanism [1]. Figure 18 shows the construction of IV when AES-GCM is used to protect an SRTCP packet. This figure corresponds to the one in [12].

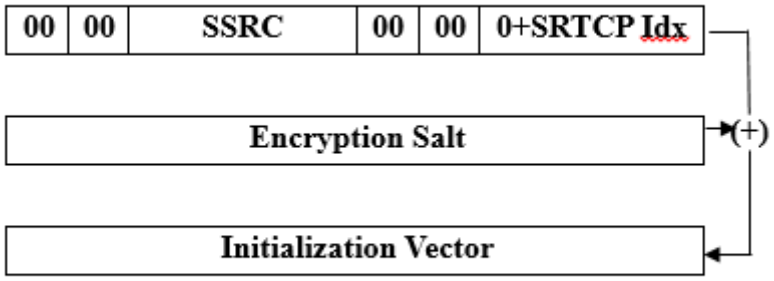


Figure 18. SRTCP IV for AES-GCM

The IV is formed by adding two octets of zeros and 32 bit SSRC with the result of the addition of two octet of zeros and 31 bit SRTCP index with one bit zero at the front of the index. The result of the final addition is Exclusiver-ORedwith the 12 octet encryption salt to receive a 12 octet IV which is used for encrypting the mentioned portion from Figure 17 [12].

IV for each of the packet is kept unique by using unique SRTCP index found from that packet. This prohibits key-stream reuse which can lead to serious compromise of security. Inclusion of SSRC makes sure that using the same master key for protecting more than one SRTCP streams in an audio/video session is safe since SSRC(s) are also maintained as unique values between all the sources participating in the session. Replay protection of SRTCP packets are achieved by logging the SRTCP indexes of the authenticated packets received from a particular source [1].

3.1.2 Necessity of a New Design

In the cloud conference architecture, the Media Switching Server in the cloud transmits SRTP packets from one Endpoint to others. In traditional RTP topologies, some middleboxes process the RTCP packets received from different sources and create compound packets out of them and forward them to intended recipients. If these packets are secured by SRTCP specification, then the middleboxes have access to the encryption keys to perform the processing. Some of the middleboxes create their own RTCP control packets, typically Sender and Receiver reports and send them to the participants. Therefore, it makes sense that the Media Switching Server from the cloud conferencing structure should also be able to generate its own SRTCP packets or have access to packets coming from other sources to create compound packets. Hop-by-hop encryption with SRTCP master keys shared by the cloud and the Endpoint is sufficient for this requirement. But there are certain RTCP packet types that carry information that are meant to be shared only with the receiving Endpoints. These are:

- SDES packet carries personal information in SDES items such as NAME, EMAIL, PHONE, LOC (location) etc. [2].
- APP packet may carry application specific information that must not be inspected by the switching server
- New RTCP packet types can be registered through IANA [2]. There is always the possibility that some of them will carry payloads that need end-to-end protection

The conclusion is that there is a need to have a solution through which parts of SRTCP can be protected end-to-end as well as hop-by-hop.

3.1.3 Alternative Design Choices

Before a final design is chosen, the requirements for the design need to be identified. Then the possible design choices matching the requirements are considered.

3.1.3.1 Requirements of Modified SRTCP for Cloud Conferencing

From the discussion in previous Section, it is understood why a new design of SRTCP for cloud conferencing is needed. Summarising the discussion, below requirements for the solution are listed (including the requirements from Section 2.1.5):

- The new design must conform to RTP topologies for multi-party conference from [10].
- It must be possible for the modified SRTCP packet format to be used for compound packet generation following Section 6.1 of RFC 3550 [2].
- Any RTCP packet type can be hop-by-hop encrypted or end-to-end encrypted based on the sensitivity of the data. A compound SRTCP packet must be able to carry both types of packets in it.
- If certain SRTCP header fields are used as parameters for end-to-end cryptographic transform, then their end-to-end authentication is

mandatory. This also includes those fields that carry information which should not be shared with the Media Switching Server.

3.1.3.2 The process and the Design Choices

According to the proposals by Cisco and Ericsson, the idea behind the end-to-end protection of SRTP packet is to encrypt the payload of the SRTP packet and authenticate the header fields required for end-to-end encryption mechanism by including them in Associated Data. In case of SRTCP, following this mechanism does not satisfy the requirements listed above. Because a single compound SRTCP packet may carry more than one type of RTCP packets with their own separate payloads. These packet types may be generated from more than one source. Therefore, the challenge for this design is to manage selective encryption of more than one payload, ensure authentication of the source for each of the payload at the receiving Endpoint and ensure authentication of the header fields required for end-to-end cryptographic transform of a compound SRTCP packet. The alternative design choices that are considered in this thesis for enabling the selective encryption are given below.

- **Using the field PT and External Signalling**
  The field PT (packet type) is an 8 bit constant value which is specific for a particular type of RTCP packet. PT belongs to the header of that particular packet type. A Sender Report RTCP packet has the PT value of 200, a Receiver Report RTCP packet carries PT=201 etc. [2]. The selective encryption can be done based on the value of PT. The payloads that will be encrypted end-to-end, their corresponding values of PT will be externally signalled to each of the entity belonging to the conference. Same will be done for RTCP payloads that are to be hop-by-hop encrypted. Looking at the value of PT, the Endpoint will know which operation to perform on the packet type. Only those values of PT that are associated with payloads that are hop-by-hop encrypted will be signalled to the Media Switching Server. This way, the server will not try to decrypt those RTCP packet types which are end-to-end encrypted and forward them as they are.

- **Using E Tag**
  A specific field in SRTCP packet can be used to indicate which type of encrypted payload it carries. The E tag from existing design of SRTCP can be used to carry out this function. The E tag will be set to 2 bits instead of 1.
  E = 00 indicates that the packet carries no encrypted packet types
  E = 01 indicates that the packet carries hop-by-hop encrypted packet types
  E = 10 indicates that the packet carries end-to-end encrypted packet types
  E = 11 indicates that the packet carries both end-to-end and hop-by-hop encrypted packet types

- **Using a new Packet Type**

A new packet type is introduced to carry one or all the different packet types that need to have end-to-end protection. Only the trusted Endpoint is allowed to create this new combined packet type.

- **Division of Compound Packet**
  The compound packet may be divided in two categories. One holds only the end-to-end encrypted packet types and the other holds only the hop-by-hop ones. Both can carry non encrypted packets.

3.1.4 Analysis of Different Choices

An analysis of the different choices is important for identifying the suitable ones. The analysis for the above choices are given below.

- Signalling selective encryption using certain packet type (PT) values requires an implementation of external signalling mechanism which adds complexity to the existing SRTCP specification. The second drawback of this method is that it may make the selection constant for the whole session or stream. If the same packet type carries different information later in the audio/video session that may need different type of encryption, then it needs to be signalled using some other parameter than PT. Therefore, signalling only PT values may not be enough in this case.

- Using different values for E tag is a useful way of performing selective encryption but it leads to redundancy while generating a compound packet. Such compound packet will carry both end-to-end and hop-by-hop protected packet types. Specifying the value of E is not sufficient for the server to know which packet type is hop-by-hop encrypted or the other way. In this case, once again, external signalling is needed to let the server know which packet types are end-to-end encrypted.

- Introduction of a new packet type for carrying only the end-to-end encrypted packet types make both selective encryption and compound packet generation by the Media Switching Server easier. Inclusion of a mandatory end-to-end PUV and optional end-to-end SSS with the new packet type will give the switching server freedom to generate SRTCP compound packets with its own SRTCP index. The server will also be able to add its own SSRC or remap the SSRC value if needed.

- Division of compound packets based on end-to-end and hop-by-hop encryption deviates from existing RTP specification. If the selective encryption using E tag is followed then division of compound packets is a better idea. On the other hand, using a new packet type does not lead to division. The new packet type carrying end-to-end protected payloads can easily be incorporated in any compound SRTCP packet, may it be generated by the Endpoint or the switching server in the cloud.

In a conclusive note, the use of end-to-end protected packet types encapsulated in a single new packet type fulfils the requirements mentioned for designing a SRTCP packet for cloud conferencing.

3.1.5 Final Design

The design choice with a new RTCP packet type has been finalized as a solution for performing selective encryption on an SRTCP packet. The new format should be applicable to all possible topologies for cloud conferencing. Using this design, it is possible for the Endpoints or the Media Switching Servers to produce a compound SRTCP packet containing both end-to-end and hop-by-hop protected portions as required. According to this design, any RTCP packet type can be hop-by-hop protected or end-to-end protected or change its method of encryption during the session depending on the sensitivity of the data. The details are described below.

3.1.5.1 The New Packet Format

The design introduces a new packet type named 'ENC' that will be used for combining any or all RTCP packet types carrying information that need to be end-to-end protected. Figure 19 below shows an example of ENC packet type and its format.

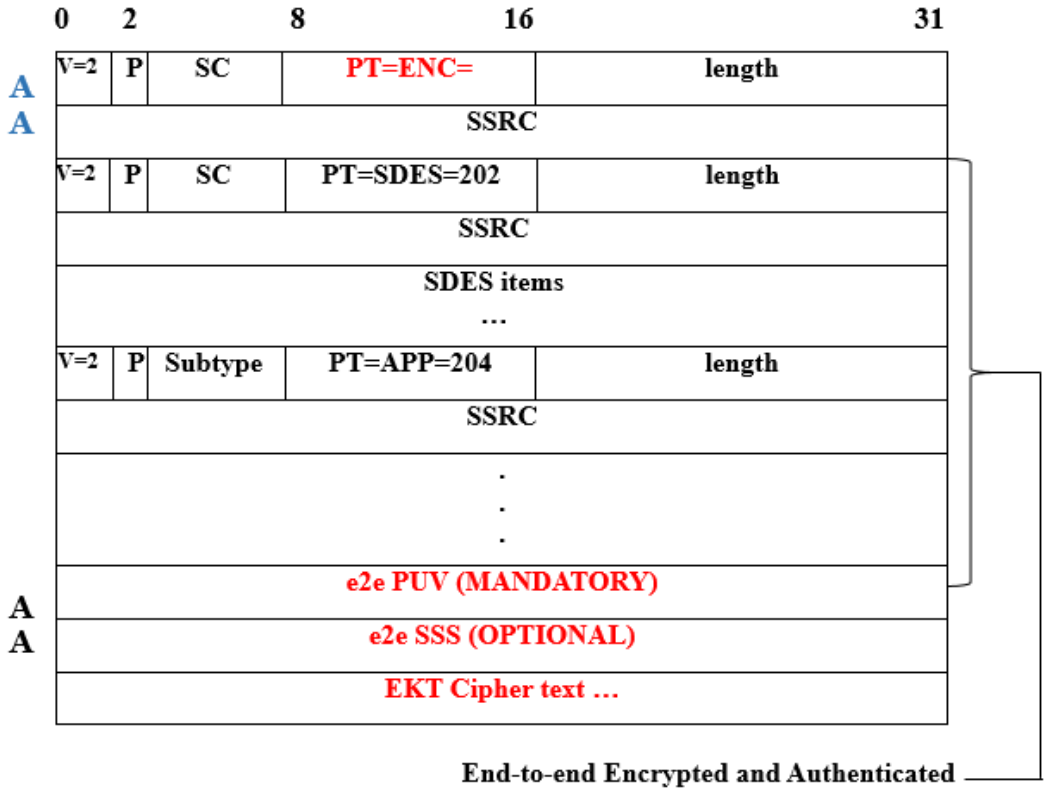


Figure 19. An example of an End-to-end protected part of a Secure RTCP packet

The typical header fields of ENC packet (V, P, SC, length) hold the same definitions as described in RTP [2]. The first SSRC on top is the identifier of the source which generates this particular combination of encrypted packets. The

encrypted packets within the ENC packet have their own headers with their own SSRCs identifying the sources to which the particular information in the packet belongs to. The individual format of each of the nested packet types follow the exact specification from RFC 3550 [2].

