

A comparative analysis of the security aspects of the multimedia key exchange protocols

John Floroiu
Tekelec, Germany
Am Borsigturm 11, 13507 Berlin
+49-30-32513216
John.Floroiu@tekelec.com

Dorgham Sisalem
Tekelec, Germany
Am Borsigturm 11, 13507 Berlin
+49-30-32513240
Dorgham.Sisalem@tekelec.com

ABSTRACT

An IP-based multimedia communication system can be roughly divided into two planes: a signaling plane and a media plane. The signaling plane provides the necessary functions for setting up, controlling and terminating the multimedia sessions. The media plane provides the support for transporting the media content (audio, video, text or applications).

Security mechanisms in the signaling planes address aspects related to user authentication, authorization or anonymization as well as the protection of the signaling messages against eavesdropping, interception and manipulation. The security aspects relevant to the media plane concern the encrypting of the media traffic as well as the efficient and secure exchange of the necessary keying material.

This paper provides a comparative analysis of the security aspects of the most representative key exchange protocols designed for VoIP communication, namely DTLS, ZRTP, MIKEY and SDP. In this context, the key exchange protocols are described in relation to various authentication mechanisms and signaling plane security. Further, a number of possible attacks against these protocols are investigated and, where applicable, mitigation measures are proposed.

Categories and Subject Descriptors

C.2.0 Computer Systems Organization / Computer Communication Networks / General / Security and Protection

General Terms

Security.

Keywords

Real-time IP multimedia communications, Signaling plane security, Media plane security, Key exchange protocol, DoS attack.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IPTCOMM'09, July 7-8, 2009, Atlanta, Georgia, USA.
Copyright 2009 ACM.

1. INTRODUCTION

The IP real-time multimedia architecture is composed of a signaling plane and a media plane that can operate independently from each another; they share, however, in most scenarios the same physical endpoints (terminals). In the signaling plane the Session Initiation Protocol (SIP) [1] is the peer-to-peer protocol used for establishing communications sessions. In the media plane the Real Time Transport Protocol (RTP) [2] is used to transport the real-time data.

Designing a security mechanism to protect the real-time multimedia sessions must in the first place take into consideration the characteristics of this type of applications:

1. There must be no or a minimal increase in the size of the data packets. Therefore the use of additional headers and payload fields that carry per-packet information such as initialization vectors, sequence numbers, security association identifiers and padding must be avoided. Instead, fields that already exist in the RTP payload must be used, whenever possible, to derive these information;
2. Header compression mechanisms must not be hampered. These compression mechanisms are based on the idea that the values of many of the fields contained in the IP, UDP and RTP headers remain unchanged across the datagrams that belong to a certain data flow, while some other fields change in a predictable way. As a result, the values of all those fields can be conveyed only once when the data flow is established and need not be transmitted in the subsequent packets. At the other end, a packet is identified by a "connection ID" that enables the headers to be restored. On wireless and slow point-to-point links, these techniques (see[4] and [5]) reduce the size of the headers from 40 bytes (on IPv4) to just a few bytes. As a result, the headers up to (and including) the RTP header must be left unencrypted;
3. The decryption of the encrypted payloads must be possible despite datagrams being re-ordered or lost;
4. The decryption process must be able to start before the entire datagram is received. Also, pipelining and parallelization of the cryptographic operations must be enabled to the largest possible extent, in order to achieve real-time performance. This has resulted in making the counter mode of operation for the cryptographic algorithm the primary design choice. In counter mode the stream of key bits is generated by encrypting the successive values of a counter initialized to a public random value;

5. The hardware (in terms of number of gates) required by the cryptographic algorithms must be minimized in order to make it suitable for low power mobile terminals. This has to do not only with the design of the cryptographic algorithm but also with the mode in which the algorithm is used: for instance the stream ciphers use only the encryption operation of the algorithm for both the enciphering and deciphering, which results in reducing the silicon footprint by half;
6. The key exchange must be either piggybacked into the signaling messages in form of SDP attributes, or multiplexed onto the media ports. When taking place in the signaling plane, the key exchange must perform in at most one round trip so that it retains compatibility with the SDP offer/answer model ([6]).

All these considerations have an impact on the design of the key exchange protocol, on the structure of the protected datagrams, as well as on the selection of the cryptographic modes and key management aspects. In this context, it may be observed that a general purpose security stack such as IKE/IPsec would lead to suboptimal performance when used for protecting real-time multimedia applications (particularly with regard to points 1, 2 and 6).

2. BACKGROUND

The main components of the IP-based multimedia architecture system are the Session Initiation Protocol (SIP), used for establishing and controlling the multimedia sessions and the Real-Time Transport Protocol (RTP), used for exchanging the media content. The Secure Real-time Transport Protocol (SRTP) ([7]) defines the security mechanisms necessary to protect the RTP traffic. Finally, a key exchange protocol operating either in the signaling plane or in the media plane is necessary to provide the keying material required by SRTP.

2.1 Session Initiation Protocol (SIP)

The most important SIP operation is that of inviting new participants to a call. This functionality is achieved by the following SIP entities:

- User Agent: A SIP endpoint acting either as a User Agent Client (UAC), responsible for generating the SIP INVITE request, or as a User Agent Server (UAS), responsible for terminating the SIP signaling and answering the SIP INVITE request. The UA can be the VoIP application running on the user terminal equipment, a VoIP gateway which enables VoIP users to communicate with users in the public switched telephone network (PSTN) or an application server (e.g., a multi-party conferencing server or a voicemail server).
- SIP Proxy: The SIP proxy provides the routing logic of the VoIP service. When a SIP proxy receives a SIP request from a user agent or another SIP proxy it also conducts service specific logic, such as checking the user's profile and whether the user is allowed to use the requested services. The proxy then either forwards the request to another proxy or to another user agent or rejects the request by sending a negative response.

