Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 19, 2014

M. Westerlund
Ericsson
C. Perkins
University of Glasgow

January 15, 2014

Options for Securing RTP Sessions
draft-ietf-avtcore-rtp-security-options-10

Abstract

The Real-time Transport Protocol (RTP) is used in a large number of different application domains and environments. This heterogeneity implies that different security mechanisms are needed to provide services such as confidentiality, integrity and source authentication of RTP/RTCP packets suitable for the various environments. The range of solutions makes it difficult for RTP-based application developers to pick the most suitable mechanism. This document provides an overview of a number of security solutions for RTP, and gives guidance for developers on how to choose the appropriate security mechanism.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 19, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Background 4
2.1. Point-to-Point Sessions 4
2.2. Sessions Using an RTP Mixer 4
2.3. Sessions Using an RTP Translator 5
2.3.1. Transport Translator (Relay) 5
2.3.2. Gateway 6
2.3.3. Media Transcoder 7
2.4. Any Source Multicast 7
2.5. Source-Specific Multicast 7
3. Security Options 9
3.1. Secure RTP 9
3.1.1. Key Management for SRTP: DTLS-SRTP 11
3.1.2. Key Management for SRTP: MIKEY 13
3.1.3. Key Management for SRTP: Security Descriptions . . . 14
3.1.4. Key Management for SRTP: Encrypted Key Transport . . 15
3.1.5. Key Management for SRTP: ZRTP and Other Solutions . . 15
3.2. RTP Legacy Confidentiality 16
3.3. IPsec 16
3.4. RTP over TLS over TCP 16
3.5. RTP over Datagram TLS (DTLS) 17
3.6. Media Content Security/Digital Rights Management 17
3.6.1. ISMA Encryption and Authentication 18
4. Securing RTP Applications 18
4.1. Application Requirements 19
4.1.1. Confidentiality 19
4.1.2. Integrity 20
4.1.3. Source Authentication 20
4.1.4. Identifiers and Identity 22
4.1.5. Privacy 23
4.2. Application Structure 23
4.3. Automatic Key Management 24
4.4. End-to-End Security vs Tunnels 24
4.5. Plain Text Keys 24
4.6. Interoperability 25
5. Examples 25
5.1. Media Security for SIP-established Sessions using DTLS-SRTP 25
5.2. Media Security for WebRTC Sessions 26

5.3. IP Multimedia Subsystem (IMS) Media Security 27
5.4. 3GPP Packet Based Streaming Service (PSS) 28
5.5. RTSP 2.0 29
6. IANA Considerations 29
7. Security Considerations 30
8. Acknowledgements 30
9. Informative References 30
Authors' Addresses 35

1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] is widely used in a large variety of multimedia applications, including Voice over IP (VoIP), centralized multimedia conferencing, sensor data transport, and Internet television (IPTV) services. These applications can range from point-to-point phone calls, through centralised group teleconferences, to large-scale television distribution services. The types of media can vary significantly, as can the signalling methods used to establish the RTP sessions.

This multi-dimensional heterogeneity has so far prevented development of a single security solution that meets the needs of the different applications. Instead significant number of different solutions have been developed to meet different sets of security goals. This makes it difficult for application developers to know what solutions exist, and whether their properties are appropriate. This memo gives an overview of the available RTP solutions, and provides guidance on their applicability for different application domains. It also attempts to provide indication of actual and intended usage at time of writing as additional input to help with considerations such as interoperability, availability of implementations etc. The guidance provided is not exhaustive, and this memo does not provide normative recommendations.

It is important that application developers consider the security goals and requirements for their application. The IETF considers it important that protocols implement secure modes of operation and makes them available to users [RFC3365]. Because of the heterogeneity of RTP applications and use cases, however, a single security solution cannot be mandated [I-D.ietf-avt-srtp-not-mandatory]. Instead, application developers need to select mechanisms that provide appropriate security for their environment. It is strongly encouraged that common mechanisms are used by related applications in common environments. The IETF publishes guidelines for specific classes of applications, so it is worth searching for such guidelines.

The remainder of this document is structured as follows. Section 2 provides additional background. Section 3 outlines the available security mechanisms at the time of this writing, and lists their key security properties and constraints. That is followed by guidelines and important aspects to consider when securing an RTP application in Section 4. Finally, we give some examples of application domains where guidelines for security exist in Section 5.

2. Background

RTP can be used in a wide variety of topologies due to its support for point-to-point sessions, multicast groups, and other topologies built around different types of RTP middleboxes. In the following we review the different topologies supported by RTP to understand their implications for the security properties and trust relations that can exist in RTP sessions.

2.1. Point-to-Point Sessions

The most basic use case is two directly connected end-points, shown in Figure 1, where A has established an RTP session with B. In this case the RTP security is primarily about ensuring that any third party can't compromise the confidentiality and integrity of the media communication. This requires confidentiality protection of the RTP session, integrity protection of the RTP/RTCP packets, and source authentication of all the packets to ensure no man-in-the-middle attack is taking place.

The source authentication can also be tied to a user or an end-point's verifiable identity to ensure that the peer knows who they are communicating with. Here the combination of the security protocol protecting the RTP session (and hence the RTP and RTCP traffic) and the key-management protocol becomes important to determine what security claims can be made.

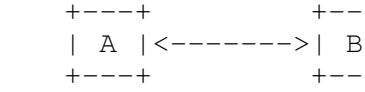


Figure 1: Point-to-point topology

2.2. Sessions Using an RTP Mixer

An RTP mixer is an RTP session-level middlebox that one can build a multi-party RTP based conference around. The RTP mixer might actually perform media mixing, like mixing audio or compositing video images into a new media stream being sent from the mixer to a given participant; or it might provide a conceptual stream, for example the

video of the current active speaker. From a security point of view, the important features of an RTP mixer is that it generates a new media stream, and has its own source identifier, and does not simply forward the original media.

An RTP session using a mixer might have a topology like that in Figure 2. In this example, participants A through D each send unicast RTP traffic to the RTP mixer, and receive an RTP stream from the mixer, comprising a mixture of the streams from the other participants.



Figure 2: Example RTP mixer Topology

A consequence of an RTP mixer having its own source identifier, and acting as an active participant towards the other end-points is that the RTP mixer needs to be a trusted device that has access to the security context(s) established. The RTP mixer can also become a security enforcing entity. For example, a common approach to secure the topology in Figure 2 is to establish a security context between the mixer and each participant independently, and have the mixer source authenticate each peer. The mixer then ensures that one participant cannot impersonate another.

2.3. Sessions Using an RTP Translator

RTP translators are middleboxes that provide various levels of in-network media translation and transcoding. Their security properties vary widely, depending on which type of operations they attempt to perform. We identify three different categories of RTP translator: transport translators, gateways, and media transcoders. We discuss each in turn.

2.3.1. Transport Translator (Relay)

A transport translator [RFC5117] operates on a level below RTP and RTCP. It relays the RTP/RTCP traffic from one end-point to one or more other addresses. This can be done based only on IP addresses and transport protocol ports, with each receive port on the translator can have a very basic list of where to forward traffic. Transport translators also need to implement ingress filtering to

prevent random traffic from being forwarded that isn't coming from a participant in the conference.

Figure 3 shows an example transport translator, where traffic from any one of the four participants will be forwarded to the other three participants unchanged. The resulting topology is very similar to Any Source Multicast (ASM) session (as discussed in Section 2.4), but implemented at the application layer.