Each ENC packet must have the field ‘e2e PUV’ (end-to-end Packet Unique Value) and inclusion of 'e2e SS' (end-to-end SRTCP source) is optional. The definition of ‘e2e PUV’ and ‘e2e SSS’ follow the same from Section 2.3.2.2.

PUV is inserted at the end of the encrypted packets for calculating IV needed for the end-to-end encryption transform. If SRTP indexes are used in this case, then the ENC packets becomes vulnerable to replay attacks. PUV is used as a specific end-to-end counter for the ENC packets coming from a particular source. Also, SRTCP index is a hop-by-hop value that changes with each hop, but PUV will remain constant until another Endpoint processes the ENC packet and attaches a new PUV. So it makes the end-to-end protection completely independent from any SRTCP fields that are used for hop-by-hop protection.

The use of an ‘e2e SSS’ is optional here because it depends on the topology of the cloud conference and implementation at the centralized server. An inclusion of ‘e2e SSS’ becomes necessary for calculating end-to-end IV and identifying the source responsible for the encryption when the top SSRC after the ENC header is changed by any Endpoint or any server in the cloud.

The last field of an ‘ENC’ packet contains the e2e master key encrypted with EKT key. This master key is used to encrypt the ENC packet.

The section of the packet starting after first SSRC field up to the ‘e2e PUV’ field is encrypted and authenticated end-to-end. That means all the complete nested packets including the headers and payloads inside the ‘ENC’ packet are encrypted and authenticated. An AEAD algorithm must be used for the process. The contents of the Associated Data are discussed in the next Section.

3.1.5.2 Compound Packet Generation

The server in the cloud can include the ENC packets from different Endpoints in the compound packet in two ways. These are:

1. It can send a single ENC packet containing a chunk for each contributing source from which it is receiving the source’s own ENC packet. Each chunk will comprise the start of the ‘end-to-end encrypted and authenticated part’ to the end of ‘EKT ciphertext’ from Figure 19. The format is shown in Figure 20.

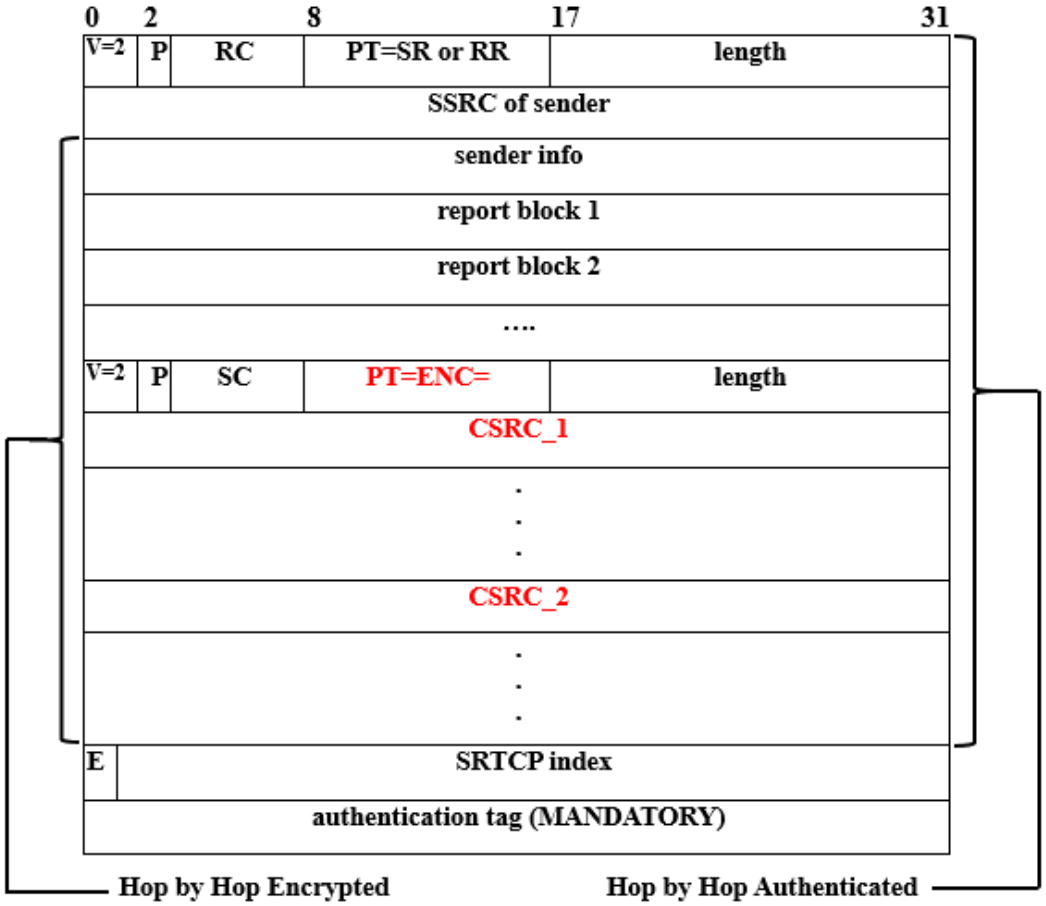


Figure 20. SRTCP compound packet for cloud conferencing

In this case, the server produces its own 'ENC' header and adds the SSRC value of the source it receives an ENC packet from, in the CSRC_1 field. Then it copies the above mentioned chunk under it. Following the same method, it inserts the other ENC chunks one after another. The total number of sources of the attached ENC chunks is entered in 'SC' (Source Count) field of ENC header by the server.

The content of the Associated Data for end-to-end cryptographic transform for this method is 'e2e PUV' and 'e2e SSS' (if included). These fields are marked with 'A' in Figure 19 on the left side of the packet.

2. The second way is to send multiple complete ENC packets stacked upon each other by the server. In this case, the whole packet including the original ENC header from the source will be copied in the compound packet. If this method is followed, end-to-end Associated Data will include all the fields from the ENC header (V, P, SC, PT, length, SSRC), e2e PUV, e2e SSS (if included). The fields from ENC header are marked with 'A' in Figure 19.

The format of the rest of the compound SRTCP packet remains same which starts with a SR/RR, includes an SDES packet carrying a CNAME item and other non e2e protected payload(s). The compound packet ends with usual

fields of SRTCP index, E flag and hop-by-hop authentication tag. The whole packet is hop-by-hop encrypted and mandatorily authenticated according to RFC 3711 using existing SRTCP cryptographic transforms [1].

3.1.5.3 End-to-end Cryptographic Transform

End-to-end protection of SRTCP packet is done by AEAD algorithms. The pre-defined transform is AES-GCM mentioned in [12]. In the current specification, the IV that is used for the encryption mechanism is computed from the addition of SSRC and SRTCP index Exclusiver-ORed with the session encryption salt [12]. In the new design for end-to-end protection, the use of SRTCP index is replaced by 'e2e PUV' and SSRC is replaced by 'e2e SSS' if 'e2e SSS' is present. Therefore, the IV is updated with a new format for end-to-end operation of SRTCP. It is shown in Figure 21.

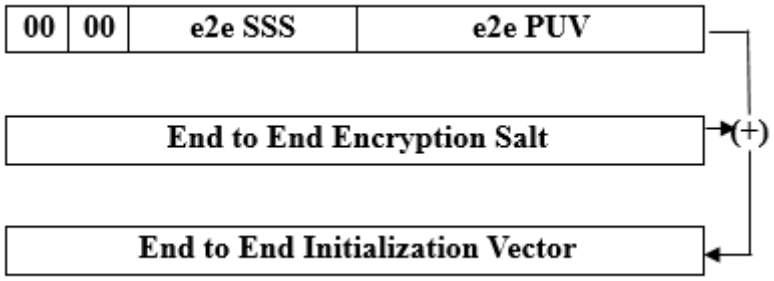


Figure 21. End-to-end SRTCP IV Formation

Here 'e2e SSS' is concatenated with 2 leading octet of zeros and added with 'e2e PUV'. The result of the addition is then bitwise Exclusiver-ORed with the session encryption salt to produce the 12 octet IV needed for end-to-end encryption of ENC payload. To make the 'e2e SSS,' and 'e2e PUV' a 4 octet and 6 octet value respectively, leading zeros are added with each of them.

The end-to-end SRTCP master key has to be completely unique from the hop-by-hop SRTCP master key according to the requirements of cloud conference architecture. The derivation of session keys for both the operations are done the same way as they are mentioned in Section 4.3 of [1]. The authentication key derived from the master key for end-to-end protection is ignored as an authenticated encryption algorithm is followed for the process.

3.2 Secure RTP Header Extension for Cloud Conferencing

RTP header extension is an optional part of RTP/SRTP packet. It is attached to media packets to experiment with new payload format independent functions [2].

It can be optionally encrypted following RFC 6904 [16]. A new modified format for encrypted RTP header extension has been proposed in this thesis which supports both end-to-end and hop-by-hop protection to be used for cloud based conferences.

3.2.1 Existing Design

Section 5.3.1 of RFC 3550 specifies a single field for RTP header extension to be carried in a RTP packet [2]. RFC 5285 provides a mechanism to fit multiple small header extensions under this single dedicated field for RTP header extension [17].

3.2.1.1 Packet Design

RFC 5285 specifies two formats to be followed to generate a complete 'RTP header extension' field carrying smaller extensions inside. These are – 'one-byte header' and 'two-byte header'. Each smaller extension is preceded with a 'local identifier' (ID) and 'length' fields. In 'one-byte header', the size of 'ID' and 'length' field are 4 bits each. In 'two-byte header', the size becomes one byte for each of these fields. Therefore, two-byte header format permits lengthier extensions [17]. An example of RTP header extension is shown below with one-byte header format in Figure 22.