In SIP, a user is identified through a SIP URI in the form of *user-name@domain-name*. This address is resolved to a SIP proxy that

is responsible for the user's domain. To identify the actual location of the user in terms of an IP address, the user needs to register his IP address at the SIP registrar responsible for his domain. Thereby when inviting a user, the caller sends the invitation to the SIP proxy responsible for the user's domain, which looks up the location of the user in the SIP registrar's database and then forwards the invitation to the callee. The callee can either accept or reject the invitation. The session initiation is then finalized by having the caller acknowledging the reception of the callee's answer. During this message exchange, the caller and the callee exchange the transport addresses at which they would like to receive the media and what kind of media they can accept. This information is included in the SIP message bodies using the Session Description Protocol (SDP, [3]). After finishing the session establishment, the end systems can exchange data directly without the involvement of the SIP proxy.

The mechanism for authenticating a user towards a SIP proxy is based on the HTTP Digest, a challenge/response protocol that enables a server to verify that the user knows a certain password.

2.2 Real-Time Transport Protocol

The Real-Time Transport Protocol (RTP) is not a transport protocol in the common sense. That is, it offers no reliability mechanisms, has no notion of a connection and is usually implemented as part of the application rather than the operating system kernel. Instead, RTP offers the application timing and sequencing information that enables the reconstruction of the media stream at the receiving end, as well as the capability to distinguish between different media streams and to produce various statistics describing the quality of the session as seen by other members of the session.

An RTP session consists of two lower-layer data streams, namely the actual data stream for, say, audio or video and a stream of control packets, which is handled by a sub-protocol called the RTP Control Protocol (RTCP). An RTP stream is uniquely identified by the destination transport address and a 32 bit long Synchronization Source (SSRC) and is characterized by a profile, which describes how the specific media is encoded and transported in RTP. Using the RTP Control Protocol (RTCP), each session member periodically sends control packets to all other session members. The control packets contain information about the received and transmitted data rates, delay jitter and packet losses.

2.3 Secure Real-Time Transport Protocol

The Secure Real-time Transport Protocol (SRTP) ([7]) was specifically designed to be suitable for real time data transmission. SRTP specifies a new RTP profile (RTP/SAVP) that offers confidentiality, integrity protection, data origin authentication in case of point-to-point communications, and replay protection to the RTP and RTCP traffic. To achieve this, SRTP defines extensions to the RTP packet formatting that allow the protected RTP and RTCP payloads to be transported. Also, SRTP defines the key derivation procedures used to generate the keying material necessary to protect the data traffic and manages the SRTP cryptographic contexts that specify the mapping between the cryptographic algorithms and keys and the RTP and RTCP streams.

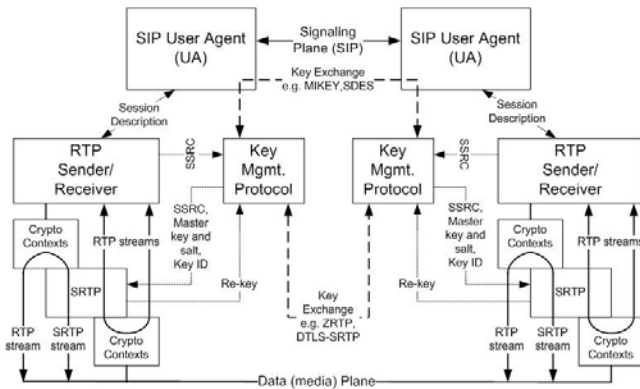


Figure 1. The media security protocol stack

SRTP relies on a key exchange protocol to provide the session keys that SRTP uses to derive the keying material necessary to generate the authentication tags as well as the per-packet keystream that encrypts/decrypts the data traffic. The biggest security threat at the SRTP layer is the keystream reuse, which is a general problem of the stream ciphers, also denoted as a “two-time pad” ([8]). A “two-time pad” occurs when the same session key is used to encrypt different RTP packets that share the same SSRC value and packet index. This may happen in the case of a SSRC collision between different RTP streams or an RTP index roll-over. SSRC collisions may occur in case of the call forking scenario (see section 3.1); even though in practice an UAC would establish a multimedia sessions with only one UAS and would terminate the other forked branches, the theoretical possibility exists for two or more such UASs to pick the same SSRC value and start sending early media. The role of the key exchange protocol in avoiding the keystream reuse is to either allow the peers to signal the SSRC values or provide for a precautionary key negotiation scheme that negotiates a distinct session key per RTP stream. Also, it must be possible for the SRTP subsystem to trigger the key exchange protocol to provide fresh keying material each time the space of the SRTP packet index is close to being exhausted (and hence a roll-over will take place).

Figure 1 illustrates the main components of the media plane security architecture defined by the current IETF specifications. On the sending leg, the outgoing RTP and RTCP datagrams produced by the RTP sender are matched against the SRTP cryptographic contexts before sending them out. On the reception leg, the incoming RTP and RTCP datagrams are matched against the SRTP cryptographic contexts before forwarding them to the RTP receiver. If a match occurs, then the respective SRTP cryptographic context provides the necessary parameters (keys and cryptographic algorithms) that define the type of processing that needs to be performed: enciphering and authentication tag calculation, and respectively authentication tag validation and deciphering.