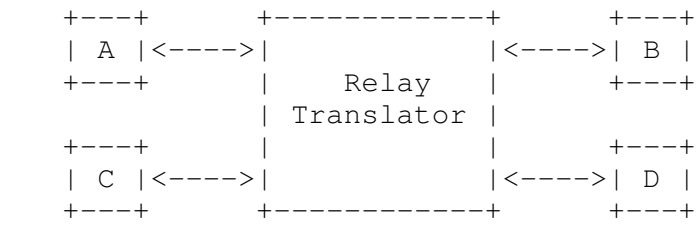


Figure 3: RTP relay translator topology

A transport translator can often operate without needing access to the security context, as long as the security mechanism does not provide protection over the transport-layer information. A transport translator does, however, make the group communication visible, and so can complicate keying and source authentication mechanisms. This is further discussed in Section 2.4.

2.3.2. Gateway

Gateways are deployed when the endpoints are not fully compatible. Figure 4 shows an example topology. The functions a gateway provides can be diverse, and range from transport layer relaying between two domains not allowing direct communication, via transport or media protocol function initiation or termination, to protocol or media encoding translation. The supported security protocol might even be one of the reasons a gateway is needed.



Figure 4: RTP gateway topology

The choice of security protocol, and the details of the gateway function, will determine if the gateway needs to be trusted with access to the application security context. Many gateways need to be trusted by all peers to perform the translation; in other cases some or all peers might not be aware of the presence of the gateway. The security protocols have different properties depending on the degree

of trust and visibility needed. Ensuring communication is possible without trusting the gateway can be strong incentive for accepting different security properties. Some security solutions will be able to detect the gateways as manipulating the media stream, unless the gateway is a trusted device.

2.3.3. Media Transcoder

A Media transcoder is a special type of gateway device that changes the encoding of the media being transported by RTP. The discussion in Section 2.3.2 applies. A media transcoder alters the media data, and thus needs to be trusted with access to the security context.

2.4. Any Source Multicast

Any Source Multicast [RFC1112] is the original multicast model where any multicast group participant can send to the multicast group, and get their packets delivered to all group members (see Figure 5). This form of communication has interesting security properties, due to the many-to-many nature of the group. Source authentication is important, but all participants with access to group security context will have the necessary secrets to decrypt and verify integrity of the traffic. Thus use of any group security context fails if the goal is to separate individual sources; alternate solutions are needed.

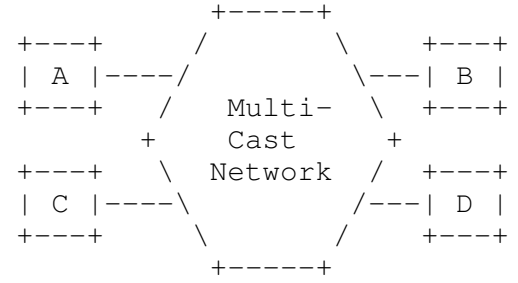


Figure 5: Any source multicast (ASM) group

In addition the potential large size of multicast groups creates some considerations for the scalability of the solution and how the key-management is handled.

2.5. Source-Specific Multicast

Source-Specific Multicast [RFC4607] allows only a specific end-point to send traffic to the multicast group, irrespective of the number of RTP media sources. The end-point is known as the media Distribution Source. For RTP session to function correctly with RTCP over an SSM

session extensions have been defined in [RFC5760]. Figure 6 shows a sample SSM-based RTP session where several media sources, MS1...MSm, all send media to a Distribution Source, which then forwards the media data to the SSM group for delivery to the receivers, R1...Rn, and the Feedback Targets, FT1...FTn. RTCP reception quality feedback is sent unicast from each receiver to one of the Feedback Targets. The feedback targets aggregate reception quality feedback and forward it upstream towards the distribution source. The distribution source forwards (possibly aggregated and summarised) reception feedback to the SSM group, and back to the original media sources. The feedback targets are also members of the SSM group and receive the media data, so they can send unicast repair data to the receivers in response to feedback if appropriate.

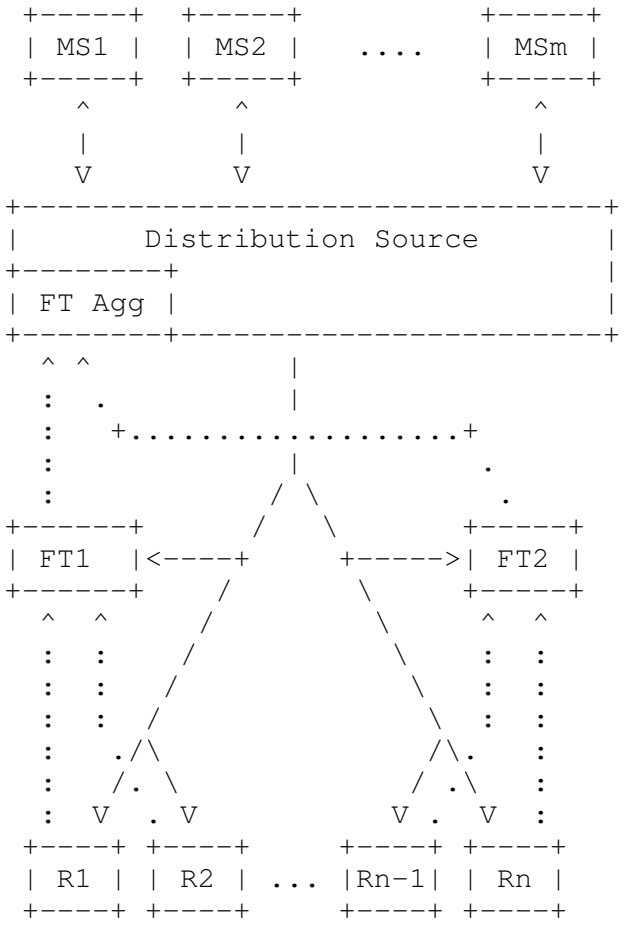


Figure 6: Example SSM-based RTP session with two feedback targets

The use of SSM makes it more difficult to inject traffic into the multicast group, but not impossible. Source authentication

requirements apply for SSM sessions too, and an individual verification of who sent the RTP and RTCP packets is needed. An RTP session using SSM will have a group security context that includes the media sources, distribution source, feedback targets, and the receivers. Each has a different role and will be trusted to perform different actions. For example, the distribution source will need to authenticate the media sources to prevent unwanted traffic being distributed via the SSM group. Similarly, the receivers need to authenticate both the distribution source and their feedback target, to prevent injection attacks from malicious devices claiming to be feedback targets. An understanding of the trust relationships and group security context is needed between all components of the system.

3. Security Options

This section provides an overview of security requirements, and the current RTP security mechanisms that implement those requirements. This cannot be a complete survey, since new security mechanisms are defined regularly. The goal is to help applications designer by reviewing the types of solution that are available. This section will use a number of different security related terms, described in the Internet Security Glossary, Version 2 [RFC4949].