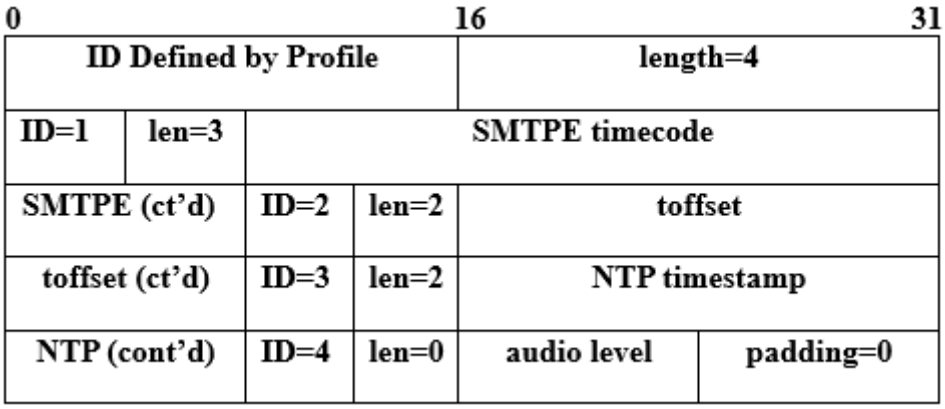


Figure 22. Example of RTP header extension

The first field is defined by RTP profile of the RTP packet the extension belongs to. The second field shows the length of the whole RTP header extension in 32 bit word count [17]. Like in the above example, the length is 4.

A single RTP header Extension field can be accommodated for more than one extension with the above format. In this example, four different extensions are included, these are SMTPE code, transmission offset, audio level and NTP timestamp.

Each type of extension is inserted with two header fields discussed above – 'ID' and 'Len'. The local identifiers (IDs) are distinct for each extension sent in a particular RTP stream or a packet. The values of them are negotiated with some external signalling mechanism such as SDP (Session Description Protocol). They may be mapped to a larger name space during session signalling. The next field shows the length of the extension payload that follows in bytes. Without any alignment, the next extension is inserted with its own local ID and 'length' fields [17].

3.2.1.2 SDP Signalling Design

The presence of a particular extension, its mapping to a larger namespace and negotiation of its local identifier value are done over session signalling. For example, using SIP offer/answer through SDP. Each extension is mapped with a URI (Uniform Resource Identifier). Extensions which are defined with existing RFC(s) are named with URI(s) starting with “urn:ietf:params:rtp-hdrext:”. In SDP a particular attribute (a =) called 'extmap' is used to indicate the value of the identifier of a particular extension and its URI [17]. Figure 22 carries an SMPTE timecode with local ID '1'. An example of how it is signalled using SDP is shown below.

a=extmap:1 urn:ietf:params:rtp-hdrext: smpte -tc

3.2.1.3 Encryption Mechanism

In the event of RTP header extensions carrying data that need confidentiality, an encryption process is provided in RFC 6904 [16]. This method of encryption is followed when the header extensions are formatted using the mechanisms given in RFC 5285 [17]. This is a selective encryption method. It means that out of all the extensions carried in one RTP packet, only those which are deemed to be confidential are encrypted and rest of the others remain unencrypted. This process of encryption is designed to be used only with SRTP packets. No separate authentication of these extensions are needed since they are authenticated along with the rest of the SRTP packet as specified in RFC 3711 [1], [16].

The field 'defined by profile’, followed by the 'length' field , any of the one-byte or two-byte headers of the extension elements and padding bits are never encrypted through this process. A separate header encryption key and salt are derived from the same SRTP master key that is used for protecting the SRTP header and payload [17]. The key derivation and the parameters used in the process remains same as mentioned in Section 4.3.1 of RFC 3711 [1].

If pre-defined SRTP transforms are used for encrypting the SRTP payload, the same transforms must be used to generate header extension keystream for encrypting them. The process and generation of IV(s) remain same as well [17].

The first step is to create the keystream. Then an encryption mask is computed. The encryption mask has all the octets as 1 for each of the octet belonging to extensions which are to be encrypted and the rest are computed as zeros. The local identifiers of the extensions to be encrypted are always pre-negotiated through session signalling between the SRTP sources which are willing to receive such extensions. This way the sender and the receiver can compute the correct encryption mask based on the signalled ID(s) [17].

Next, the encryption mask is bitwise-ANDed with the corresponding keystream to produce a masked keystream. The result is bitwise exclusive-ORed with the header extension to create the ciphertext version of the header extension [17].

An example of a plaintext header extension and corresponding encryption mask is shown below.

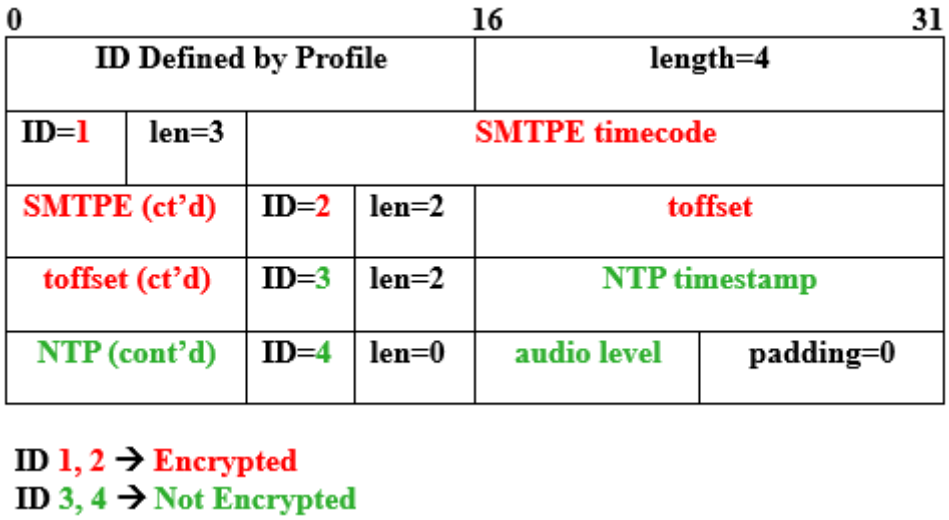


Figure 23. Example of a plaintext RTP header extension

Figure 23 shows that two extensions with IDs 1 and 2 are to be encrypted and the rest are not. How the corresponding encryption mask of this RTP header extension field looks like is shown in Figure 24.

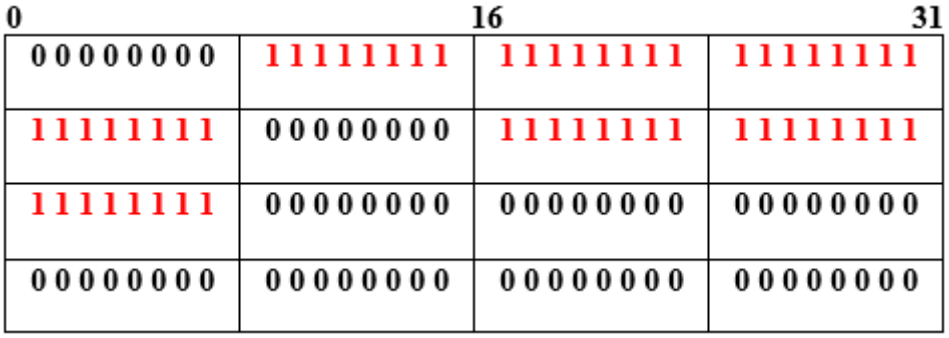


Figure 24. Encryption mask for the example from Figure 22

Figure 24 shows the encryption mask of the example RTP header extension from Figure 23. Here, the octets belonging to the extensions (SMTPE timecode and transmission offset) that are to be encrypted are computed as 1s and everything else as zeros.

If SDP is used as session signalling protocol, then the URI used for signalling encrypted extension is "urn:ietf:params:rtp-hdrext:encrypt". This URI is added before the original URI of the extension that it is mapped to [17]. In the above example, the local identifier values of SMTPE timecode and transmission offset are 1 & 2 respectively. They are signalled as encrypted extensions and their SDP attribute descriptions are:

a=extmap:1 urn:ietf:params:rtp-hdrext:encrypt \

urn:ietf:params:rtp-hdrext:smpte-tc

a= extmap:2 urn:ietf:params:rtp-hdrext:encrypt \

urn:ietf:params:rtp-hdrext:toffset

3.2.2 Necessity of a New Design

For the current structure of cloud conferencing, the Media Switching Server can be enabled to forward media streams using voice level indication. The employment of such method is performed by using RTP header extension following procedures mentioned in RFC 6464 [18]. In this case, the server should be able to access information contained in such particular header extension. In other scenarios, header extensions can be used to send data that is related to the transmission of media. Server in the cloud is also responsible and trusted to check, alter and include such information from the extensions.

But in future, new header extensions may be registered or used that can carry application specific information or personal information that the server in the cloud is not trusted to access. For example, the latest version of Cisco solution proposes that the end-to-end IV (EEIV) is to be sent with RTP header extension [15]. This piece of information carried by a header extension is directly related to media encryption and should not be viewed by the server from a conference hosting cloud.

Therefore, the above discussion points to the fact that a robust security solution for SRTP cloud conference should include a mechanism that facilitates carrying both hop-by-hop and end-to-end protected header extensions using the formats specified in RFC 5285 [17].

3.2.3 Alternative Design Choices

The requirements for designing a modified format of RTP header extension which includes end-to-end protection are given below.

- The format should be able to carry both end-to-end and hop-by-hop protected extensions under one field as designated by RTP specification in RFC 3550 [1].
- The encrypted data of end-to-end protected header extensions should be end-to-end authenticated.
- The server should be able to remove, modify or include unencrypted or hop-by-hop protected extensions if needed. These actions must not affect the validity of end-to-end protected part of the RTP header extension field.

Two alternative design choices can be used to accommodate the above requirements. They are described below.

3.2.3.1 Using Independent Extension

According to RFC 6904, local identifiers of encrypted header extensions are differentiated from unencrypted ones during signaling with the use of a

separate URI [16]. Similarly, a distinguishing URI for header extensions can be used that are to be negotiated as end-to-end encrypted.

A separate keystream will be generated for encrypting the extensions that need end-to-end protection. Also, a separate end-to-end encryption mask will be generated which replaces the octets of extensions which are to be end-to-end encrypted with 1 and the rest becomes zeros.

Hop-by-hop operation follows the same procedure mentioned in RFC 6904 [16]. Only exception is the computation of encryption mask which replaces the octets of extensions which are to be hop-by-hop encrypted with 1 and the rest with zeros. A summary of the design is shown in Figure 25.