The key exchange protocols may be classified into signaling plane and media plane key exchange protocols, depending on whether their payloads are piggybacked into the signaling messages or are multiplexed over the media path. Some key management protocols need to interact with the RTP application in order to retrieve the SSRC values for setting up the SRTP cryptographic contexts (while some others perform this binding dynamically). Also, key management protocols should allow

SRTP to trigger re-keying, that is, the re-negotiation of fresh keys. Because the old and new keys coexist for a certain amount of time, the key management protocol must provide the possibility for the SRTP endpoints to announce either the key identifiers, if they are transported in the SRTP packets, or the corresponding SRTP packet index range.

2.4 The Main Key Exchange Protocol Candidates

Before discussing their security properties in more detail, a brief introduction of the following key exchange protocols will be provided: SDES ([9]) and MIKEY ([10], [12], [13]), which operate in the signaling plane and ZRTP ([14]) and DTLS-SRTP ([15]), which operate in the media plane.

MIKEY ([10]) is a signaling plane key exchange protocol where the UAC acts as the initiator and the UAS is the responder. The MIKEY messages are carried in the SDP payloads in form of $a=key-mgmt$ attributes ([11]) that may be specified either at the session or at the media level. MIKEY supports several key exchange modes that make use of either the Diffie-Hellman key exchange or key transport schemes. MIKEY-DH ([10]) and MIKEY-DHMAC ([13]) are examples in the first category. MIKEY-PSK and MIKEY-RSA ([10]) are key transport schemes where the master secret is generated by the initiator and encrypted using the responder’s pre-shared key and respectively public RSA encrypting key. MIKEY-RSA-R ([12]) is a key transport scheme where the master secret is generated by the responder and sent back encrypted with the initiator’s RSA public encrypting key. The MIKEY handshake occurs in one round trip or may consist of one single message sent by the initiator. Along with the keying material, the MIKEY endpoints exchange during the handshake an SSRC map (containing the SSRC values of the media streams that they exchange) as well as the security policies (carrying the other security parameters required to set up the SRTP cryptographic contexts, such as: cryptographic algorithms, key lengths and re-keying rate). With MIKEY, the security policies are pushed by the sender of the media stream toward the receiver. The MIKEY endpoints make use of either pre-shared keys (MIKEY-PSK, MIKEY-DHMAC) or digital signatures (the other key exchange modes) to achieve mutual authentication. The MIKEY key derivation mechanism generates one distinct session key for each media stream specified in the SSRC map.

SDES [9] is a signaling plane key exchange scheme that enables the media sender to specify the session key and the related security parameters (such as the key lifetime, the key index and the cryptographic algorithms) used to protect the media stream. SDES uses for this purpose the media-level $a=crypto$ attribute. SDES is not a key exchange protocol per se, since it provides neither mutual authentication of the SDES endpoints, nor protection of the payloads that it exchanges (the $a=crypto$ attributes). Most notably, SDES also does not provide any means for the participants to exchange the SSRC values of the media streams, which makes it necessary for a “late binding” mechanism to be implemented (see Section 3.2). SDES provides distinct session keys for each media stream in a multimedia session, in each direction.

ZRTP ([14]) is a media plane key exchange protocol that multiplexes the handshake messages over the same port numbers

as the media. A ZRTP endpoint initiates the discovery phase of the ZRTP exchange by sending *Hello* messages, as soon as it learns the media transport address of the peer (advertised in the SDP attributes). The initiator of the ZRTP handshake is the ZRTP endpoint that, following the discovery phase, is the first to initiate the key exchange phase by sending a *Commit* message. A resolution algorithm is provided to address the cases when *Commit* messages are simultaneously sent by the ZRTP endpoints. The key exchange phase consists of an exchange of ephemeral Diffie-Hellman keys and also enables the ZRTP endpoints to identify the “retained secrets” that the endpoints may share from previous sessions. The retained shared secrets are used as part of a key continuity mechanism, which enables the ZRTP endpoints to ensure that a certain key exchange is being performed with a peer with whom communication sessions had previously been established. One distinct ZRTP handshake occurs for each RTP session in a multimedia session. ZRTP defines three key exchange modes: (i) a pre-shared mode, where the ZRTP endpoints derive a master secret from a shared secret that they cached from a previous session, (ii) a Diffie-Hellman mode, where the ZRTP endpoints perform a Diffie-Hellman exchange and use the Diffie-Hellman secret along with retained shared secrets to derive the master secret, and (iii) a multistream mode, where the master secrets of the remaining RTP sessions in a multimedia session are derived from the master secret of the initial RTP session, which was generated using one of the first two key exchange modes. A variety of mutual authentication mechanisms are available in ZRTP: (i) Short Authentication String (SAS) interactive validation, (ii) digital signatures, (iii) end-to-end integrity protected exchange of the $a=zrtp\text{-}hash$ SDP attribute over the signaling path or (iv) a key continuity mechanism (that make use of retained shared secrets, which are updated with every new session established between the peers). The ZRTP key derivation generates distinct secret keys sets for each direction.

