Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 25, 2013

M. Westerlund
Ericsson
C. Perkins
University of Glasgow

October 22, 2012

Options for Securing RTP Sessions
draft-ietf-avtcore-rtp-security-options-01

Abstract

The Real-time Transport Protocol (RTP) is used in a large number of different application domains and environments. This hetrogeneity 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

Copyright Notice

Table of Contents

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. 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, and makes available to the user, secure modes of operation [RFC3365]. Because of the heterogeneity of RTP applications and use cases, however, a single security solution cannot be mandated. 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 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, and combinations of topologies, due to it’s support for unicast, multicast groups, and broadcast topologies, and the existence of 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-points 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 its RTP and RTCP traffic and the key-management protocol becomes important in which security statements one can do.

```
+---+         +---+
| A |<------->| B |
+---+         +---+
```

Figure 1: Point to Point Topology

2.2. Sessions Using an RTP Mixer

An RTP mixer is a an RTP session level middlebox that one can build an 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 featurs 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 examples, participants A-D each send unicast RTP traffic between themselves and the RTP mixer, and receive a RTP stream from the mixer, comprising a mixture of the streams from the other participants.

```
                    +---+      +------------+      +---+
                    | A |<---->|            |<---->| B |
                    +---+      |            |      +---+
                               |    Mixer   |
                    +---+      |            |      +---+
                    | C |<---->|            |<---->| D |
                    +---+      +------------+      +---+
```

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 is part of 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 impersonsate 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 should also 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.

```
+---+      +------------+      +---+
| A |<---->|            |<---->| B |
+---+      |    Relay   |      +---+
           | Translator |
+---+      |            |      +---+
| C |<---->|            |<---->| D |
+---+      +------------+      +---+
```

Figure 3: RTP relay translator topology

A transport translator can often operate without needing to be in 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.

```
+---+      +-----------+      +---+
| A |<---->|  Gateway  |<---->| B |
+---+      +-----------+      +---+
```

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 a trusted part of the application security context or not. 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 so almost certainly needs to be trusted device that is part of 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 in the group security context will have access to the necessary secrets to decrypt and verify integrity of the traffic. Thus use of any symmetric security functions fails if the goal is to separate individual sources within the security context; alternate solutions are needed.

```
                 +-----+
      +---+     /       \    +---+
      | A |----/         \---| B |
      +---+   /   Multi-  \  +---+
             +    Cast     +
      +---+   \  Network  /  +---+
      | C |----\         /---| D |
      +---+     \       /    +---+
                 +-----+
```

Figure 5: Any Source Multicast 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. That end-point is labelled the Distribution Source in Figure 6. It distributes traffic from a number of RTP media sources, MS1 to MSm. Figure 6 also depicts the feedback part of the SSM RTP session using unicast feedback [RFC5760] from a number of receivers R1..Rn that sends feedback to a Feedback Target (FT) and eventually aggregated and distributed to the group.

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 a non-symmetric verification of who sent the RTP and RTCP packets is needed.

The SSM communication channel needs to be securely established and keyed. In addition one also have the individual unicast feedback that also needs to be secured.

```
                     +-----+  +-----+          +-----+
                     | MS1 |  | MS2 |   ....   | MSm |
                     +-----+  +-----+          +-----+
                        ^        ^                ^
                        |        |                |
                        V        V                V
                    +---------------------------------+
                    |       Distribution Source       |
                    +--------+                        |
                    | FT Agg |                        |
                    +--------+------------------------+
                      ^ ^           |
                      :  .          |
                      :   +...................+
                      :             |          .
                      :            / \          .
                    +------+      /   \       +-----+
                    | FT1  |<----+     +----->| FT2 |
                    +------+    /       \     +-----+
                      ^  ^     /         \     ^  ^
                      :  :    /           \    :  :
                      :  :   /             \   :  :
                      :  :  /               \  :  :
                      :   ./\               /\.   :
                      :   /. \             / .\   :
                      :  V  . V           V .  V  :
                     +----+ +----+     +----+ +----+
                     | R1 | | R2 | ... |Rn-1| | Rn |
                     +----+ +----+     +----+ +----+
```

Figure 6: SSM-based RTP session with Unicast Feedback

3. Security Options

This section provides an overview of a number of currently defined security mechanisms that can be used with RTP.