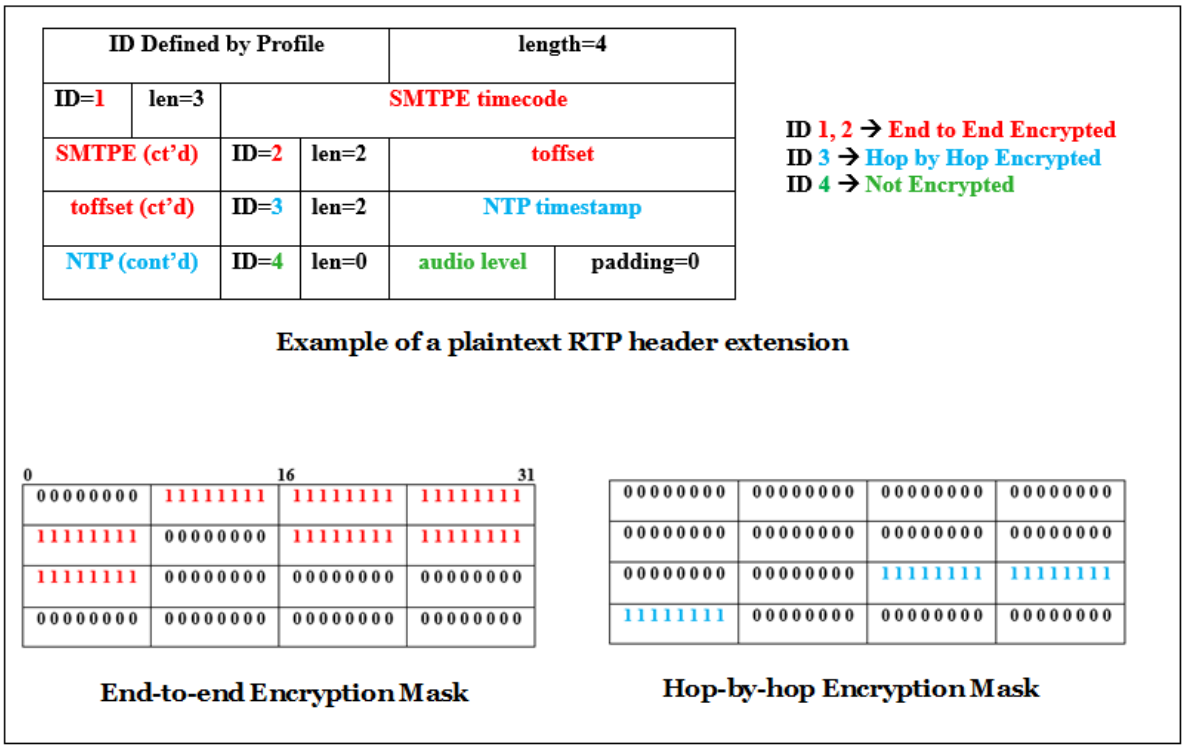


Figure 25. Use of independent extension for providing End-to-end protection

According to this design, the Endpoint needs to compute two encryption masks but the Media Switching Server needs to compute just one. The server may eliminate the other extensions or add new. If it does so, the end-to-end encryption mask computed at the Endpoint will not be effected if they are placed at the top preceded by 'defined by profile' and 'length' fields.

3.2.3.2 Using Encapsulated Extension

In this alternative design, the header extensions in need of end-to-end protection will be placed one after another and they will be identified with a single local identifier and considered as one extension. The idea here is to encapsulate all end-to-end extensions in a hop-by-hop header extension. Inside the encapsulated hop-by-hope extension, each of the end-to-end extension will be identified with local values that are only negotiated with trusted Endpoints.

The protection of these extensions does not have to depend on the encryption process specified in RFC 6904 [16]. At the server, this end-to-end portion will be considered as a single hop-by-hop extension. Therefore, this addition of encapsulation in the overall format will not alter the processing of the whole RTP header extension field as mentioned in RFC 6904 [16].

3.2.4 Analysis of Different Choices

As a final design, use of encapsulated extension is selected by this thesis to incorporate end-to-end protection in the already existing formats of RTP header extension by [17]. A comparative analysis of the two choices presented in Section 3.2.3.2 is given below to understand the motivation behind choosing one over the other.

- The first design restricts the placement of end-to-end protected extensions to be at the top followed by the other two types of extension – hop-by-hop and unencrypted. Use of an encapsulated extension frees the Endpoint to insert them anywhere in the series of extensions. It also lets the server freely move them around as a single extension.
- If independent extensions are used for end-to-end protection, then the Endpoint has to compute two encryption masks in the event of the packet carrying both hop-by-hop and end-to-end protected extensions. Use of encapsulated header extension follows a simpler method to protect the extensions by removing mask operation from the end-to-end encryption method.
- By encapsulating end-to-end header extensions inside a hop-by-hop one, the mechanism provided by RFC 6904 will not be needed any modification and be followed as it is currently specified [16]. This way the end-to-end encryption mechanism becomes completely independent from the hop-by-hop one.

3.2.5 Final Design

How encapsulation is used for protecting header extensions end-to-end is shown in Figure 26.

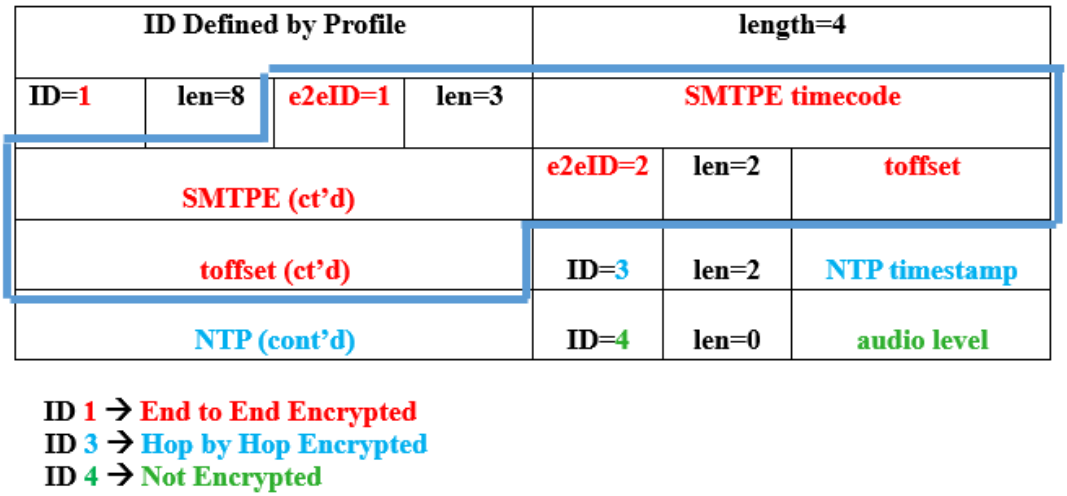


Figure 26. RTP header extension for cloud conferencing

The new design encapsulates all the end-to-end extensions under one single local identifier. In Figure 26, it is ID=1. Inside the encapsulation, the extensions are distinguished from each other by using e2eID(s). They have the same functionalities as the local IDs but only negotiated with trusted Endpoints with session signalling.

The same SRTP master key that is used for protecting the media in the session is used for deriving the end-to-end session header encryption key. The derivation process is done as mentioned in Section 3 of RFC 6904 [16]. The encryption algorithm is the AEAD algorithm chosen as part of the SRTP end-to-end security context. The default algorithm is AES-GCM specified in [12]. Therefore, the process of encryption and calculation of IV remain same for both SRTP payload and header extensions that need end-to-end protection. After the end-to-end encryption on the encapsulated extensions are performed, the resulted ciphertext should be added in the Associated Data for checking its authenticity at the receiving Endpoints.

Though Figure 26 shows only one encapsulated header extension, there is no limit to the number of encapsulated extensions a SRTP packet can have.

SDP may be used for signalling existence of RTP header extensions in the packet and negotiating the local identifiers, as well as the e2eID(s). End-to-end encrypted header extension element is signalled in SDP with extmap attribute using the URI ending with the words 'e2e encrypt'. For the example in Figure 26, signalling messages of end-to-end extensions between the Endpoints will contain:

a = extmap:**1** urn: ietf:params:rtp-hdrext:**e2e encrypt**\

urn:ietf:params:rtp-hdrext:smtpe-tc

a= extmap:**2** urn:ietf:params:rtp-hdrext: **e2e encrypt** \

urn:ietf:params:rtp-hdrext:toffset

According to header extension encryption mechanisms, local identifier, 'len' and padding – these fields are never encrypted [16]. Therefore, when the server in the cloud receives such SRTP packet with RTP header extension from Figure 26, it inspects the local ID (here it is 1) and the length (which is 8 here) and it knows from where to start the decryption mechanism of the hop-by-hop extensions.

3.3 DTLS-SRTP for Cloud Conferencing

HTTP header or DTLS-SRTP are among the several solutions that can be used as key management protocol for cloud conferencing. In this particular thesis, several design modification for DTLS-SRTP have been proposed to accommodate the changes needed for Cloud Conferencing.

The existing design of DTLS-SRTP has already been discussed in Section 2.2.2. This Section discusses the requirements and necessity of alterations

3.3.1 Necessity of a New Design

The requirements and necessity of modifying the current specifications of DTLS-SRTP are listed below.

- The use case for cloud conferencing requires a Key Management Function (KMF) to distribute EKT keys and negotiate end-to-end security context with each of the Endpoint. The desired key management using DTLS-SRTP should be able to incorporate such entity into the design which does not take part in the exchange of media but only distributes keying materials for media protection.

- DTLS-SRTP for cloud conferencing should be able to accommodate the transfer of end-to-end keying material and context from KMF to the Endpoints in a secure way ensuring that they have no relation to the hop-by-hop keying materials. It also has to be made sure that the materials are inaccessible by Media Switching Server in the cloud or any other third party which is not recognized as a trusted entity in the conference.

- DTLS-SRTP also needs to be modified to accommodate the exchange of certain identity parameters which are exclusive to cloud conferencing. These parameters are:

  Participant ID – It is a unique identifier of a participant in the conference. It may be associated with its certificate fingerprint. It may be an ID from the enterprise the participant belongs to or can simply be some identity information provided by some service trusted by the KMF.

  Conference ID – It is an identity value for the conference created with the service of the cloud provider.

  Key ID – It is an identifier value for the EKT key if more than one EKT key is used to secure the conference.

  The use of the above parameters in setting up the conference prior to the establishment of DTLS session is explained in the next Section.

3.3.2 Exclusive Parameters of Cloud Conferencing

A flow diagram is shown in Figure 27 to provide a high level picture of how the identity parameters mentioned in the previous Section are used in creating the conference before they are used in DTLS-SRTP sessions.