DTLS-SRTP ([15]) specifies an extension to the DTLS key exchange protocol ([19]) that enables the participants in a multimedia session to derive the SRTP master key. DTLS-SRTP handshakes take place in the media plane and are initiated by each participant as soon as he learns the media transport address of the peer, thus, the receiver of the SDP media description will act as the client in the DTLS exchange. One distinct DTLS-SRTP handshake will take place for each RTP and RTCP stream. The initial DTLS-SRTP handshake uses a Diffie-Hellman exchange, while the subsequent DTLS-SRTP handshakes will typically make use of the DTLS session resumption mechanism in order to reduce the burden of the cryptographic calculations associated with the Diffie-Hellman key exchange. The need for multiple DTLS-SRTP exchanges may be further reduced if symmetric RTP or RTP-RTCP multiplexing is used. DTLS-SRTP does not offer support for the peers to advertise the SSRC values, which results in a “late binding” mechanism being necessary to map the DTLS associations to the SRTP media streams. DTLS-SRTP has been further extended in [19] to support group key distribution and the exchange of the SSRC values (hence eliminate the need for “late binding”). In the media plane, the DTLS-SRTP endpoints authenticate each other using self-signed certificates. The self-signed certificates are bound to the identities of the endpoints by exchanging a hash value calculated over self-signed certificates in the signaling plane, in form of $a=fingerprint$ SDP attributes.

3. BASIC ASPECTS OF THE KEY EXCHANGE PROTOCOLS

This section addresses two key aspects of the multimedia key exchange protocols: the mutual authentication mechanisms and, in this context, the interdependency between the media plane key exchange protocols and the signaling plane security. It also provides a short overview of the signaling plane of the identities of the participants in a multimedia session.

3.1 Mutual Authentication

The most critical task in a key exchange protocol is the mutual authentication of the peers. The use of a PKI to achieve end-to-end mutual authentication is a largely unpractical option because it requires a terminal to be able to build and validate a certificate chain that could point to any other terminal. In many deployments the mutual authentication between the communication endpoints relies on the subscribers trusting the service providers, who in turn establish (transitive or direct) trust relationships between each other. A PKI-based solution is easier to be mapped onto this latter hop-by-hop transitive model.

An alternative to the PKI for end-to-end mutual authentication schemes is based on the validation of a Short Authentication String (SAS). The SAS-based schemes take advantage of the interactive nature of the real-time multimedia communication and help bootstrapping a security relationship between the communicating endpoints by having one participant read aloud the SAS value displayed by his terminal so that the other participant can compare it with the SAS displayed on his own terminal. The SAS itself represents a hash calculated over key exchange protocol payloads using the just-generated shared secret; therefore, if successfully validated, it guarantees the integrity of the messages and provides key confirmation. The fact that the SAS often consists of a few alphanumeric characters (so that it can be easily read by users) makes it more vulnerable to a man-in-the middle attack where the attacker alters the content of the key exchange messages (e.g. by replacing the public keys of the legitimate participants with his own) in such a way that the resulting SAS collides with the original value. For this reason, SAS authentication is always used in conjunction with a commitment scheme.

ZRTP is one example of a key exchange protocol that uses a commitment-based SAS mutual authentication scheme. The commitment consists of a hash calculated over the ZRTP initiator’s public ephemeral Diffie-Hellman key (g^I) and the ZRTP responder’s *Hello* message. The actual key exchange starts with the initiator sending the commitment, receiving the responder’s public ephemeral Diffie-Hellman key (g^R) and finally sending its own public ephemeral Diffie-Hellman key (g^I). Assuming a man-in-the-middle has substituted g^R by its own g^X , he has to substitute g^I with a g^Y that must be chosen so that two conditions are satisfied: (i) the SAS calculated by the initiator and responder collide and (ii) g^Y yields the same commitment value as g^I (see Figure 2). This is difficult to achieve because the commitment includes the ZRTP initiator’s ephemeral Diffie-Hellman key as well as a ZRTP message that contains a random value generated by the ZRTP responder. The commitment’s value is therefore unpredictable (even if the initiator mistakenly reuses g^I), making it impossible to the attacker to pre-calculate a large number of g^Y keys.

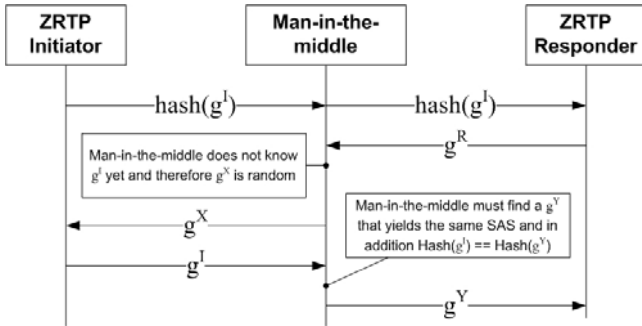


Figure 2. The SAS Commitment Scheme in ZRTP

Assuming that such a g^Y key is found, in the next step the attacker needs to calculate the SAS and check that a collision occurs, otherwise he has to try another g^Y key. This however requires the attacker to calculate the Diffie-Hellman shared secret g^{BY} (which involves an exponentiation and is therefore computationally expensive) for as many g^Y keys, until a SAS collision is detected.

Impersonation attacks on the SAS-based mutual authentication are difficult to mount and are easy to discover as soon as the attacker leaves the communication. The difficulty arises from the fact that the attacker cannot determine in advance when, how and in what context the participants will read the SAS. Moreover, the participants may validate the SAS in a puzzle-solving fashion (that is by suggesting the SAS symbols rather than directly spelling them). The attack therefore may not be as simple as recording the victim saying the symbols in the SAS alphabet during previous calls and then replaying them. Impersonation becomes much easier when the participants do not know their voices in advance, but even in that case the attack will be discovered as soon as during the initial call or in a subsequent call the attacker drops off the conversation or fails to intercept it. Thus, a good practice of establishing the initial contact would be the participants to get to learn each other's voice by establishing several sessions over different networks. It is therefore arguable that voice impersonation is in many scenarios more difficult to realize than substituting a public key and its fingerprint during the session establishment signaling. However, SAS-based mutual authentication brings limited benefit when one of the participants is an automaton, such as an IVR system, or centralized media mixing is taking place, as for instance when a conference bridge is used to interconnect a number of participants.