3.1. Secure RTP

The Secure RTP (SRTP) protocol [RFC3711] is one of the most commonly used mechanisms to provide confidentiality, integrity protection, source authentication and replay protection for RTP. SRTP was developed with RTP header compression and third party monitors in mind. Thus the RTP header is not encrypted in RTP data packets, and the first 8 bytes of the first RTCP packet header in each compound RTCP packet are not encrypted. The entirety of RTP packets and compound RTCP packets are integrity protected. This allows RTP header compression to work, and lets third party monitors determine what RTP traffic flows exist based on the SSRC fields, but protects the sensitive content.

SRTP works with transforms where different combinations of encryption algorithm, authentication algorithm, and pseudo-random function can be used, and the authentication tag length can be set to any value. SRTP can also be easily extended with additional cryptographic transforms. This gives flexibility, but requires more security knowledge by the application developer. To simplify things, SDP Security Descriptions (see Section 3.1.3) and DTLS-SRTP (see Section 3.1.1) use pre-defined combinations of transforms, known as SRTP crypto suites and SRTP protection profiles, that bundle together transforms and other parameters, making them easier to use but

reducing flexibility. The MIKEY protocol (see Section 3.1.2) provides flexibility to negotiate the full selection of transforms. At the time of this writing, the following transforms, SRTP crypto suites, and SRTP protection profiles are defined or under definition:

AES-CM and HMAC-SHA-1: AES Counter Mode encryption with 128-bit keys combined with 160-bit keyed HMAC-SHA-1 with 80-bit authentication tag. This is the default cryptographic transform that needs to be supported. The transforms are defined in SRTP [RFC3711], with the corresponding SRTP crypto suite in [RFC4568] and SRTP protection profile in [RFC5764].

AES-f8 and HMAC-SHA-1: AES f8 mode encryption using 128-bit keys combined with keyed HMAC-SHA-1 using 80-bit authentication. The transforms are defined in [RFC3711], with the corresponding SRTP crypto suite in [RFC4568]. The corresponding SRTP protection profile is not defined.

SEED: A Korean national standard cryptographic transform that is defined to be used with SRTP in [RFC5669]. Three options are defined, one using SHA-1 authentication, one using Counter mode with CBC-MAC, and finally one using Galois Counter mode.

ARIA: A Korean block cipher [I-D.ietf-avtcore-aria-srtp], that supports 128-, 192- and 256- bit keys. It also defines three options, Counter mode where combined with HMAC-SHA-1 with 80 or 32 bits authentication tags, Counter mode with CBC-MAC and Galois Counter mode. It also defines a different key derivation function than the AES based systems.

AES-192-CM and AES-256-CM: Cryptographic transforms for SRTP based on AES-192 and AES-256 counter mode encryption and 160-bit keyed HMAC-SHA-1 with 80- and 32-bit authentication tags. These provide 192- and 256-bit encryption keys, but otherwise match the default 128-bit AES-CM transform. The transforms are defined in [RFC3711] and [RFC6188], with the SRTP crypto suites in [RFC6188].

AES-GCM and AES-CCM: AES Galois Counter Mode and AES Counter with CBC MAC for AES-128 and AES-256. This authentication is included in the cipher text which becomes expanded with the length of the authentication tag instead of using the SRTP authentication tag. This is defined in [I-D.ietf-avtcore-srtp-aes-gcm].

NULL: SRTP [RFC3711] also provides a NULL cipher that can be used when no confidentiality for RTP/RTCP is requested. The corresponding SRTP protection profile is defined in [RFC5764].

The source authentication guarantees provided by SRTP depend on the cryptographic transform and key-management used. Some transforms give strong source authentication even in multiparty sessions; others give weaker guarantees and can authenticate group membership but not sources. TESLA [RFC4383] offers a complement to the regular symmetric keyed authentication transforms, like HMAC-SHA-1, and can provide per-source authentication in some group communication scenarios. The downside is need for buffering the packets for a while before authenticity can be verified.

[RFC4771] defines a variant of the authentication tag that enables a receiver to obtain the Roll over Counter for the RTP sequence number that is part of the Initialization vector (IV) for many cryptographic transforms. This enables quicker and easier options for joining a long lived secure RTP group, for example a broadcast session.

RTP header extensions are normally carried in the clear and only integrity protected in SRTP. This can be problematic in some cases, so [RFC6904] defines an extension to also encrypt selected header extensions.

SRTP is specified and deployed in a number of RTP usage contexts; Significant support in SIP-established VoIP clients including IMS; RTSP [I-D.ietf-mmusic-rfc2326bis] and RTP based media streaming. Thus SRTP in general is widely deployed. When it comes to cryptographic transforms the default (AES-CM and HMAC-SHA-1) is the most commonly used, but it might be expected that AES-GCM, AES-192-CM, and AES-256-CM will gain usage in future, especially due to the AES- and GCM-specific instructions in new CPUs.

SRTP does not contain an integrated key-management solution, and instead relies on an external key management protocol. There are several protocols that can be used. The following sections outline some popular schemes.

3.1.1. Key Management for SRTP: DTLS-SRTP

A Datagram Transport Layer Security extension exists for establishing SRTP keys [RFC5763][RFC5764]. This extension provides secure key-exchange between two peers, enabling Perfect Forward Secrecy (PFS) and binding strong identity verification to an end-point. Perfect Forward Secrecy is a property of the key-agreement protocol that ensures that a session key derived from a set of long-term keys will not be compromised if one of the long-term keys is compromised in the future. The default key generation will generate a key that contains material contributed by both peers. The key-exchange happens in the media plane directly between the peers. The common key-exchange procedures will take two round trips assuming no losses. TLS

resumption can be used when establishing additional media streams with the same peer, and reduces the set-up time to one RTT for these streams (see [RFC5764] for a discussion of TLS resumption in this context).

The actual security properties of an established SRTP session using DTLS will depend on the cipher suites offered and used, as well as the mechanism for identifying the end-points of the hand-shake. For example some cipher suits provide PFS , while other do not. When using DTLS, the application designer needs to select which cipher suites DTLS-SRTP can offer and accept so that the desired security properties are achieved. The next choice is how to verify the identity of the peer end-point. One choice can be to rely on the certificates and use a PKI to verify them to make an identity assertion. However, this is not the most common way, instead self-signed certificate are common to use, and instead establish trust through signalling or other third party solutions.

DTLS-SRTP key management can use the signalling protocol in four ways. First, to agree on using DTLS-SRTP for media security. Secondly, to determine the network location (address and port) where each side is running a DTLS listener to let the parts perform the key-management handshakes that generate the keys used by SRTP. Thirdly, to exchange hashes of each side's certificates to bind these to the signalling, and ensure there is no man-in-the-middle attack. This assumes that one can trust the signalling solution to be resistant to modification, and not be in collaboration with an attacker. Finally to provide an assertable identity, e.g. [RFC4474] that can be used to prevent modification of the signalling and the exchange of certificate hashes. That way enabling binding between the key-exchange and the signalling.

This usage is well defined for SIP/SDP in [RFC5763], and in most cases can be adopted for use with other bi-directional signalling solutions. It is to be noted that there is work underway to revisit the SIP Identity mechanism [RFC4474] in the IETF STIR working group.

The main question regarding DTLS-SRTP's security properties is how one verifies any peer identity or at least prevents man-in-the-middle attacks. This do requires trust in some DTLS-SRTP external party, either a PKI, a signalling system or some identity provider.

DTLS-SRTP usage is clearly on the rise. It is mandatory to support in WebRTC. It has growing support among SIP end-points. DTLS-SRTP was developed in IETF primarily to meet security requirements for RTP based media established using SIP. The requirements considered can be reviewed in "Requirements and Analysis of Media Security Management Protocols." [RFC5479].