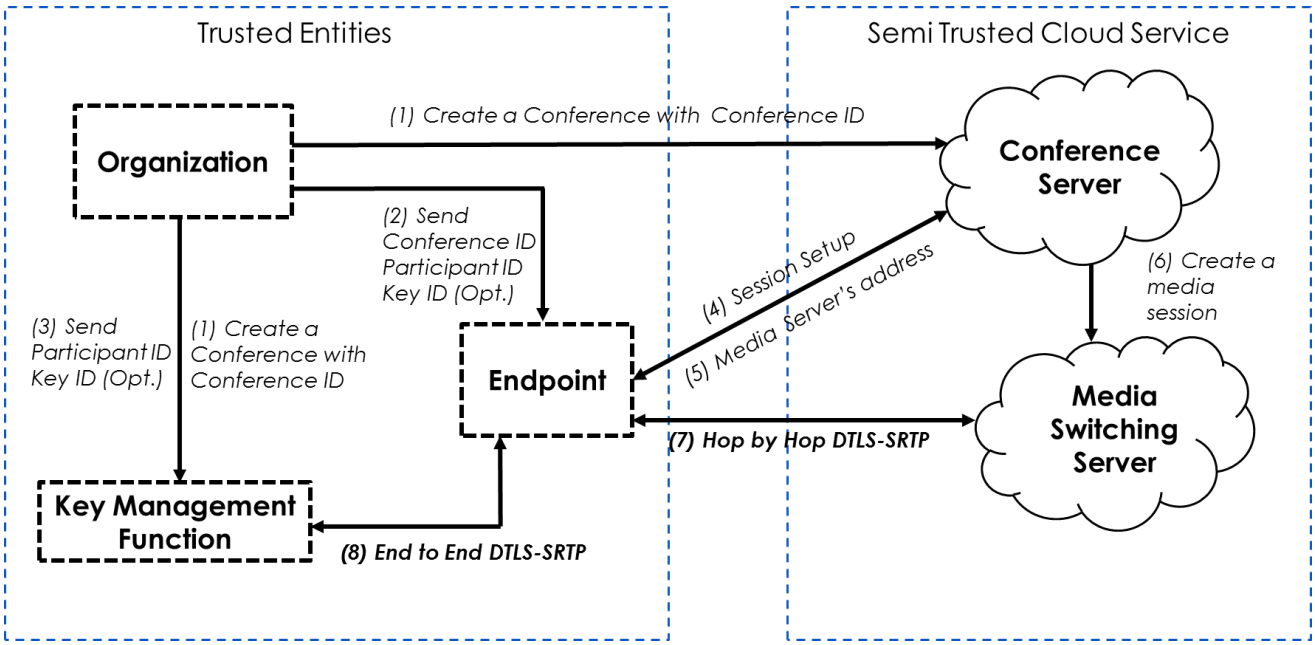


Figure 27. Use of identity parameters in cloud conferencing

All the entities in the left side box of Figure 27 are trusted. On the right side, the servers from the cloud provider are shown which are semi-trusted. The exchange of identity parameters and preliminary signalling for setting up of the conference are shown with numbered messages in the diagram. The messages are described in detail below.

(1) At first, the organization which wants to hold a conference with its clients schedules a conference in the conference server cloud with a specific conference ID. At the same time it lets the Key Management Function know about the conference and its ID.

(2) Secondly the participants are invited to this particular conference and are given the conference ID, their respective participant IDs and Key ID, if there is one.

(3) All the participant IDs and Key IDs corresponding to that conference will be notified to the KMF. Later on, KMF can check the IDs of the participants before delivering the EKT keys to them.

(4) Next, the Endpoint contacts the Conference Server in the cloud. By means of some external signalling, they exchange the needed parameters to set up the session for the given conference. This particular server in the cloud handles the signalling needed to setup the conference but has nothing to do with the exchange of actual media packets.

(5) The Conference Server provides the address of the Media Switching Server to the Endpoint.

(6) The Conference Server creates a media session in the Media Switching Server for that particular conference. The later server is responsible for forwarding the media packet to and from the Endpoints.

(7) The Endpoint establishes a DTLS-SRTP session with the Media Switching Server to negotiate the hop-by-hop security context and derive the hop-by-hop keying materials.

(8) The Endpoint also establishes a second DTLS-SRTP session with the KMF to receive the EKT key for the conference and also negotiate the end-to-end security context.

The Endpoint uses both sets of keys to secure the SRTP and SRTCP packets and send them along the media path created with hop-by-hop DTLS-SRTP session.

3.3.3 Alternative Design Choices

Several ideas of implementations have been considered to facilitate the requirements listed above. The choices mostly consider the use of a particular data type or a particular means that is going to differentiate between the hop-by-hop and end-to-end DTLS messages during a media session. All of these choices that were considered before the final design choice is made are listed below.

- **Use of SNI**
  Server Name Indication (SNI) lets the DTLS client mention the name of the server it is trying to contact by using the extension 'server_name' in the client hello [19]. This extension can be used by the Endpoint to indicate whether it wants to contact the KMF from the enterprise or the MSS (Media Switching Server) in the cloud. This could be a means for distinguishing hop-by-hop DTLS messages from end-to-end ones.

- **Use of Extended Hello**
  Similar to 'use-srtp' extension of current DTLS-SRTP specification mentioned in [3], another designated extension for end-to-end key management can be applied. The use of such extension in the extended Hello indicates that the client or the server is willing to establish a secure DTLS session to negotiate end-to-end security profile and related keying parameters. The 'extension_data' of this particular extension field can also carry the identity parameters mentioned in Section 3.3.2 for the KMF to inspect before delivering the EKT key to the requesting Endpoint.

- **Use of new TLS content type**
  According to TLS specification from [20], new TLS content types can be declared for specifying a particular type of application data. For designing a suitable DTLS-SRTP session for cloud conferencing, it can be a useful choice to introduce a new TLS content type that is going to be used to differentiate between hop-by-hop and end-to-end DTLS messages. This design choice can encapsulate end-to-end DTLS

messages inside hop-by-hop ones and facilitate negotiation of both end-to-end and hop-by-hoop keying materials and security profiles in a single session.

- **Use of ports**
  Using ports to differentiate between DTLS sessions for end-to-end and hop-by-hop key management can become very convenient. Specially, when all these sessions pass through or terminate at the MSS. Moreover, the use of an extended Hello or a new content type can be mixed with the use of different ports of MSS and can introduce suitable design choices.

3.3.4 Analysis of Different Choices

An analysis of the different choices is important for identifying the suitable ones. The choices that are mentioned in the previous Section are going to be considered for designing both direct and tunnelled session between each of the Endpoints and the KMF in cloud conferencing. As it is mentioned before, a tunnelled DTLS session is the one where the Endpoint connects through the MSS to the KMF to receive EKT key.

An SNI or a designated extension can be used for direct DTLS connection between the KMF and Endpoint. Proposing a new extension is a better idea. Because it indicates the session request is being sent particularly for end-to-end key management and at the same time it carries the identity parameters related to the conference and the participant requesting the connection. Moreover, the mechanisms remains similar to the use of current DTLS-SRTP extension.

A tunnelling protocol can be implemented with the use of a new TLS content type. None of the other choices except for a designated content type can facilitate the negotiation of end-to-end and hop-by-hop management in a single DTLS session.

Using different ports of MSS can still be used independently with tunnelling protocol or a Hello message extension.

3.3.5 Final Designs

Based on the requirements and the analysis from the previous Sections, several mechanisms have been proposed in this thesis to establish DTLS-SRTP connection for cloud conferencing. These mechanisms are categorized as follows:

- A direct DTLS-SRTP session between KMF and Endpoint
- A tunneled DTLS-SRTP session from KMF to Endpoint with Media Server in the middle

For the tunnelled connection, end-to-end and hop-by-hop DTLS messages are differentiated using:

- Ports or
- A new TLS content type

The above mentioned mechanisms are explained in the following Sections.

3.3.5.1 Direct DTLS-SRTP session between KMF and Endpoint

In this scenario, a DTLS-SRTP session is created between KMF and each of the Endpoint for the KMF to hand over the EKT key and select an end-to-end security profile to be used in the conference. A new extension is introduced for this purpose which is known as e2e_srtp_context.

The function of this extension is similar to that of use_srtp mentioned in RFC 5764 [3]. The difference between the two extensions is that e2e_srtp_context is used when SRTP payloads are protected with an end-to-end key instead of a typical SRTP hop-by-hop key.

The details of how this extension is used in relation to cloud conferencing is explained in Figure 28.

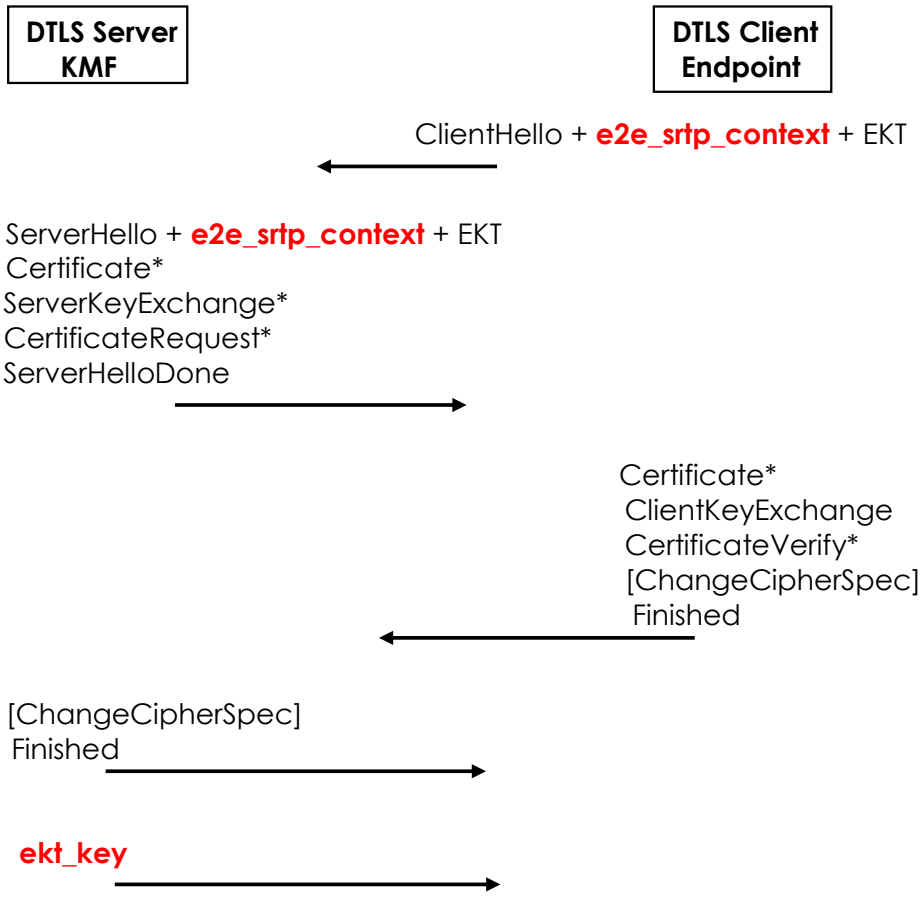


Figure 28. Direct DTLS-SRTP session between KMF and Endpoint

The Endpoint takes the role of DTLS client and the server is the Key Management Function.

Client initiates the session with sending ClientHello with the extension 'e2e_srtp_context + EKT'. The EKT extension enables secure transport of EKT keying material from one DTLS-SRTP peer to another as explained in the Section 2.2.3.3. The other extension, 'e2e_srtp_context' has been specifically proposed in this solution. It adds an extended Hello message that enables the Endpoint to send its participant ID, conference ID, optionally the Key ID and the list of security profiles it can support for the end-to-end media protection with the ClientHello. The extended Hello message format is shown in Figure 29.