3.1. Secure RTP

The Secure RTP (SRTP) protocol [RFC3711] is one of the most commonly used mechanisms to provide confidentiality, integrity protection and source authentication 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 entirity 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.

The source authentication guarantees provided by SRTP are highly dependent on the cryptographic transform and key-management scheme used. In some cases all a receiver can determine is whether the packets come from someone in the group security context, and not what group member send the packets. Thus, the source authentication guarantees depend also on the session topology. Some cryptographic transform have stronger authentication properties which can guarantee a given source, even over a multi-party session, e.g. those based on TESLA [RFC4383].

SRTP can easily be extended with additional cryptographic transforms. At the time of this writing, the following transforms are defined or under definition:

AES CM and HMAC-SHA-1: AES Counter Mode encryption with 128 bits keys combined with 128 bits keyed HMAC-SHA1 using 80 or 32 bits authentication tags are the default cryptographic transform which need to be supported. Defined in SRTP [RFC3711].

AES-f8 and HMAC-SHA-1: AES f8 mode encryption with 128 bits keys combined with keyed HMAC-SHA1 using 80 or 32 bits authentication. Defined in SRTP [RFC3711].

TESLA: As a complement to the regular symmetric keyed authentication transforms, like HMAC-SHA1. The TESLA based authentication scheme 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. The TESLA transform for SRTP is defined in [RFC4383].

SEED: An Korean national standard cryptographic transform that is defined to be used with SRTP in [RFC5669]. It has three modes, one using SHA-1 authentication, one using Counter with CBC-MAC, and finally one using Galois Counter mode.

ARIA: An Korean block cipher [I-D.ietf-avtcore-aria-srtp], that supports 128, 192 and 256 bits keys. It also has three modes, Counter mode where combined with HMAC-SHA1 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.

AES-192 and AES-256: cryptographic transforms for SRTP based on AES-192 and AES-256 counter mode encryption and 160-bit keyed HMAC-SHA1 with 80 and 32 bits authentication tags for authentication. Thus providing 192 and 256 bits encryption keys and NSA Suite B included cryptographic transforms. Defined in [RFC6188].

AES-GCM: There is also ongoing work to define AES-GCM (Galois Counter Mode) and AES-CCM (Counter with CBC) authentication 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].

[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 in normally carried in the clear and only integrity protected in SRTP. This can be problematic in some cases, so [I-D.ietf-avtcore-srtp-encrypted-header-ext] defines an extension to also encrypt selected header extensions.

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, including perfect forward secrecy and enabling binding strong identity verification to an end-point. 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, used reducing the setup time to one RTT.

DTLS-SRTP key management can use the signalling protocol in three 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 an DTLS listener to let the parts perform the key-management handshakes that generate the keys used by SRTP. Finally, to exchange hashes of each sides certificates to enable each side to verify that they have connected to the by signalling indicated port and not a man in the middle. That way enabling some 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-directions signalling solutions.

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 possible to use 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 Traffic Generation Key (TGK) that is used to derive the 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 TGK. This system is the most efficient from the perspective of having small messages and processing demands.

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

Diffie-Hellman: Uses Diffie-Hellman key-agreement to generate the TGK, thus providing perfect forward secrecy. The downside is increased resource consumption in bandwidth and processing. 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 responders certificate. This methods lets both the initiator and the responder to specify the TGK material depending on use case. Usage of this mode requires one round trip time.

TICKET: [RFC6043] is a MIKEY extension using trusted centralized key management service and tickets, like Kerberos.

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

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

MIKEY messages has several different defined 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 an 3GPP defined usage of MIKEY that sends MIKEY messages directly over UDP to key the receivers of Multimedia Broadcast and Multicast Service (MBMS) [3GPP.33.246].

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].

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 SDP Offer/Answer, 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 as the SSRC uniqurness property can't be guaranteed.

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 the common use of security descriptions with SIP, where instead hop by hop security is provided between signalling nodes using TLS. This still leaves the keying material sensitive to capture by the traversed signalling nodes. Thus in most cases the security properties of security description are weak.

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 that can’t support group keys, like DTLS-SRTP. It is designed for centralized conferencing, but can also be used in sessions where an 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 encryption cycles on the bridge or gateway.

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 are only sent occasionally or periodically depending on use cases depending on what requirements exist for timely delivery or notification on when the key is needed by someone.

3.1.5. Key Management for SRTP: Other systems

There exist at least one additional SRTP key-management system, namely ZRTP [RFC6189]. This was a candidate for IETF standardization that wasn’t chosen, and was published for information instead. Its properties are somewhat similar to DTLS.