3.1.2. Key Management for SRTP: MIKEY

Multimedia Internet Keying (MIKEY) [RFC3830] is a keying protocol that has several modes with different properties. MIKEY can be used in point-to-point applications using SIP and RTSP (e.g., VoIP calls), but is also suitable for use in broadcast and multicast applications, and centralized group communications.

MIKEY can establish multiple security contexts or cryptographic sessions with a single message. It is useable in scenarios where one entity generates the key and needs to distribute the key to a number of participants. The different modes and the resulting properties are highly dependent on the cryptographic method used to establish the session keys actually used by the security protocol, like SRTP.

MIKEY has the following modes of operation:

Pre-Shared Key: Uses a pre-shared secret for symmetric key crypto used to secure a keying message carrying the already generated session key. This system is the most efficient from the perspective of having small messages and processing demands. The downside is scalability, where usually the effort for the provisioning of pre-shared keys is only manageable if the number of endpoints is small.

Public Key encryption: Uses a public key crypto to secure a keying message carrying the already-generated session key. This is more resource intensive but enables scalable systems. It does require a public key infrastructure to enable verification.

Diffie-Hellman: Uses Diffie-Hellman key-agreement to generate the session key, thus providing perfect forward secrecy. The downside is high resource consumption in bandwidth and processing during the MIKEY exchange. This method can’t be used to establish group keys as each pair of peers performing the MIKEY exchange will establish different keys.

HMAC-Authenticated Diffie-Hellman: [RFC4650] defines a variant of the Diffie-Hellman exchange that uses a pre-shared key in a keyed HMAC to verify authenticity of the keying material instead of a digital signature as in the previous method. This method is still restricted to point-to-point usage.

RSA-R: MIKEY-RSA in Reverse mode [RFC4738] is a variant of the public key method which doesn’t rely on the initiator of the key-exchange knowing the responder’s certificate. This method lets both the initiator and the responder to specify the session keying

material depending on use case. Usage of this mode requires one round-trip time.

TICKET: [RFC6043] is a MIKEY extension using a trusted centralized key management service (KMS). The Initiator and Responder do not share any credentials; instead, they trust a third party, the KMS, with which they both have or can establish shared credentials.

IBAKE: [RFC6267] uses a key management services (KMS) infrastructure but with lower demand on the KMS. Claims to provides both perfect forward and backwards secrecy.

SAKKE: [RFC6509] provides Sakai-Kasahara Key Encryption in MIKEY. Based on Identity based Public Key Cryptography and a KMS infrastructure to establish a shared secret value and certificate less signatures to provide source authentication. Its features include simplex transmission, scalability, low-latency call set-up, and support for secure deferred delivery.

MIKEY messages have several different transports. [RFC4567] defines how MIKEY messages can be embedded in general SDP for usage with the signalling protocols SIP, SAP and RTSP. There also exist a 3GPP defined usage of MIKEY that sends MIKEY messages directly over UDP [T3GPP.33.246] to key the receivers of Multimedia Broadcast and Multicast Service (MBMS) [T3GPP.26.346]. [RFC3830] defines the application/mikey media type allowing MIKEY to be used in, e.g., email and HTTP.

Based on the many choices it is important to consider the properties needed in ones solution and based on that evaluate which modes that are candidates for ones usage. More information on the applicability of the different MIKEY modes can be found in [RFC5197].

MIKEY with pre-shared keys are used by 3GPP MBMS [T3GPP.33.246] and IMS media security [T3GPP.33.328] specifies the use of the TICKET mode transported over SIP and HTTP. RTSP 2.0 [I-D.ietf-mmusic-rfc2326bis] specifies use of the RSA-R mode. There are some SIP end-points that support MIKEY. The modes they use are unknown to the authors.

3.1.3. Key Management for SRTP: Security Descriptions

[RFC4568] provides a keying solution based on sending plain text keys in SDP [RFC4566]. It is primarily used with SIP and the SDP Offer/Answer model, and is well-defined in point-to-point sessions where each side declares its own unique key. Using Security Descriptions to establish group keys is less well defined, and can have security

issues since it's difficult to guarantee unique SSRCs (as needed to avoid a "two-time pad" attack - see Section 9 of [RFC3711]).

Since keys are transported in plain text in SDP, they can easily be intercepted unless the SDP carrying protocol provides strong end-to-end confidentiality and authentication guarantees. This is not normally the case, where instead hop-by-hop security is provided between signalling nodes using TLS. This leaves the keying material sensitive to capture by the traversed signalling nodes. Thus, in most cases, the security properties of security descriptions are weak. The usage of security descriptions usually requires additional security measures, e.g. the signalling nodes be trusted and protected by strict access control. Usage of security descriptions requires careful design in order to ensure that the security goals can be met.

Security Descriptions is the most commonly deployed keying solution for SIP-based end-points, where almost all end-points that support SRTP also support Security Descriptions. It is also used for access protection in IMS Media Security [T3GPP.33.328].

3.1.4. Key Management for SRTP: Encrypted Key Transport

Encrypted Key Transport (EKT) [I-D.ietf-avtcore-srtp-ekt] is an SRTP extension that enables group keying despite using a keying mechanism like DTLS-SRTP that doesn't support group keys. It is designed for centralized conferencing, but can also be used in sessions where end-points connect to a conference bridge or a gateway, and need to be provisioned with the keys each participant on the bridge or gateway uses to avoid decryption and encryption cycles on the bridge or gateway. This can enable interworking between DTLS-SRTP and other keying systems where either party can set the key (e.g., interworking with security descriptions).

The mechanism is based on establishing an additional EKT key which everyone uses to protect their actual session key. The actual session key is sent in a expanded authentication tag to the other session participants. This key is only sent occasionally or periodically depending on use cases and depending on what requirements exist for timely delivery or notification.

The only known deployment of EKT so far are in some Cisco video conferencing products.

3.1.5. Key Management for SRTP: ZRTP and Other Solutions

The ZRTP [RFC6189] key-management system for SRTP was proposed as an alternative to DTLS-SRTP. ZRTP provides best effort encryption independent of the signalling protocol and utilizes key continuity,

Short Authentication Strings, or a PKI for authentication. ZRTP wasn't adopted as an IETF standards track protocol, but was instead published as an informational RFC. Commercial implementations exist.

Additional proprietary solutions are also known to exist.

3.2. RTP Legacy Confidentiality

Section 9 of the RTP standard [RFC3550] defines a DES or 3DES based encryption of RTP and RTCP packets. This mechanism is keyed using plain text keys in SDP [RFC4566] using the "k=" SDP field. This method can provide confidentiality but, as discussed in Section 9 of [RFC3550], it has extremely weak security properties and is not to be used.

3.3. IPsec

IPsec [RFC4301] can be used in either tunnel or transport mode to protect RTP and RTCP packets in transit from one network interface to another. This can be sufficient when the network interfaces have a direct relation, or in a secured environment where it can be controlled who can read the packets from those interfaces.

The main concern with using IPsec to protect RTP traffic is that in most cases using a VPN approach that terminates the security association at some node prior to the RTP end-point leaves the traffic vulnerable to attack between the VPN termination node and the end-point. Thus usage of IPsec requires careful thought and design of its usage so that it meets the security goals. A important question is how one ensures the IPsec terminating peer and the ultimate destination are the same. Applications can have issues using existing APIs with determining if IPsec is being used or not, and when used who the authenticated peer entity is.