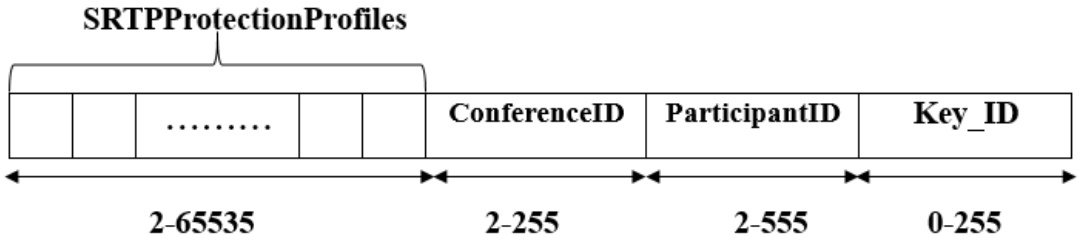


Figure 29. The data extension format of e2e_srtp_extension. The size of the fields are in bytes.

Figure 29 shows the contents of e2e_srtp_context data carried with DTLS hello message. SRTPProtectionProfiles field carries the values of the supported SRTP security profiles for secure transfer of RTP packets. In this case, the default algorithm is AES-GCM specified in [12]. This value is followed with the values of conference ID and participant ID carried in the data types named ConferenceID and ParticipantID. Inclusion of these values are mandatory. They hold a variable length which is between 2 and 255 bytes. The extended data can optionally carry a Key ID in Key_ID type.

KMF checks the validity of the identification information. It replies with a ServerHello with its own extension 'e2e_srtp_context' if the information are valid. Thus KMF accepts the connection and selects the profile that can be supported for this particular conference with the extended ServerHello.

Next the typical DTLS authentication is performed with exchange of certificates. Parameters for deriving TLS session keys are exchanged as well. After the Finished messages are received, KMF sends the EKT key using the secure DTLS channel. This session is never used to derive any SRTP keys or exchange SRTP packets like the usual DTLS-SRTP sessions do. The session may cease to exist after the EKT key has been sent.

3.3.5.2 Tunnelled DTLS-SRTP session through Media Server

The alternative of a direct DTLS-SRTP session between the KMF and the Endpoint is to establish a session keeping the Media Switching Server in the middle. This thesis proposes a new protocol named DTLSTunnel which sets up a tunnel through the Media Switching Server to exchange end-to-end DTLS messages from KMF to each of the Endpoints in the conference.

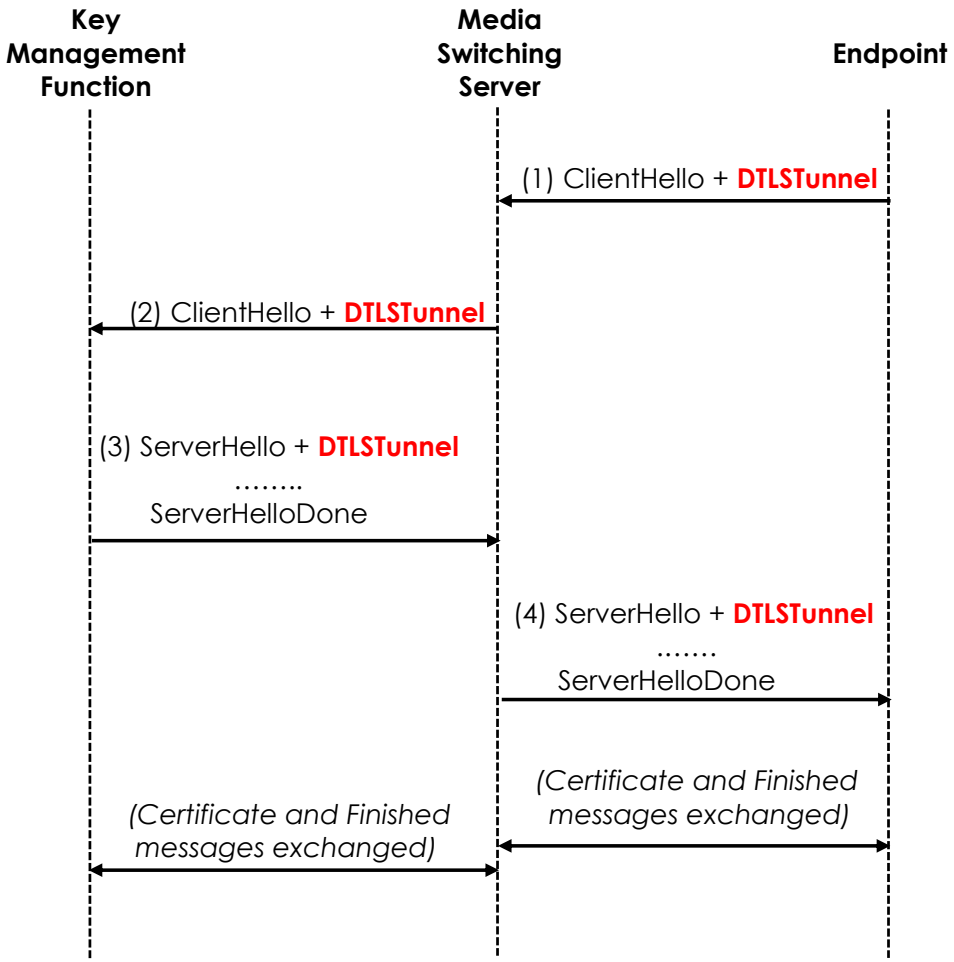
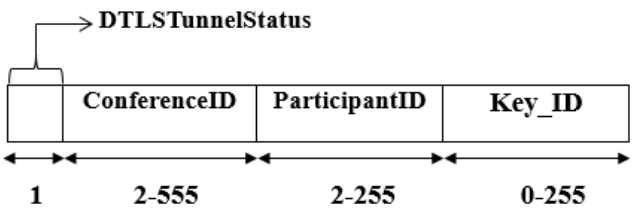


Figure 30. Tunnelled DTLS-SRTP session through Media Server

Figure 30 explains how DTLSTunnel is used in cloud conferencing to hold a secure tunnelled session to negotiate end-to-end SRTP security context. The Endpoint initiates the DTLS session sending ClientHello with the extension 'DTLSTunnel' to Media Switching Server (MSS) requesting for a tunnel connection. The extended hello message format for DTLSTunnel is shown below in Figure 31.



TunnelStatus Types	
Code	Description
0	request_tunnel_connection
1	accept_tunnel_connection
2	reject_tunnel_connection

Figure 31. Extended hello message format of DTLSTunnel. The size of the fields are in bytes.

The extended hello named 'DTLSTunnel' carries fours fields, which are 'DTLSTunnelStatus', 'ConferenceID', 'ParticipantID' and 'Key_ID'. The first

field holds a one byte value that signifies any of the three valid statuses involving the DTLS peer. If the peer requests for a tunnel connection, value '1' is sent. If it accepts the connection, then '2' or if it rejects the request from the other peer, then '3' is sent. The extended message also sends Conference ID, Participant ID and the optional Key ID for the KMF to check and deliver the EKT key upon validation of these IDs. The description and sizes of these data types are explained in previous sections of this chapter.

After receiving the request message from Endpoint to establish a tunnel, MSS sends its own 'ClientHello+**DTLSTunnel**' message to KMF requesting for a tunnel connection. It holds the same Conference ID, Participant ID etc. and passes them along to KMF.

KMF checks the sent credentials from MSS and replies with 'ServerHello+**DTLSTunnel**' message either accepting or rejecting the connection. If the request is accepted by the KMF, MSS sends its own 'ServerHello+**DTLSTunnel**' to the Endpoint accepting the tunnel connection itself. Thus, the tunnel from MS to KMF is established.

Next, Endpoint and MS send other typical DTLS handshake messages to each other and secure the session between them. On the other hand, MS and KMF also establishes a DTLS connection between them with typical handshake message exchanges.

After both of the sessions are secure, they are used simultaneously for sending tunnelled DTLS messages from Endpoint to KMF through MSS as shown in Figure 32. The tunnelled messages are sent in a new TLS content type called 'DTLStunnel'.

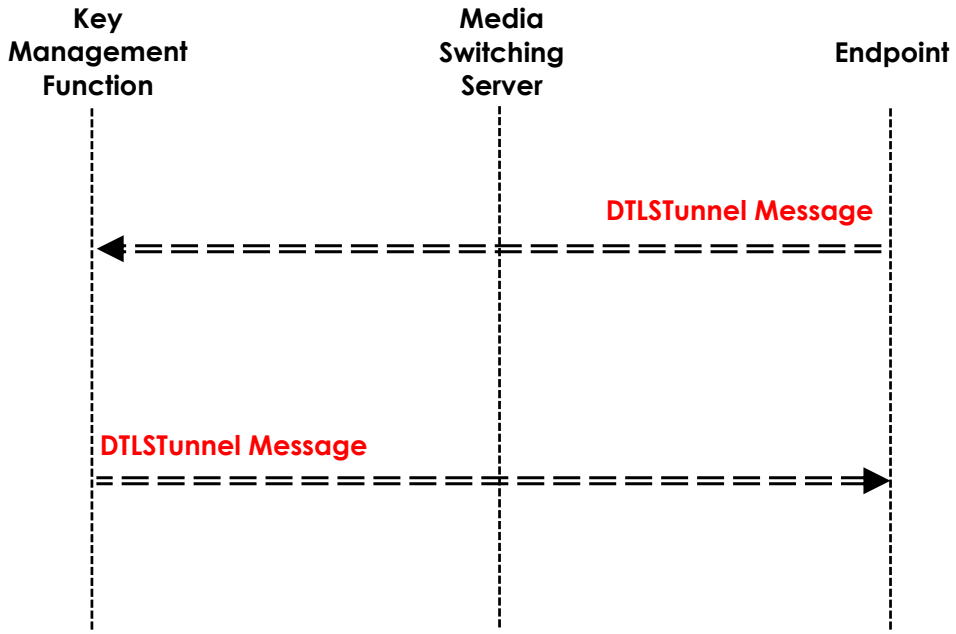


Figure 32. Tunnelled DTLS-SRTP session through Media Server

The message format of the content type is:

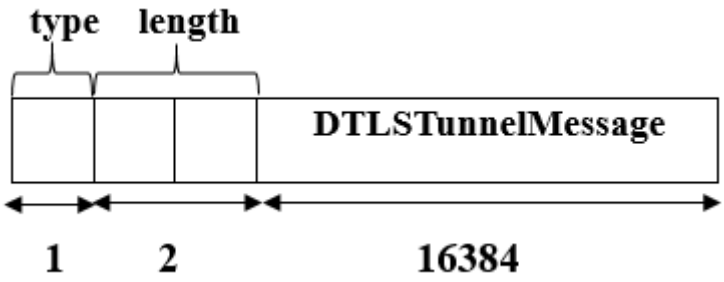


Figure 33. The message format of the new TLS content type – 'DTLSTunnel'. The size of the fields are in bytes.