A SAS extension for TLS (hence applicable to DTLS-SRTP) has been proposed in [19] and consists of the client and server exchanging commitment values (C_C and respectively C_S , see Figure 3) in the *Hello* messages. The commitment values are hash values calculated over random numbers (R_C and respectively R_S), which are revealed after the key exchange has taken place. A man-in-the middle can pre-compute a large number of random numbers that yield the same hash value, which he uses as his commitment towards the client (C_M). He then replaces the public keys exchanged by the client and the server with his own and thus establishes separate master secrets with the client and the server, respectively. Note that this attack works only under the assumption that PKI support is not available and the server and client certificates are self-signed public keys, which is however the scenario that would justify using SAS instead. After completing the handshake on the server leg, the attacker calculates the SAS value and then tries the pre-computed random

numbers one by one until a SAS collision occurs with the client leg. The attacker finally reveals the respective random number. Because the random numbers used in the commitment scheme are not bound to the keying material (and contribute their content only as message payloads), the attacker is not required to do intensive computations (such as exponentiations). The possibility for the attacker to pre-compute data and the relative low complexity of the calculations the attacker has to perform while searching for a collision, make the TLS-SAS extension weaker than the commitment scheme used in ZRTP.

Yet another mutual authentication mechanism is the so-called "baby duck model", which is most commonly used in SSH ([18]). It consists of the peers caching keying material at a time when the participants may safely assume that a man-in-the-middle is not present. The keying material are either the peer's public key or a fingerprint of it (like in case of DTLS-SRTP), or "retained" shared secrets (as in ZRTP). In case of peer-to-peer communications it is however even more difficult than in client-server communications to correctly assume that a man-in-the-middle is not present. Also, when a private key or a shared secret key is compromised, the legitimate owner may be impersonated towards all other endpoints that have the corresponding public key or the matching shared secret key cached. In absence of a centralized system to manage the revocation of keys, the danger of the keys being compromised can only be addressed by continuously updating them, for instance, with every new session. In doing so, the window of opportunity for a successful man-in-the-middle attack in case a key is compromised is reduced, in this case, to only the next session (an approach used in ZRTP).

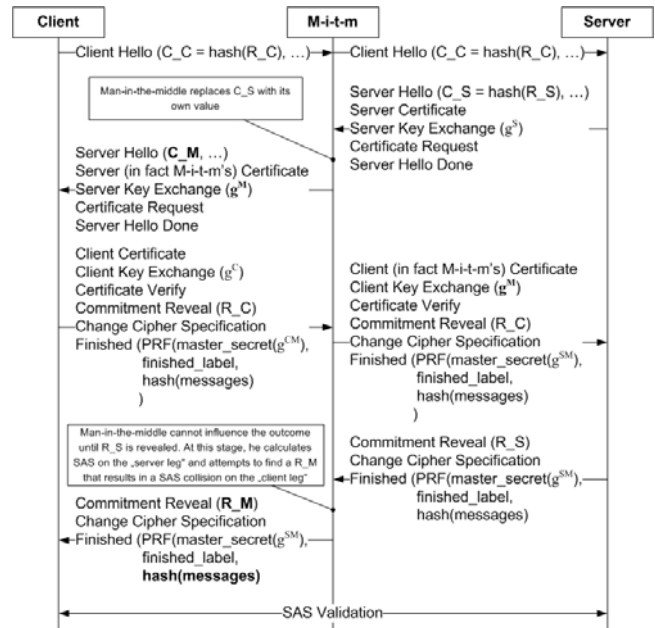


Figure 3 A possible collision attack on TLS-SAS

3.2 Binding the signaling, media and the key exchange planes

The IP multimedia architecture originally defined a signaling plane and a media plane that operate independently. With the introduction of a key exchange protocol we may also identify a key exchange plane, which is defined by the logical entities that

execute the key exchange protocol (see Figure 1). A binding must exist between the logical entities that terminate the three planes of a multimedia session on each communication endpoint.

First, one of the security requirements stated in [17] calls for an association to be established between the endpoints in the signaling, media and key exchange planes. The binding between the media and the key exchange is necessary for the correct mapping of the SRTP cryptographic contexts to the media streams. It is done by means of the SSRC values, which are either explicitly signaled by the key exchange protocol (MIKEY, ZRTP) or is performed by means of a late binding mechanism (SDS, DTLS-SRTP). The late binding mechanism requires the incoming SRTP datagrams carrying “unbound” SSRC values to be authenticated against all SRTP cryptographic contexts. This is necessary because the SSRC value of the same media stream, or the master key that corresponds to the same SSRC may change throughout the lifetime of a session. In case the authentication succeeds, the respective SSRC will be “bound” to the matching SRTP cryptographic context. This introduces an amplification factor for DoS attacks. Arguably, the authentication procedure is not CPU intensive but the overhead involved could be significant if SDS or DTLS-SRTP were deployed on a media gateway that terminates thousands of STRP streams. This problem is solved for DTLS-SRTP by the protocol extension proposed in [19] that enables the SSRC values to be explicitly signaled.