IPsec with RTP is more commonly used as a security solution between infrastructure nodes that exchange many RTP sessions and media streams. The establishment of a secure tunnel between such nodes minimizes the key-management overhead.

3.4. RTP over TLS over TCP

Just as RTP can be sent over TCP [RFC4571], it can also be sent over TLS over TCP [RFC4572], using TLS to provide point-to-point security services. The security properties TLS provides are confidentiality, integrity protection and possible source authentication if the client or server certificates are verified and provide a usable identity. When used in multi-party scenarios using a central node for media distribution, the security provide is only between the central node

and the peers, so the security properties for the whole session are dependent on what trust one can place in the central node.

RTSP 1.0 [RFC2326] and 2.0 [I-D.ietf-mmusic-rfc2326bis] specifies the usage of RTP over the same TLS/TCP connection that the RTSP messages are sent over. It appears that RTP over TLS/TCP is also used in some proprietary solutions that uses TLS to bypass firewalls.

3.5. RTP over Datagram TLS (DTLS)

Datagram Transport Layer Security (DTLS) [RFC6347] is a based on TLS [RFC5246], but designed to work over a unreliable datagram oriented transport rather than requiring reliable byte stream semantics from the transport protocol. Accordingly, DTLS can provide point-to-point security for RTP flows analogous to that provided by TLS, but over an datagram transport such as UDP. The two peers establish an DTLS association between each other, including the possibility to do certificate-based source authentication when establishing the association. All RTP and RTCP packets flowing will be protected by this DTLS association.

Note that using DTLS for RTP flows is different to using DTLS-SRTP key management. DTLS-SRTP uses the same key-management steps as DTLS, but uses SRTP for the per packet security operations. Using DTLS for RTP flows uses the normal datagram TLS data protection, wrapping complete RTP packets. When using DTLS for RTP flows, the RTP and RTCP packets are completely encrypted with no headers in the clear; when using DTLS-SRTP, the RTP headers are in the clear and only the payload data is encrypted.

DTLS can use similar techniques to those available for DTLS-SRTP to bind a signalling-side agreement to communicate to the certificates used by the end-point when doing the DTLS handshake. This enables use without having a certificate-based trust chain to a trusted certificate root.

There does not appear to be significant usage of DTLS for RTP.

3.6. Media Content Security/Digital Rights Management

Mechanisms have been defined that encrypt only the media content, operating within the RTP payload data and leaving the RTP headers and RTCP unaffected. There are several reasons why this might be appropriate, but a common rationale is to ensure that the content stored by RTSP streaming servers has the media content in a protected format that cannot be read by the streaming server (this is mostly done in the context of Digital Rights Management). These approaches then use a key-management solution between the rights provider and

the consuming client to deliver the key used to protect the content and do not give the media server access to the security context. Such methods have several security weaknesses such as the fact that the same key is handed out to a potentially large group of receiving clients, increasing the risk of a leak.

Use of this type of solution can be of interest in environments that allow middleboxes to rewrite the RTP headers and select which streams are delivered to an end-point (e.g., some types of centralised video conference systems). The advantage of encrypting and possibly integrity protecting the payload but not the headers is that the middlebox can't eavesdrop on the media content, but can still provide stream switching functionality. The downside of such a system is that it likely needs two levels of security: the payload level solution to provide confidentiality and source authentication, and a second layer with additional transport security ensuring source authentication and integrity of the RTP headers associated with the encrypted payloads. This can also results in the need to have two different key-management systems as the entity protecting the packets and payloads are different with different set of keys.

The aspect of two tiers of security are present in ISMACryp (see Section 3.6.1) and the deprecated 3GPP Packet Based Streaming Service Annex.K [T3GPP.26.234R8] solution.

3.6.1. ISMA Encryption and Authentication

The Internet Streaming Media Alliance (ISMA) has defined ISMA Encryption and Authentication 2.0 [ISMACryp2]. This specification defines how one encrypts and packetizes the encrypted application data units (ADUs) in an RTP payload using the MPEG-4 Generic payload format [RFC3640]. The ADU types that are allowed are those that can be stored as elementary streams in an ISO Media File format based file. ISMACryp uses SRTP for packet level integrity and source authentication from a streaming server to the receiver.

Key-management for a ISMACryp based system can be achieved through Open Mobile Alliance (OMA) Digital Rights Management 2.0 [OMADRMv2], for example.

4. Securing RTP Applications

In the following we provide guidelines for how to choose appropriate security mechanisms for RTP applications.

4.1. Application Requirements

This section discusses a number of application requirements that need be considered. An application designer choosing security solutions requires a good understanding of what level of security is needed and what behaviour they strive to achieve.

4.1.1. Confidentiality

When it comes to confidentiality of an RTP session there are several aspects to consider:

Probability of compromise: When using encryption to provide media confidentiality, it is necessary to have some rough understanding of the security goal and how long one expect the protected content to remain confidential. National or other regulations might provide additional requirements on a particular usage of an RTP. From that, one can determine which encryption algorithms are to be used from the set of available transforms.

Potential for other leakage: RTP based security in most of its forms simply wraps RTP and RTCP packets into cryptographic containers. This commonly means that the size of the original RTP payload is visible to observers of the protected packet flow. This can provide information to those observers. A well-documented case is the risk with variable bit-rate speech codecs that produce different sized packets based on the speech input [RFC6562]. Potential threats such as these need to be considered and, if they are significant, then restrictions will be needed on mode choices in the codec, or additional padding will need to be added to make all packets equal size and remove the informational leakage.

Another case is RTP header extensions. If SRTP is used, header extensions are normally not protected by the security mechanism protecting the RTP payload. If the header extension carries information that is considered sensitive, then the application needs to be modified to ensure that mechanisms used to protect against such information leakage are employed.

Who has access: When considering the confidentiality properties of a system, it is important to consider where the media handled in the clear. For example, if the system is based on an RTP mixer that needs the keys to decrypt the media, process, and repacketize it, then is the mixer providing the security guarantees expected by the other parts of the system? Furthermore, it is important to consider who has access to the keys. The policies for the handling of the keys, and who can access the keys, need to be considered along with the confidentiality goals.

As can be seen the actual confidentiality level has likely more to do with the application's usage of centralized nodes, and the details of the key-management solution chosen, than with the actual choice of encryption algorithm (although, of course, the encryption algorithm needs to be chosen appropriately for the desired security level).

4.1.2. Integrity

Protection against modification of content by a third party, or due to errors in the network, is another factor to consider. The first aspect that one considers is what resilience one has against modifications to the content. Some media types are extremely sensitive to network bit errors, whereas others might be able to tolerate some degree of data corruption. Equally important is to consider the sensitivity of the content, who is providing the integrity assertion, what is the source of the integrity tag, and what are the risks of modifications happening prior to that point where protection is applied? These issues affect what cryptographic algorithm is used, and the length of the integrity tags, and whether the entire payload is protected.

RTP applications that rely on central nodes need to consider if hop-by-hop integrity is acceptable, or if true end-to-end integrity protection is needed? Is it important to be able to tell if a middlebox has modified the data? There are some uses of RTP that require trusted middleboxes that can modify the data in a way that doesn't break integrity protection as seen by the receiver, for example local advertisement insertion in IPTV systems; there are also uses where it is essential that such in-network modification be detectable. RTP can support both, with appropriate choices of security mechanisms.