Inside 'DTLSTunnelMessage', end-to-end messages for a DTLS connection between Endpoint and KMF are sent. The messages actually resemble the messages shown in Figure 28 that are used for the alternative choice - 'Direct DTLS-SRTP session between KMF and Endpoint' from Section 3.3.5.1. From Figure 33, we can see that each message is preceded by one byte 'type' field and two byte 'length' field. 'type' will hold the value of the number which will be provided by IANA once the new content type is registered with them. The data type 'length' mentions the length of the message in the content type. The length cannot exceed 16384 or 2^{18} bytes according to TLS specification.

Every time MSS receives such message with this particular content type, it forwards the message as it is to the KMF encapsulated in a hop-by-hop DTLS message. The EKT key is sent and negotiation of end-to-end SRTP context is done using the content of the encapsulated message.

This particular tunnelled session can remain persistent throughout the conference.

3.3.5.3 Use of Ports in Tunnelled DTLS-SRTP

It is possible to have various combinations of tunnelled DTLS-SRTP by using ports to differentiate them. The combinations are depicted in three scenarios with references to three separate figures.

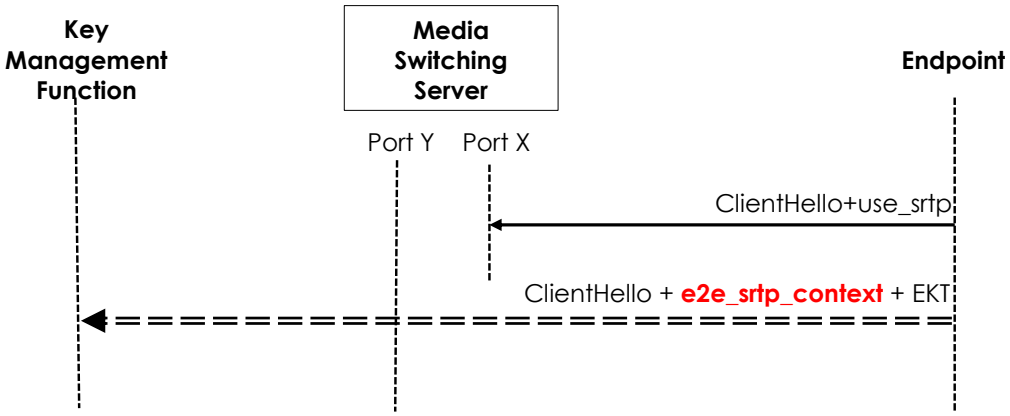


Figure 34. Scenario I

In the first scenario shown in Figure 34, the hop-by-hop negotiation for keying material is done in one port (port X) and the end-to-end negotiation in another port (port Y) of the MSS. In port X, 'use_srtp' is the extension to be used according to RFC 5764 for SRTP key management. In the other port, 'e2e_strp_context', the extension proposed in this thesis is used for end-to-end key management. In this case, Media Switching Server simply forwards the ClientHello message to KMF by recognizing the extension 'e2e_srtp_extension' in ClientHello. Once the DTLS connection is established, MSS is able to distinguish between the hop-by-hop and end-to-end DTLS messages as their corresponding connection is assigned to different ports. Then the Media Switching Server's job is to simply forward any message received at port Y to the other side. These messages are authenticated and protected with DTLS protection mechanism. Thus MSS is not able to interfere with the communication without detection. The endpoint uses the EKT key received from the end-to-end DTLS-SRTP session and uses it to send SRTP packets in the other session from port X with the MSS.

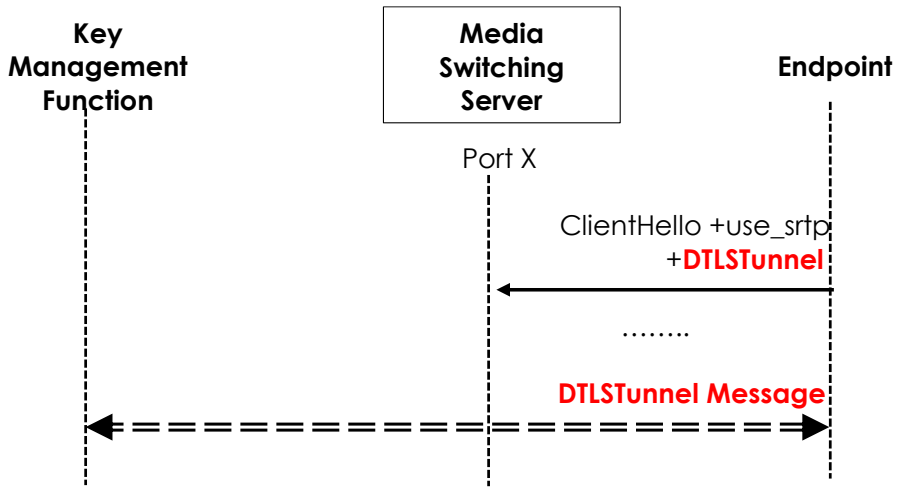


Figure 35. Scenario II

In the second scenario from Figure 35, both end-to-end and hop-by-hop key management are done using the same port (port X) belonging to Media Switching Server. The use of the content type – 'DTLSTunnel' differentiates end-to-end DTLS messages from hop-by-hop ones. The DTLSTunnel protocol makes it possible to use the same port for transferring both hop-by-hop and end-to-end keys.

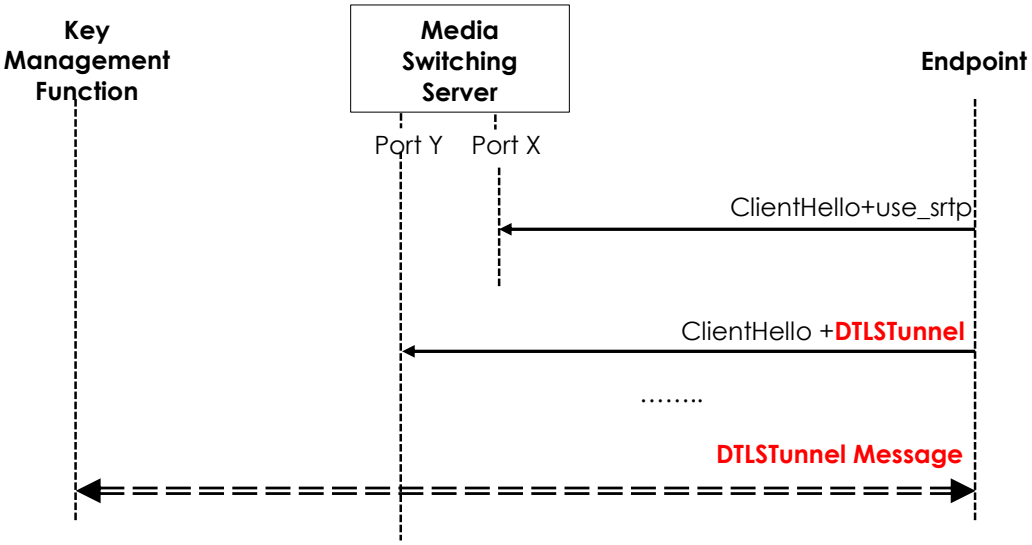


Figure 36. Scenario III

Scenario III depicted in Figure 36 is similar to the first scenario. The difference is that Port Y is used by DTLSTunnel protocol, instead by the extension 'e2e_srtp_context' directly. Here, he extension messages are encapsulated within tunnel messages for end-to-end key management. Though the proposed tunnelling protocol enables the use of a single port for overall SRTP key management, it still can be used with separation of ports.

3.3.5.4 Analysis of Final Designs

A direct DTLS connection with KMF from each of the endpoints introduces fewer changes to the existing DTLS-SRTP mechanism. One of the change is to use the Hello extension 'e2e_srtp_context'. Interestingly, the same procedure can be used but the DTLS connection between the KMF and the endpoints does not have to remain direct. This is made possible when an endpoint sends the end-to-end DTLS messages to a particular port of MSS as described in scenario I of Section 3.3.5.3 instead of sending them to the KMF directly. In this particular case, the extension 'e2e_srtp_context' emulates the functionality of DTLSTunnel protocol proposed in this thesis.

What are the advantages of using DTLSTunnel protocol in cloud conferencing? From Endpoint's perspective, the simplified structure should be to connect always to the cloud to receive all the information about the conference. Therefore, a single DTLS connection that employs DTLSTunnel protocol is needed. The Endpoint can use the same secure channel to receive both EKT key and hop-by-hop key.

Secondly, a DTLS connection to KMF just to receive the EKT key is never used for sending application data. Hence, a direct DTLS connection from KMF to the Endpoint can be thought of a as a redundant option.

Thirdly and most importantly, DTLSTunnel protocol enables the KMF to open just a limited number ports to designated conference servers in the cloud. KMF is usually an entity that belongs to the organization which holds the conference and it is most likely to have security settings such as firewalls protecting its ports. Having multiple DTLS connections open to multiple Endpoints with different IP addresses makes KMF change its security settings every time an Endpoint joins or leaves the conference. Use of a tunnelling protocol removes this complexity and makes the implementation on KMF's side simpler and more effective.

4 Conclusions

As it is mentioned before in this report that use of services hosted in cloud provided by third party companies is a popular choice nowadays for organizations to hold conferences between members located in different locations. The purpose of this thesis was to devise solutions to keep the flow of information through servers in the cloud secure. Here information indicates to audio, video or other form of media exchange that takes place between the participants in the conference. The work has been performed for Ericsson. Its objective was to find the necessary solution through the modification of related protocols of cloud conferencing and standardize them. The purpose of this thesis has been fulfilled by proposing solutions and modifications related to SRTP protocols to facilitate secure exchange of information for the use case defined in the thesis.

4.1 Results and Evaluation

The methodology mentioned in Section 1.5 has been followed step by step to develop the solution for certain parts of SRTP as well as DTLS-SRTP so that they fulfill the requirements for making cloud conferencing end-to-end secure.

Firstly, already existed solutions proposed by Ericsson and Cisco for securing RTP payloads which can be applicable to cloud conferencing scenario have been scrutinized and their drawbacks have been discussed.

Secondly, effective mechanisms to secure information carried in RTCP packets and RTP header extensions which should not be revealed to the cloud have been proposed in the thesis. Security of both RTCP and RTP header extensions are part of a broader end-to-end SRTP security solution for the use case of cloud conference.

Thirdly, needed modifications of DTLS-SRTP specifically for cloud conferencing are pointed out in the thesis and supported protocols are elaborated and offered as solutions. These proposals are solutions for a protocol that handles the special key management needed for scenarios such as multimedia conferences held in cloud.

The above deliverables from the thesis fulfill each of the requirements mentioned in Section 2.1.5. The discussion below shows how they are fulfilled.

- The proposals from Ericsson and Cisco provide mechanisms that fulfill the requirements to ensure end-to-end confidentiality and source authentication of media carried in RTP packets. These mechanisms also allow the Media Switching Server to make needed changes to certain RTP header fields and extensions. These ideas are discussed, reviewed in detail and there drawbacks were pointed out in this thesis.