The binding between the signaling and the key exchange plane is necessary in order to ensure that the key exchange occurs between the intended communicating parties. MIKEY uses either digital certificates or pre-shared keys (PSKs) to authenticate the key exchange. In the former case, validating a digitally signed message using the peer’s digital certificate ensures that the message was indeed produced by the respective peer. A binding between the signaling and the key exchange plane can be established if in addition to this also the SIP URIs contained in the *From* header and respectively the *Request-URI* are consistent with the names included in the certificates’ *Subject Name*. When PSKs are employed, the mutual authentication is implicit, in that only the party that posses the matching PSK can produce a valid message authentication code and respectively successfully retrieve the keying material. The identity of the peer is indirectly involved in determining which PSK to use to protect and respectively to authenticate a MIKEY message.

The binding between the signaling and the key exchange planes in case of media plane key exchange protocols is achieved by means of hash values calculated over data exchanged in the media plane and delivered over the signaling plane. In DTLS-SRTP, the endpoints mutually authenticate using self-signed certificates, while fingerprints calculated over the public keys are exchanged in the signaling plane. This scheme is however vulnerable to man-in-the-middle attacks where the attacker has read/write access to both the signaling and the media plane (which in fact can be achieved by having read/write access to the signaling plane in the first place and altering the media transport addresses). Therefore the DTLS-SRTP security relies on the existence of a security infrastructure able to authenticate the identities of the signaling endpoints and to integrity protect the fingerprints.

For the purpose of binding the signaling and the key exchange planes, ZRTP exchanges in the signaling plane the hash values calculated over the key exchange *Hello* messages. A *Hello*

message contains the final hash value (denoted as *H0*) of a hash chain, which ensures that the sequence of key exchange messages is generated by the very same peer with whom the Diffie-Hellman secret is eventually derived. Also this mechanism is vulnerable to man-in-the-middle attacks where the attacker has read/write access to both the signaling and the media plane, unless one of the following end-to-end mutual authentication mechanisms is deployed:

- A security infrastructure to integrity protect the signaling end-to-end, as in the case of DTLS-SRTP.
- Interactive SAS authentication.
- Digital signatures.
- Pre-shared keys (PSKs).

We may therefore conclude that since PSK and PKI-based solutions raise scalability and respectively operational complexity issues when deployed on an end-to-end basis to a large number of terminals that belong to multiple administrative domains, the security of both the signaling plane and the media plane key exchange protocols depend on the availability of a signaling security solution. The only exception is provided by the SAS authentication, which may be used by ZRTP for sessions that involve humans at both ends of the communication.

3.3 Signaling Plane Security

A signaling security solution able to offer protection to a multimedia key exchange protocol must provide identity authentication of the communicating peers and integrity protection of the signaling messages.

SDS in particular requires the signaling to be confidential because the keys are carried in clear text as SDP attributes and therefore read access to the signaling plane is enough for an attacker to get full access to the encrypted communication. Also, a binding between the keying material and the identities of the endpoints cannot be established, unless the messages are authenticated and encrypted end-to-end (e.g. using S/MIME).

Shifting the problem to the signaling plane however does not make the solution any easier since the same drawbacks apply to any PSK or PKI-based solution and, moreover, the signaling must traverse a number of intermediate SIP entities that need to read or even modify various headers and attributes. In the absence of a pure end-to-end solution, a signaling-layer security solution will implement a hop-by-hop model that is mapped onto the transitive trust relationship established between the various interacting entities (intra-domain user terminal-to-proxy and inter-domain proxy-to-proxy). There are mainly two approaches to securing the signaling plane, one based on TLS and another based on the SIP Identity mechanism.

The TLS-based solution ([22]) and consists of:

- UAs establishing a TLS connection with their inbound/outbound SIP proxy. The SIP proxy authenticates towards the UA with a certificate signed by the UA’s service provider, while the UA authenticates towards the SIP proxy using the HTTP Digest mechanism ([1]).
- SIP proxies in different administrative domains authenticating each other using digital certificates established through cross-certification or certification through Bridge CA. When sending a SIP request through

a TLS connection, the domain part of the *Request-URI* is checked to match the *Subject name* of certificate presented by the peer TLS endpoint. When receiving a SIP request through a TLS connection the domain part of the *From* header is checked to match the *Subject name* of the certificate presented by the peer TLS endpoint.

With this approach the signaling can be integrity protected or encrypted on a hop-by-hop basis, enabling the SDP attributes carrying key exchange payloads, public key fingerprints or message hashes to be securely exchanged on the basis of a transitive trust relationship.

Alternatively, a SIP Identity ([20]) infrastructure may be used, which consists of the following elements:

- o An identity service that authenticates and securely communicates with an UAC (e.g. by means of HTTP Digest over TLS). The identity service asserts the identity provided by the UAC in the *From* header of a SIP request (which was previously authenticated) by calculating a signature that covers the *From* header along with a number of other headers and the SDP body (where payloads relevant to the key exchange protocol are transported). The signature is included in an *Identity* header.
- o A verifier that is able to validate the signature provided by the identity service. This involves that the verifier has established a trust relationship with the identity service and as a result he is able to validate and trust the signing digital certificate of the identity service in the first place, e.g. by means of cross-certification. The verifier then securely delivers the SIP request to the UAS (e.g. over TLS).

The advantage of the SIP Identity mechanism over the previously presented TLS-based solution is that it allows the UAC identity to be asserted by the originating domain and validated by the terminating domain.