Integrity of the data is commonly closely tied to the question of source authentication. That is, it becomes important to know who makes an integrity assertion for the data.

4.1.3. Source Authentication

Source authentication is about determining who sent a particular RTP or RTCP packet. It is normally closely tied with integrity, since a receiver generally also wants to ensure that the data received is what the source really sent, so source authentication without integrity is not particularly useful. Similarly, integrity protection without source authentication is also not particularly useful; a claim that a packet is unchanged that cannot itself be validated as from the source (or some from other known and trusted party) is meaningless.

Source authentication can be asserted in several different ways:

Base level: Using cryptographic mechanisms that give authentication with some type of key-management provide an implicit method for source authentication. Assuming that the mechanism has sufficient strength to not be circumvented in the time frame when you would accept the packet as valid, it is possible to assert a source-authenticated statement; this message is likely from a source that has the cryptographic key(s) to this communication.

What that assertion actually means is highly dependent on the application and how it handles the keys. If only the two peers have access to the keys, this can form a basis for a strong trust relationship that traffic is authenticated coming from one of the peers. However, in a multi-party scenario where security contexts are shared among participants, most base-level authentication solutions can't even assert that this packet is from the same source as the previous packet.

Binding the source and the signalling: A step up in the assertion that can be done in base-level systems is to tie the signalling to the key-exchange. Here, the goal is to at least be able to assert that the source of the packets is the same entity that the receiver established the session with. How feasible this is depends on the properties of the key-management system, the ability to tie the signalling to a particular source, and the degree of trust the receiver places on the different nodes involved.

For example, systems where the key-exchange is done using the signalling systems, such as Security Descriptions [RFC4568], enable a direct binding between signalling and key-exchange. In such systems, the actual security depends on the trust one can place in the signalling system to correctly associate the peer's identifier with the key-exchange.

Using Identifiers: If the applications have access to a system that can provide verifiable identifiers, then the source authentication can be bound to that identifier. For example, in a point-to-point communication even symmetric key crypto, where the key-management can assert that the key has only been exchanged with a particular identifier, can provide a strong assertion about the source of the traffic. SIP identity [RFC4474] provides one example of how this can be done, and could be used to bind DTLS-SRTP certificates used by an end-point to the identity provider's public key to authenticate the source of a DTLS-SRTP flow.

Note that all levels of the system need to have matching capability to assert identifiers. If the signalling can assert that only a given entity in a multiparty session has a key, then the media layer might be able to provide guarantees about the identifier used by the media sender. However, using an signalling authentication mechanism built on a group key can limit the media layer to asserting only group membership.

4.1.4. Identifiers and Identity

There exist many different types of systems providing identifiers with different properties (e.g., SIP identity [RFC4474]). In the context of RTP applications, the most important property is the possibility to perform source authentication and verify such assertions in relation to any claimed identifiers. What an identifier really represent can also vary but, in the context of communication, one of the most obvious is the identifiers representing the identity of the human user one communicates with. However, the human user can also have additional identifiers in a particular role. For example, the human Alice, can also be a police officer and in some cases a identifier for her role as police officer will be more relevant than one that assert that she is Alice. This is common in contact with organizations, where it is important to prove the persons right to represent the organization. Some examples of identifier/Identity mechanisms that can be used:

Certificate based: A certificate is used to assert the identifiers used to claim an identity, by having access to the private part of the certificate one can perform signing to assert ones identity. Any entity interested in verifying the assertion then needs the public part of the certificate. By having the certificate, one can verify the signature against the certificate. The next step is to determine if one trusts the certificate's trust chain. Commonly by provisioning the verifier with the public part of a root certificate, this enables the verifier to verify a trust chain from the root certificate down to the identifier in the certificate. However, the trust is based on all steps in the certificate chain being verifiable and trusted. Thus provisioning of root certificates and the ability to revoke compromised certificates are aspects that will require infrastructure.

Online Identity Providers: An online identity provider (IdP) can authenticate a user's right to use an identifier, then perform assertions on their behalf or provision the requester with short-term credentials to assert the identifiers. The verifier can then contact the IdP to request verification of a particular identifier. Here the trust is highly dependent on how much one

trusts the IdP. The system also becomes dependent on having access to the relevant IdP.

In all of the above examples, an important part of the security properties are related to the method for authenticating the access to the identity.

4.1.5. Privacy

RTP applications need to consider what privacy goals they have. As RTP applications communicate directly between peers in many cases, the IP addresses of any communication peer will be available. The main privacy concern with IP addresses is related to geographical location and the possibility to track a user of an end-point. The main way of avoid such concerns is the introduction of relay (e.g., a TURN server [RFC5766]) or centralized media mixers or forwarders that hides the address of a peer from any other peer. The security and trust placed in these relays obviously needs to be carefully considered.

RTP itself can contribute to enabling a particular user to be tracked between communication sessions if the CNAME is generated according to the RTP specification in the form of user@host. Such RTCP CNAMEs are likely long term stable over multiple sessions, allowing tracking of users. This can be desirable for long-term fault tracking and diagnosis, but clearly has privacy implications. Instead cryptographically random ones could be used as defined by Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs) [RFC7022].

If there exist privacy goals, these need to be considered, and the system designed with them in mind. In addition certain RTP features might have to be configured to safeguard privacy, or have requirements on how the implementation is done.

4.2. Application Structure

When it comes to RTP security, the most appropriate solution is often highly dependent on the topology of the communication session. The signalling also impacts what information can be provided, and if this can be instance specific, or common for a group. In the end the key-management system will highly affect the security properties achieved by the application. At the same time, the communication structure of the application limits what key management methods are applicable. As different key-management have different requirements on underlying infrastructure it is important to take that aspect into consideration early in the design.

4.3. Automatic Key Management

The Guidelines for Cryptographic Key Management [RFC4107] provide an overview of why automatic key management is important. They also provide a strong recommendation on using automatic key management. Most of the security solutions reviewed in this document provide or support automatic key management, at least to establish session keys. In some more long term use cases, credentials might in certain cases need to be be manually deployed.

For SRTP an important aspect of automatic key management is to ensure that two time pads do not occur, in particular by preventing multiple end points using the same session key and SSRC. In these cases automatic key management methods can have strong dependencies on signalling features to function correctly. If those dependencies can't be fulfilled, additional constrains on usage, e.g., per-end point session keys, might be needed to avoid the issue.

When selecting security mechanisms for an RTP application it is important to consider the properties of the key management. Using key management that is both automatic and integrated will provide minimal interruption for the user, and is important to ensure that security can, and will remain, to be on by default.

4.4. End-to-End Security vs Tunnels

If the security mechanism only provides a secured tunnel, for example like some common uses of IPsec (Section 3.3), it is important to consider the full end-to-end properties of the system. How does one ensure that the path from the endpoint to the local tunnel ingress/ egress is secure and can be trusted (and similarly for the other end of the tunnel)? How does one handle the source authentication of the peer, as the security protocol identifies the other end of the tunnel. These are some of the issues that arise when one considers a tunnel based security protocol rather than an end-to-end. Even with clear requirements and knowledge that one still can achieve the security properties using a tunnel based solution, one ought to prefer to use end-to-end mechanisms, as they are much less likely to violate any assumptions made about deployment. These assumptions can also be difficult to automatically verify.