- Authenticated encryption algorithm, AES-GSM is used in the designs of SRTCP and RTP header extensions proposed in this thesis to achieve security and authentication of their respective contents in end-to-end

manner. Using two new fields which are PUV and SSS makes sure that MSS is free to change the corresponding hop-by-hop values which are respectively SEQ and SSRC. The segregation of end-to-end security and hop-by-security in the design of RTP header extension also fulfills the requirement of MSS changing certain header extensions in a secure manner. Both of these have been important design requirements for this thesis.

- The third requirement is to allow MSS to generate its own control packets. This has been possible by introducing a new packet type ENC to keep all the end-to-end secured packets under a single packet type. This design choice enables MSS to generate control packets which are processed from hop to hop and are separated from ENC packets which are processed end to end.

- The designs have been done in a way so they support RTP topologies which require to change SSRC field of SRTP and SRTCP packets. For these particular topologies, the field SSS is included in the end-to-end part of these packets which allows MSS to change SSRC freely. This fulfills another requirement of secure cloud conference architecture.

- The end-to-end security of SRTCP and RTP header extension is based on end-to-end key negotiation. This thesis proposes how to perform this negotiation using DTLS-SRTP. The proposed solutions consist of alternatives where it is possible to do the negotiation via MSS without addressing the endpoints directly. The thesis proposes a mechanism of having tunneled DTLS connection to achieve a structure with a centralized media switching server which takes care of passive key negotiation as well as media and control packet switching between the endpoints. This accomplishes the requirement for reduction of resource consumption on part of the endpoints by supporting a structure with a centralized media distribution point.

- The solutions proposed in this thesis are designed solely for cloud based environments. For each of the design, the security questions were raised with respect to a scenario of arranging a media conference using servers situated in the cloud. To prevent having data and information leaked to cloud based environment, end-to-end and hop-by-hop processing are separated for media, control information and key negotiation which are reflected in all the proposed solutions.

- Hop by hop replay protection is made mandatory for SRTP and SRTCP packets. After each packet is authenticated, their unique packet index values are logged so that particular packet cannot be replayed by an adversary. The thesis also proposes mechanism to provide end-to-end encryption for parts of SRTCP packets and certain RTP header extensions if they ae required by endpoints. Both are achieved by encapsulating multiple packets or extensions in one packet type or one extension respectively.

- While establishing DTLS-SRTP for end-to-end key negotiation, it is a requirement to preserve the secrecy of end-to-end key from the cloud or other third parties. It is achieved by maintaining a dedicated DTLS-SRTP connection between the key management server and the endpoint. In the event of DTLS messages flowing through the MSS in the middle to reach either side, the end-toe-end key still remains secret. The solution introduces a new content type called DTLSTunnel that keeps the end-to-key secured using DTLS keys only known to the KMF and the negotiating endpoint in the conference.

- It is a requirement to end-to-end authenticate the values of SSRC, SEQ and ROC if they are used in end-to-end security. The solutions described in this thesis use end-to-end PUV as a mandatory replacement for SEQ for keeping track of end-to-end portions of SRTP or SRTCP packets. In the same way end-to-end SSS can be used optionally as an alternative for SSRC inside the end-to-end parts. The inclusion of these fields make sure that end-to-end authentication of SSRC, SEQ and ROC values are not needed. In the event of SSS is not used, SSRC values must remain unchanged throughout the transmission and needs to be end-to-end authenticated.

The above deliverables from the thesis can be employed to strengthen the security of using cloud conferencing services by helping preserve the confidentiality of both corporate and personal information. In many cases, certain information related to RTP payload (for example: payload type), certain RTP header extensions (for example: header extensions with audio level indication) etc. need to be shared with the conference servers in the cloud. Jeopardizing such information can lead to disruption of the conference. But the conference server is semi-trusted to handle such information and also it is the responsibility of the third party cloud operator to provide its services to its clients without any disruption.

The solutions proposed in thesis show that end-to-end privacy in cloud conferencing is achievable and there are alternative ways to make that happen. The solutions vary with complexities of their implementations. The constraints lie with how much compatible the proposed modifications are with the current standards and how much changes need to be performed in the existing specifications of SRTP and DTLS-SRTP.

4.2 Workability

It is now important to discuss why these solutions should work if they are implemented in practical scenarios. The primary security feature of cloud conference is to ensure end-to-end security. This has been made possible in the proposed designs by using a security protocol, EKT to facilitate carrying the end-to-end key encrypted with a separate key in SRTP/SRTCP packets. A trusted designated server (KMF) is responsible for distributing the EKT keys to conference users upon proper authentication. The success of retaining privacy of SRTP packets from cloud providers or others external networks depends on

effectivity of EKT and robustness of KMF. The use of EKT has already been established as a working protocol in securing products such as Cisco Tele Presence Products [25]. As long as the key management servers and the Endpoints do not get compromised and reveal their secret keys, the privacy of the SRTP packets remain secure. The end-to-end authentication is also guaranteed using same procedure of key separation. Thus, the primary requirement of cloud conferencing for end-to-end privacy and source authentication is made practically viable.

The solutions for providing privacy for SRTCP packets also follow the same techniques as SRTP packets. Apart from the security aspect, the other changes include compatibility with different topologies by keeping the source identification field (SSS) optional. The design changes are related to making the transition toward cloud conference less complex. The effectivity of such changes can be tested with practical implementations in the future. The same discussion applies for designs proposed for RTP-header extensions.

Two new mechanisms of DTLS-SRTP for cloud conferencing have been proposed in this thesis. The first one is based on direct communication between the KMF and the Endpoint. This mechanisms establishes a secure path for EKT key transfer and the security of the path is ensured by the standard TLS mechanism mentioned in RFC 5246 [20]. No new security mechanism has been introduced for this particular method.

The second mechanism involves a tunneled DTLS-SRTP session between the KMF and the endpoint with MSS in the middle. The key implementation of this design is to use a new TLS content type. Similar mechanism has already been used in an established protocol known as Heartbeat protocol mentioned in RFC 6520 [26]. The tunneled connection depends on correct transmission of DTLS packets by MSS between a KMF and an Endpoint. This dependency should not be categorized as a security risk as MSS is already considered as a semi trusted entity for responsible forwarding of media transmission in this use case. The privacy and authenticity of the tunneled DTLS contents that include EKT secrets are maintained by building a secure channel within another secure channel using standardized TLS security structures.
From the above discussion, it is understood that the proposed designs are built over already standardized protocols e.g. SRTP & TLS, which are widely used and tested. Thus, the known security attacks and vulnerabilities of these protocols which are listed in their designated RFCs also affect the execution of the proposed solutions. Another major point of failure for these solutions is the compromise of the trusted devices. Such devices in this use case are the KMF and the participating Endpoints in the conference which have full access to the end-to-end keys. Having adversaries taken over the physical devices and their storages is a common risk for any network security protocol. Apart from the security concerns, other modifications proposed in the solutions, for example, adding new fields or packet types can be made more efficient, if needed, with practical implementations which was out of scope for this particular thesis.

4.3 Future Work

This thesis only worked on finding solution to make it work with DTLS-SRTP. The future work on this topic should focus on making it compatible with other key management protocols and web architectures or signalling systems such as WebRTC (Web Real-Time Communication), SIP (Session Initiation Protocol) etc.

The importance of having such solutions made IETF to create a separate working group called PERC (Privacy Enhancing RTP Conferencing) to work on the specific trust model required in multiparty online conferences [21].

References

[1] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
[2] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.
[3] McGrew, D., Rescorla, E., "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
[4] Jones, P. et al., "Requirements for Private Media in a Switched Conference" https://tools.ietf.org/html/draft-jones-avtcore-private-media-reqts-01, March 2015.
[5] Westerlund, M., Wenger, S., "RTP topologies", RFC 5117, January 2008.
[6] Cheng, Y., Mattsson, J., Naslund, M., "Secure Real-time Transport Protocol (SRTP) for Cloud Services", https://tools.ietf.org/html/draft-cheng-avtcore-srtp-cloud-00, March 2015.
[7] Jones, P., Ismail, N., Benham, D., "A Solution Framework for Private Media in a Switched Conferencing", https://tools.ietf.org/html/draft-jones-avtcore-private-media-framework-01, March 2015.
[8] Rescorla, E., "WebRTC Security Architecture" https://tools.ietf.org/html/draft-ietf-rtcweb-security-arch-11, March 2015.
[9] Jones, P. et al., "Private Media Requirements in Privacy Enhanced RTP Conferencing", https://tools.ietf.org/html/draft-jones-perc-private-media-reqts-00, July, 2015.
[10]Westerlund, M., Wenger, S., "RTP topologies" https://tools.ietf.org/html/draft-ietf-avtcore-rtp-topologies-update-09, July, 2015.
[11] Schulzrinne, H., Casner, S., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 3551, July, 2003.
[12] McGrew, D., Igoe, K., "AES-GCM Authenticated Encryption in Secure RTP (SRTP)", https://tools.ietf.org/html/draft-ietf-avtcore-srtp-aes-gcm-17, June, 2015.
[13] Rescorla, E., Modadugu, N., "Datagram Transport Layer Security", RFC 4347, April, 2006.
[14] McGrew, D. et al., "Encrypted Key Transport for Secure RTP", https://tools.ietf.org/html/draft-ietf-avt-srtp-ekt-03, October, 2011.
[15] Jones, P., Ismail, N., Benham, D., "A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing", https://tools.ietf.org/html/draft-jones-perc-private-media-framework-00, July, 2015.
[16] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, April, 2013.
[17] Singer, D., Desineni, H., "A General Mechanism for RTP Header Extensions", RFC 5285, July, 2008.
[18] Lennox, J. et al., "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, December, 2011.
[19] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January, 2011.
[20] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August, 2008.

[21] Perc Status Pages, IETF, https://tools.ietf.org/wg/perc/, June, 2015.
[22] "Video Conferencing -FAQ", *The university of Iowa*. [Online]. Available: http://its.uiowa.edu/support/article/100451#whatisvideoconferencing [Accessed: 03-December-2015].
[23] M. Khalifa and M. Abdellaoui, "Video conference Android platform by your mobile phone" in *International Journal of Advanced Computer Science and Applications*, Vol. 6, No. 6, pp. 276, 2015.
[24] L. Vaquero, et al., "A Break in the Clouds: Towards a Cloud Definition" in *ACM SIGCOMM Computer Communication Review*, Vol. 39, No. 1, pp. 50-51, January, 2009.
[25] "Encrypted Key Transport (EKT) and CTMS Secure Communications", [Online]. Available: http://www.cisco.com/c/en/us/td/docs/telepresence/security_solutions/1_8/CTSS/ctss_app_b.html [Accessed: 1-January-2016]
[26] Seggelmann, R., Tuexen, M., Williams, M., "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February, 2012.
[27] R. Glitho, "Cloud-based Multimedia Conferencing: Business Model, Research Agenda, State-of-the-Art" in *IEEE Conference on Commerce and Enterprise Computing*, 2011, pp. 226.
[28] C. Deyan, H. Zhao, "Data Security and Privacy Protection Issues in Cloud Computing" in *International Conference on Computer Science and Electronics Engineering*, 2012, pp. 648.