The identity service and the verifier function would be typically provided by the outbound and respectively inbound SIP proxies providing services to the UAC and respectively to the UAS. Alternatively, they may be implemented by the UAC and respectively the UAS themselves; however, when used in an end-to-end fashion, the support of an end-to-end PKI infrastructure is required, which, if available, would allow the key exchange protocols such as MIKEY and ZRTP to perform the mutual authentication themselves.

The SIP Identity mechanism only authenticates the caller towards the callee, thus providing only a single-sided authentication. Man-in-the-middle attacks can still be detected by the callee, however this is of little use since the caller cannot securely assert the callee's identity so as to make sure that the callee indeed "cooperates". For this reason a SIP Connected Identity ([21]) mechanism must be used in order to authenticate the identity of the UAS towards the UAC. The SIP Connected Identity is essentially the SIP Identity mechanism applied to an in-dialog SIP request sent in the reverse direction (e.g. an UPDATE request sent by the UAS and carrying an *Identity* header that authenticates the identity of the UAS). In addition, the SIP Connected Identity mechanism defines how the content of the *From* and *To* headers

have to be handled so that retargeting scenarios can be accommodated.

An obvious DoS attack on the SIP Identity mechanism itself is flooding a verifier with dialog initiating requests fabricated to appear as arriving from a legitimate identity service, thus forcing the verifier to perform a large number of computationally expensive operations. The solution is the verifier to accept authenticated identity requests only over a TLS connection established with a trusted identity service.

4. DOS ATTACKS USING KEY EXCHANGE PROTOCOLS

One of the easiest ways to mount a DoS attack is to flood the victim with dialog initiating requests, e.g. INVITE requests. In case of MIKEY-RSA, MIKEY-RSA-R or MIKEY-DH, for instance, this will result in the victim performing a large amount of digital signature verification operations. The attacker can achieve this relatively easily by providing a faked *From* header that is consistent with a certificate trusted by the victim, and some random data that looks like a signature, instead of a valid one.

Because the SIP responses are not authenticated, an UAC sending an INVITE may be flooded with *18X* and *200 OK* SIP responses by an attacker that has eavesdropped the INVITE message. This will result in the UAC having to perform a large number of digital signature verifications if MIKEY-RSA-R or MIKEY-DH is used.

In DTLS-SRTP, upon receiving an INVITE request, the victim will initiate a DTLS-SRTP handshake by sending a *Client Hello* message. In order to inflict computationally expensive operations on the victim, the attacker just needs to send one INVITE request and then flood the UAS with *Server Hello*, *Server Certificate*, *Server Key Exchange*, *Certificate Request* and *Server Hello Done* messages. These messages may be generated without requiring the DTLS server (which is the attacker in this scenario) to perform any cryptographic computations over variable data. The DoS consists in case of the ephemeral Diffie-Helman exchange of the client having to validate the *Server Key Exchange* message, which contains a signature that covers the client random, and will therefore presumably fail (since the attacker will not commit resources to provide a valid one). In case of RSA encryption, the client will have to generate a *Client Verify* message for each DTLS session; the handshake will fail with the verification of the server *Finished* message (since the attacker will not bother decrypting the client's secret).

A tentative mitigation of this attack would be "blacklisting" the fingerprints and self-signed certificates that correspond to DTLS key exchanges that fail. However, the problem is that an attacker able to eavesdrop legitimate INVITE and *Client Hello* messages can still flood the victim with a large number of *Server Hello*, *Server Certificate*, *Server Key Exchange*, *Certificate Request* and *Server Hello Done* messages. In order to match the *fingerprint* attribute in the legitimate INVITE message, the *Server Certificate* must be replayed from a valid message that the legitimate UAC sent during a previous call. Note that the self-signed certificate and hence the *fingerprint* attribute are unlikely to change too often (see the "baby-duck security model" described in Section 3.1). Also, the messages sent by the attacker will have to use a spoofed IP address and port number (matching those of the *Client Hello*); note that the DTLS static cookie mechanism only protects

the server against malicious clients and not the other way around. Therefore, when receiving a SIP dialog initiating request, the UAS must allow for a time window during which the UAS will attempt to process all the DTLS sessions that appear to originate from the server, until a valid key exchange is executed.

Finally, a UAC sending an INVITE request may be flooded with a large number of faked *Client Hello* messages and subsequent *Client Certificate*, *Client Key Exchange*, *Client Verify*, *Change Cipher Specification* and *Finished* messages. In this scenario the attacker must be able to eavesdrop the signaling plane and be able to exchange messages in the media plane. The attacker has to use a valid IP address, because the victim (the DTLS server, in this scenario) will possibly respond with a *HelloVerifyRequest* message containing a cookie. If the attacker successfully returns the cookie, the UAC will have to calculate the signature in the *Server Key Exchange* message, if an ephemeral Diffie-Hellman exchange takes place, and to verify the digital signature provided by the attacker in the *Client Verify*. Since the digital signature covers variable data, the attacker will very likely send an invalid one and the victim will terminate the DTLS association before performing more cryptographic calculations to determine the shared secret. Alternatively, the UAC may delay the processing of the DTLS handshake until the *18X* or *200 OK* SIP response that contains the fingerprint of the valid UAS (DTLS client) self-signed certificate, is received. While this saves the victim from performing complex computations, it may lead to memory depletion. Nevertheless, if the *Client Certificate* is a valid replayed message, then the *Client Verify* validation still needs to be performed since the valid sequence of messages cannot be otherwise identified. In this latter case, the receiving of the *18X* or *200 OK* SIP response is only useful if symmetric RTP is used and hence the source transport address of the DTLS handshake can be determined.