There might exist additional non-IETF defined solutions.

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 of providing confidentiality has extremely weak security properties and is not to be used.

3.3. IPsec

IPsec [RFC4301] can be used independent of 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 really meets the security goals. A important question is how one ensure the IPsec terminating peer and the ultimate destination is the same.

IPsec with RTP is more commonly used as security solution between central nodes in an infrastructure that exchanges many RTP sessions and media streams between the peers. The establishment of a secure tunnel between these peers minimizes the key-management overhead between these two boxes.

3.4. DTLS

Datagram Transport Layer Security (DTLS) [RFC6347] can provide point to point security for RTP flows. The two peers would 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: using DTLS is different to using DTLS-SRTP key management. DTLS-SRTP has the core key-management steps in common with DTLS, but DTLS-SRTP uses SRTP for the per packet security operations, while DTLS uses the normal datagram TLS data protection. When using DTLS, RTP and RTCP packets are completely encrypted with no headers in the clear, while DTLS-SRTP leaves the headers in the clear.

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.

3.5. TLS over TCP

When RTP is 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 then 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.

3.6. Payload-only Security Mechanisms

Mechanisms have been defined that encrypt only the payload of the RTP packets, and leave the RTP headers and RTCP in the clear. There are several reasons why this might be appropriate, but a common rationale is to ensure that the content stored in RTP hint tracks in 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 uses a key-management solution between the rights provider and the consuming client to deliver the key used to protect the content, usually after the appropriate method for charging has happened, and do not include the media server in the security context. Such methods have several security weaknesses such 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 what streams that 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 [3GPP.23.234] solution.

3.6.1. ISMA Encryption and Authentication

The Internet Streaming Media Alliance (ISMA) has defined ISMA Encryption and Authentication 2.0 [ISMACrypt2]. 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 remain confidential. From that, one can determine what encryption algorithm is 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, and details of the RTP and RTCP headers, are 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, and are the keys stored or kept somewhere? 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 consider is what resilience one has against modifications to the content. This can affect what cryptographic algorithm is used, and the length of the integrity tags. However equally, important is to consider 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? 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 advertisment 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 you also want to ensure that what you received is what the claimed source really sent, so source authentication without integrity is not particularly useful. In similar way, although not as definitive, is that integrity without source authentication is also not particular useful: you need to know who claims this packet wasn't changed.

Source authentication can be asserted in several different ways:

Base level: Using cryptographic mechanisms that give authentication with some type of key-management provides 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 that assert the source authenticated statement that this message is most probably from someone that has the cryptographic key to this communication.

What that assertion actually means is highly dependent on the application, and how it handles the keys. In an application where the key-handling is limited to two peers, this can form a basis for a trust relationship to the level that you can state as the traffic is authenticated and matching this particular context, it is coming either from me or from my peer (and I trust that neither has shared the key with anyone else). However, in a multi-party scenario where security contexts are shared among participants, most base-level authentication solution can't even assert that this packet is from the same source as the previous packet.

Binding the Source: 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 be at least be able to assert that the sender of the packets is the same entity that I have established the session with. How feasible this is depends on the properties of the key-management system used, the ability to tie the signalling to a particular peer, and what trust you place on the different nodes involved.

For example, consider a point to point communication system that use DTLS-SRTP using self-signed certificates for key-management, and SIP for signalling. In such a system the end-points for the DTLS-SRTP handshake have securely established keys that are not visible to the signalling nodes. However, as the certificates used by DTLS is not bound to any PKI they can't be verified. Instead, hashes over the certificate are sent over the signalling path. If the signalling can be trusted not to collaborate on performaning a man in the middle attack by modifying the hashes, then the end-points can verify that they have reached the peer they are doing signalling with.

Using Identities: If the applications have access to a system that can provide verifiable identities, then the source authentication can be bound to that identity. 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 identity, can provide a strong assertion about who is sending the traffic.

Note that all levels of the system much have matching capability to assert identity. Having the signalling assert that you include a particular identity in a multi-party communication session where the key-management systems establish keys in a way that one can assert that only the given identity has gotten the key. Using a authentication mechanism build on a group key and otherwise can't provide any assertion who sent the traffic than anyone that got the key provides no strong assertability on the media level than: Someone that has gotten the security context (key) sent this traffic.