4.5. Plain Text Keys

Key management solutions that use plain text keys, like SDP Security Descriptions (Section 3.1.3), require care to ensure a secure transport of the signalling messages that contain the plain text keys. For plain text keys the security properties of the system depend on how securely the plain text keys are protected end-to-end

between the sender and receiver(s). Not only does one need to consider what transport protection is provided for the signalling message including the keys, but also the degree to which any intermediaries in the signalling are trusted. Untrusted intermediaries can perform man in the middle attacks on the communication, or can log the keys with the result in encryption being compromised significantly after the actual communication occurred.

4.6. Interoperability

Few RTP applications exist as independent applications that never interoperate with anything else. Rather, they enable communication with a potentially large number of other systems. To minimize the number of security mechanisms that need to be implemented, it is important to consider if one can use the same security mechanisms as other applications. This can also reduce problems of determining what security level is actually negotiated in a particular session.

The desire to be interoperable can, in some cases, be in conflict with the security requirements of an application. To meet the security goals, it might be necessary to sacrifice interoperability. Alternatively, one can implement multiple security mechanisms, this however introduces the complication of ensuring that the user understands what it means to use a particular security system. In addition, the application can then become vulnerable to bid-down attack.

5. Examples

In the following we describe a number of example security solutions for applications using RTP services or frameworks. These examples are provided to illustrate the choices available. They are not normative recommendations for security.

5.1. Media Security for SIP-established Sessions using DTLS-SRTP

The IETF evaluated media security for RTP sessions established using point-to-point SIP sessions in 2009. A number of requirements were determined, and based on those, the existing solutions for media security and especially the keying methods were analysed. The resulting requirements and analysis were published in [RFC5479]. Based on this analysis and working group discussion, DTLS-SRTP was determined to be the best solution.

The security solution for SIP using DTLS-SRTP is defined in the Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security

(DTLS) [RFC5763]. On a high level the framework uses SIP with SDP offer/answer procedures to exchange the network addresses where the server end-point will have a DTLS-SRTP enable server running. The SIP signalling is also used to exchange the fingerprints of the certificate each end-point will use in the DTLS establishment process. When the signalling is sufficiently completed, the DTLS-SRTP client performs DTLS handshakes and establishes SRTP session keys. The clients also verify the fingerprints of the certificates to verify that no man in the middle has inserted themselves into the exchange.

DTLS has a number of good security properties. For example, to enable a man in the middle someone in the signalling path needs to perform an active action and modify both the signalling message and the DTLS handshake. There also exists solutions that enables the fingerprints to be bound to identities. SIP Identity provides an identity established by the first proxy for each user [RFC4474]. This reduces the number of nodes the connecting user User Agent has to trust to include just the first hop proxy, rather than the full signalling path. The biggest security weakness of this system is its dependency on the signalling. SIP signalling passes multiple nodes and there is usually no message security deployed, only hop-by-hop transport security, if any, between the nodes.

5.2. Media Security for WebRTC Sessions

Web Real-Time Communication (WebRTC) [I-D.ietf-rtcweb-overview] is a solution providing JavaScript web applications with real-time media directly between browsers. Media is transported using RTP protected using a mandatory application of SRTP [RFC3711], with keying done using DTLS-SRTP [RFC5764]. The security configuration is further defined in the WebRTC Security Architecture [I-D.ietf-rtcweb-security-arch].

A hash of the peer's certificate is provided to the JavaScript web application, allowing that web application to verify identity of the peer. There are several ways in which the certificate hashes can be verified. An approach identified in the WebRTC security architecture [I-D.ietf-rtcweb-security-arch] is to use an identity provider. In this solution the Identity Provider, which is a third party to the web application, signs the DTLS-SRTP hash combined with a statement on the validity of the user identity that has been used to sign the hash. The receiver of such an identity assertion can then independently verify the user identity to ensure that it is the identity that the receiver intended to communicate with, and that the cryptographic assertion holds; this way a user can be certain that the application also can't perform a MITM and acquire the keys to the media communication. Other ways of verifying the certificate hashes

exist, for example they could be verified against a hash carried in some out of band channel (e.g., compare with a hash printed on a business card), or using a verbal short authentication string (e.g., as in ZRTP [RFC6189]), or using hash continuity.

In the development of WebRTC there has also been attention given to privacy considerations. The main RTP-related concerns that have been raised are:

Location Disclosure: As ICE negotiation [RFC5245] provides IP addresses and ports for the browser, this leaks location information in the signalling to the peer. To prevent this one can block the usage of any ICE candidate that isn't a relay candidate, i.e. where the IP and port provided belong to the service providers media traffic relay.

Prevent tracking between sessions: static RTP CNAMEs and DTLS-SRTP certificates provide information that is re-used between session instances. Thus to prevent tracking, such information is ought not be re-used between sessions, or the information ought not sent in the clear. Note, that generating new certificates each time prevents continuity in authentication, however, as WebRTC users are expected to use multiple devices to access the same communication service, such continuity can't be expected anyway, instead the above described identity mechanism has to be relied on.

Note: The above cases are focused on providing privacy from other parties, not on providing privacy from the web server that provides the WebRTC Javascript application.

5.3. IP Multimedia Subsystem (IMS) Media Security

In IMS, the core network is controlled by a single operator, or by several operators with high trust in each other. Except for some types of accesses, the operator is in full control, and no packages are routed over the Internet. Nodes in the core network offer services such as voice mail, interworking with legacy systems (PSTN, GSM, and 3G), and transcoding. End-points are authenticated during the SIP registration using either IMS-AKA (using SIM credentials) or SIP Digest (using password).

In IMS media security [T3GPP.33.328], end-to-end encryption is therefore not seen as needed or desired as it would hinder for example interworking and transcoding, making calls between incompatible terminals impossible. Because of this IMS media security mostly uses end-to-access-edge security where SRTP is terminated in the first node in the core network. As the SIP

signaling is trusted and encrypted (with TLS or IPsec), security descriptions [RFC4568] is considered to give good protection against eavesdropping over the accesses that are not already encrypted (GSM, 3G, LTE). Media source authentication is based on knowledge of the SRTP session key and trust in that the IMS network will only forward media from the correct end-point.

For enterprises and government agencies, which might have weaker trust in the IMS core network and can be assumed to have compatible terminals, end-to-end security can be achieved by deploying their own key management server.

Work on Interworking with WebRTC is currently ongoing; the security will still be end-to-access-edge, but using DTLS-SRTP [RFC5763] instead of security descriptions.

5.4. 3GPP Packet Based Streaming Service (PSS)

The 3GPP Release 11 PSS specification of the Packet Based Streaming Service (PSS) [T3GPP.26.234R11] defines, in Annex R, a set of security mechanisms. These security mechanisms are concerned with protecting the content from being copied, i.e. Digital Rights Management. To meet these goals with the specified solution, the client implementation and the application platform are trusted to protect against access and modification by an attacker.

PSS is RTSP 1.0 [RFC2326] controlled media streaming over RTP. Thus an RTSP client whose user wants to access a protected content will request a session description (SDP [RFC4566]) for the protected content. This SDP will indicate that the media is ISMACryp 2.0 [ISMACryp2] protected media encoding application units (AUs). The key(s) used to protect the media are provided in either of two ways. If a single key is used then the client uses some DRM system to retrieve the key as indicated in the SDP. Commonly OMA DRM v2 [OMADRMv2] will be used to retrieve the key. If multiple keys are to be used, then an additional RTSP stream for key-updates in parallel with the media streams is established, where key updates are sent to the client using Short Term Key Messages defined in the "Service and Content Protection for Mobile Broadcast Services" section of the OMA Mobile Broadcast Services [OMABCAST].

Worth noting is that this solution doesn’t provide any integrity verification method for the RTP header and payload header information, only the encoded media AU is protected. 3GPP has not defined any requirement for supporting any solution that could provide that service. Thus, replay or insertion attacks are possible. Another property is that the media content can be protected by the ones providing the media, so that the operators of

the RTSP server has no access to unprotected content. Instead all that want to access the media is supposed to contact the DRM keying server and if the device is acceptable they will be given the key to decrypt the media.

To protect the signalling, RTSP 1.0 supports the usage of TLS. This is, however, not explicitly discussed in the PSS specification. Usage of TLS can prevent both modification of the session description information and help maintain some privacy of what content the user is watching as all URLs would then be confidentiality protected.

5.5. RTSP 2.0

Real-time Streaming Protocol 2.0 [I-D.ietf-mmusic-rfc2326bis] offers an interesting comparison to the PSS service (Section 5.4) that is based on RTSP 1.0 and service requirements perceived by mobile operators. A major difference between RTSP 1.0 and RTSP 2.0 is that 2.0 is fully defined under the requirement to have mandatory to implement security mechanism. As it specifies how one transport media over RTP it is also defining security mechanisms for the RTP transported media streams.

The security goals for RTP in RTSP 2.0 is to ensure that there is confidentiality, integrity and source authentication between the RTSP server and the client. This to prevent eavesdropping on what the user is watching for privacy reasons and to prevent replay or injection attacks on the media stream. To reach these goals, the signalling also has to be protected, requiring the use of TLS between the client and server.

Using TLS-protected signalling the client and server agree on the media transport method when doing the SETUP request and response. The secured media transport is SRTP (SAVP/RTP) normally over UDP. The key management for SRTP is MIKEY using RSA-R mode. The RSA-R mode is selected as it allows the RTSP Server to select the key despite having the RTSP Client initiate the MIKEY exchange. It also enables the reuse of the RTSP servers TLS certificate when creating the MIKEY messages thus ensuring a binding between the RTSP server and the key exchange. Assuming the SETUP process works, this will establish a SRTP crypto context to be used between the RTSP Server and the Client for the RTP transported media streams.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section can be removed on publication as an RFC.

7. Security Considerations

This entire document is about security. Please read it.

8. Acknowledgements

We thank the IESG for their careful review of [I-D.ietf-avt-srtp-not-mandatory] which led to the writing of this memo. John Mattsson has contributed the IMS Media Security example (Section 5.3).

The authors wished to thank Christian Correll, Dan Wing, Kevin Gross, Alan Johnston, Michael Peck, Ole Jacobsen, Spencer Dawkins, Stephen Farrell, John Mattsson, and Suresh Krishnan for review and proposals for improvements of the text.

9. Informative References

[I-D.ietf-avt-srtp-not-mandatory]
Perkins, C. and M. Westerlund, "Securing the RTP Protocol Framework: Why RTP Does Not Mandate a Single Media Security Solution", draft-ietf-avt-srtp-not-mandatory-14 (work in progress), October 2013.

[I-D.ietf-avtcore-aria-srtp]
Kim, W., Lee, J., Kim, D., Park, J., and D. Kwon, "The ARIA Algorithm and Its Use with the Secure Real-time Transport Protocol(SRTP)", draft-ietf-avtcore-aria-srtp-06 (work in progress), November 2013.

[I-D.ietf-avtcore-srtp-aes-gcm]
McGrew, D. and K. Igoe, "AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP)", draft-ietf-avtcore-srtp-aes-gcm-10 (work in progress), September 2013.

[I-D.ietf-avtcore-srtp-ekt]
McGrew, D. and D. Wing, "Encrypted Key Transport for Secure RTP", draft-ietf-avtcore-srtp-ekt-01 (work in progress), October 2013.

[I-D.ietf-mmusic-rfc2326bis]
Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, "Real Time Streaming Protocol 2.0 (RTSP)", draft-ietf-mmusic-rfc2326bis-38 (work in progress), October 2013.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-08 (work in progress), September 2013.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-07 (work in progress), July 2013.

[ISMACryp2]
Internet Streaming Media Alliance (ISMA), "ISMA Encryption and Authentication, Version 2.0 release version", November 2007.

[OMABCAST]
Open Mobile Alliance, "OMA Mobile Broadcast Services V1.0", February 2009.

[OMADRMv2]
Open Mobile Alliance, "OMA Digital Rights Management V2.0", July 2008.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3640] van der Meer, J., Mackie, D., Swaminathan, V., Singer, D., and P. Gentric, "RTP Payload Format for Transport of MPEG-4 Elementary Streams", RFC 3640, November 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, February 2006.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, July 2006.

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

[RFC4650] Euchner, M., "HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY)", RFC 4650, September 2006.

[RFC4738] Ignjatic, D., Dondeti, L., Audet, F., and P. Lin, "MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", RFC 4738, November 2006.

[RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, January 2007.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5197] Fries, S. and D. Ignjatic, "On the Applicability of Various Multimedia Internet KEYing (MIKEY) Modes and Extensions", RFC 5197, June 2008.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.

[RFC5669] Yoon, S., Kim, J., Park, H., Jeong, H., and Y. Won, "The SEED Cipher Algorithm and Its Use with the Secure Real-Time Transport Protocol (SRTP)", RFC 5669, August 2010.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6043] Mattsson, J. and T. Tian, "MIKEY-TICKET: Ticket-Based Modes of Key Distribution in Multimedia Internet KEYing (MIKEY)", RFC 6043, March 2011.

[RFC6188] McGrew, D., "The Use of AES-192 and AES-256 in Secure RTP", RFC 6188, March 2011.

[RFC6189] Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.

[RFC6267] Cakulev, V. and G. Sundaram, "MIKEY-IBAKE: Identity-Based Authenticated Key Exchange (IBAKE) Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", RFC 6267, June 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6509] Groves, M., "MIKEY-SAKKE: Sakai-Kasahara Key Encryption in Multimedia Internet KEYing (MIKEY)", RFC 6509, February 2012.

[RFC6562] Perkins, C. and JM. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP", RFC 6562, March 2012.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, April 2013.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

[T3GPP.26.234R11] 3GPP, "Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs", 3GPP TS 26.234 11.1.0, September 2012.

[T3GPP.26.234R8] 3GPP, "Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs", 3GPP TS 26.234 8.4.0, September 2009.

[T3GPP.26.346]
3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", 3GPP TS 26.346 10.7.0, March 2013.

[T3GPP.33.246]
3GPP, "3G Security; Security of Multimedia Broadcast/ Multicast Service (MBMS)", 3GPP TS 33.246 12.1.0, December 2012.

[T3GPP.33.328]
3GPP, "IP Multimedia Subsystem (IMS) media plane security", 3GPP TS 33.328 12.1.0, December 2012.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org
URI: http://csperkins.org/