4.1.4. Identity

As seen in the previous section, having an identity provider system can benefit the applications by enabling them to do strong assertion between identity and the actual media source. Therefore, the need for identity needs to be considered. However, having identity systems might not be suitable for all types of application, since they require trusted infrastructure.

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 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 Random algorithm for RTP CNAME generation [I-D.rescorla-avtcore-random-cname].

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. 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 the 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 determined for an application. To meet the security goals, it might be necessary to sacrifice interoperability. Alternatively, one can implement multiple security mechanisms, but then end up with an issue of ensuring that the user understands what it means to use a particular security level. 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 RTP using applications, services or frameworks. These examples are provided to show the choices that can be made. 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, and the resulting requirements and analysis were published in [RFC5479]. Based on this analysis, and the working group discussion, DTLS-SRTP was determined to be the best solution, and the specifications were finalized.

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 it 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.

At the basic level 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 the signalling message. There also exist a solution that enables the fingerprints to be bound to identities established by the first proxy for each user [RFC4916]. That reduces the number of nodes the connecting user UA has to trust to the first hop proxy, rather than the full signalling path.

5.2. Media Security for WebRTC Sessions

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

The peers hash of their certificates are provided to a Javascript application that is part of a client server system providing rendezvous services for the ones a given peer wants to communicate with. Thus the handling of the hashes between the peers is not well defined. It becomes a matter of trust in the application. But unless the application and its server is intending to compromise the communication security they can provide a secure and integrity protected exchange of the certificate hashes thus preventing any man-in-the-middle (MITM) to insert itself in the key-exchange.

The web application still have the possibility to insert a MITM. That unless one uses a Identity provider and the proposed identity solution [I-D.rescorla-rtcweb-generic-idp]. 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 which user identity that has been used to sign the hash. The receiver of such a Identity assertion then independently verifies the user identity to ensure that it is the identity it intended to communicate and that the cryptographic assertion holds. That way a user can be certain that the application also can't perform an MITM and that way acquire the keys to the media communication.

In the development of WebRTC there has also been high attention on privacy question. The main concerns that has been raised and are at all related to RTP are:

Location Disclosure: As ICE negotiation 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: RTP CNAMEs and DTLS-SRTP certificates is information that could possibly be re-used between session instances. Thus to prevent tracking the same information can't be re-used between different communication sessions.

Note: The above cases are focused on providing privacy towards other parties than the web service.

5.3. 3GPP Packet Based Streaming Service (PSS)

To be written:

5.4. IPTV

To be written:

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.

9. Informative References

[3GPP.23.234]
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.

[3GPP.33.246]
3GPP, "3G Security; Security of Multimedia Broadcast/ Multicast Service (MBMS)", 3GPP TS 33.246 10.0.0, December 2010.

[I-D.ietf-avt-srtp-not-mandatory]
Perkins, C. and M. Westerlund, "Why RTP Does Not Mandate a Single Security Mechanism", draft-ietf-avt-srtp-not-mandatory-10 (work in progress), July 2012.

[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-00

(work in progress), May 2012.

[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-03 (work in progress), September 2012.

[I-D.ietf-avtcore-srtp-ekt]
McGrew, D., Wing, D., and K. Fischer, "Encrypted Key Transport for Secure RTP", draft-ietf-avtcore-srtp-ekt-00 (work in progress), July 2012.

[I-D.ietf-avtcore-srtp-encrypted-header-ext]
Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-ietf-avtcore-srtp-encrypted-header-ext-02 (work in progress), July 2012.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-04 (work in progress), June 2012.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "RTCWEB Security Architecture", draft-ietf-rtcweb-security-arch-03 (work in progress), July 2012.

[I-D.rescorla-avtcore-random-cname]
Rescorla, E., "Random algorithm for RTP CNAME generation", draft-rescorla-avtcore-random-cname-00 (work in progress), July 2012.

[I-D.rescorla-rtcweb-generic-idp]
Rescorla, E., "RTCWEB Generic Identity Provider Interface", draft-rescorla-rtcweb-generic-idp-01 (work in progress), March 2012.

[ISMACrypt2]
"ISMA Encryption and Authentication, Version 2.0 release version", November 2007.

[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.

[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.

[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.

[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.

[RFC4916] Elwell, J., "Connected Identity in the Session Initiation Protocol (SIP)", RFC 4916, June 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.

[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.

[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.

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