When ZRTP is used, flooding a victim with INVITE requests does not inflict any ZRTP-specific processing on the victim in the first stage (the victim just sends back *Hello* messages). The key exchange will be performed with the initiator to whom a SIP response has been returned. Conversely, an UAC sending an INVITE may be flooded with a large number of *18X* or *200 OK* SIP responses and corresponding ZRTP *Hello*, *DHPart1* and *Confirm1* messages by an attacker that has gained control over both the signaling and the media planes. As a result, the UAC will be forced to perform a large number of Diffie-Hellman secret key derivations. The subsequent decryption of the *Confirm1* message will fail since presumably the attacker will not perform himself any Diffie-Hellman secret key derivation. This attack scenario is possible even if the mutual authentication is performed using SAS because the SAS validation takes place after the multimedia session has been established.

We may conclude that, while protecting against a flood with dialog initiating requests is difficult to achieve in a distributed DoS attack situation, it is crucial for the key exchange protocols that the UAS performs the key exchange only with the UAC to whom a SIP response has been returned. Authenticating the SIP responses is also important to avoid DoS attacks against the UAC. The time window for such an attack is however limited because SIP proxies forward only one *18X* SIP response in an increasing response code sequence and forwards subsequent *200 OK* responses only a small time interval following the reception of the

first *200 OK*. Also, exchanging the signaling over TLS connections mitigates against SIP response injection.

4.1 Security Aspects and Weaknesses Introduced by Forking

Call forking is defined as forwarding a dialog initiating request to multiple destinations either in parallel or sequentially, e.g. if the previous destination is busy or unavailable. A call forking will result in the UAC establishing multiple key exchange sessions with the UAS entities that terminate the fork branches. Call forking may lead to keystream reuse if (i) the same master key is used by all the UASs, (ii) a SSRC collision occurs between media streams sent by different UASs and (iii) the packet indexes of the respective media streams overlap.

Key exchange protocols must ensure that the first condition is not fulfilled, which is always the case of the key exchange protocols that make use of the Diffie-Hellman key exchange mechanism (MIKEY-DH, MIKEY-DHMAC, ZRTP, DTLS-SRTP) or enable the UAS to specify the secret keys for the media streams that it sends (MIKEY-RSA-R, SDES). In case of MIKEY-RSA and MIKEY-PSK, the first condition is however satisfied, which involves that forking cannot be guaranteed to be secure with these key exchange protocols because a possible SSRC collision may not be detected before early media is already sent (i.e. before the SIP responses containing the SSRC maps are received).

Another aspect related to call forking concerns the correct mapping of the key exchange sessions to the media streams that correspond to each UAS. The mapping is important even if in the end a multimedia session will be established only with one UAS (while the others will be terminated) because the shared secrets used to protect the media streams will differ for each UAC-UAS pair. The mapping may be done using: (i) the SSRC values, when they are exchanged within the key exchange protocol messages, (ii) the port numbers, when they are shared between the key exchange protocol and the media, or (iii) the binding that exists between key exchange session and the signaling (public key fingerprints or message hashes, see Section 2.2). One notable exception is SDES, which cannot associate the SIP responses with the media sent by each individual UAS that terminates a fork branch. For this reason, when SDES is used, the UAC must update the local media transport endpoint to a distinct value (which will be then used to associate the keying material with the media) for each responding UAS.

Forking is not supported by key exchange protocols that require the UAC to use the peer's public key or pre-shared key for instance in order to encrypt a master secret, unless all the forked targets share the private and respectively the pre-shared keys (this is the case of MIKEY-RSA and MIKEY-PSK). The same problem prevents these key exchange protocols to support retargeting (the process by which a SIP intermediary, such as a proxy server, alters the *Request-URI* of a SIP request before he forwards that request).

Forking has another effect on the key exchange protocols: the SDP answer is basically contained in the *180 Ringing* response but in case of forking the UAC will receive only one *180 Ringing*

(or a sequence of increasing *18X* SIP responses¹) but he may receive media streams from multiple UASs and a corresponding number of final *200 OK* responses. The UAC will typically establish a session with the UAS that first sends a *200 OK* response, which is however not necessary the same from whom the *180 Ringing* has been received, and will terminate the sessions with all the subsequent *200 OK* responders. For a DTLS-SRTP endpoint, this involves that DTLS exchanges that contain client certificates that do not match the *fingerprint* contained in the *18X* responses may in fact be valid. This extends the window of opportunity for DoS attacks to the range of minutes. The UAC is also a potential victim to faked *200 OK* responses followed by *Server Hello*, *Server Certificate*, *Server Key Exchange*,... floods.

5. THE INTERACTION OF THE KEY EXCHANGE PROTOCOLS WITH INTERMEDIATE ENTITIES

While initially conceived to work in an end-to-end manner, the IP multimedia protocols stack was subsequently adapted to fit telecom service models that involved middleware entities. Session Border Controllers (SBCs) are intermediate entities commonly deployed in VoIP infrastructures. SBCs are used to protect, monitor and support the NAT traversal and interworking of enterprise VoIP networks, by controlling both the signaling and the media plane ([23]). In order to achieve this purpose, SBCs often act as back-to-back user agents (B2BUAs). Media protection can still be performed in an end-to-end manner through SBCs provided that they transfer the media payloads unchanged (including the media plane key exchange protocols messages) and that they also do not remove the SDP attributes related to the key exchange. However, SBCs usually work by inserting themselves in the signaling plane key exchange and blocking the media plane key exchange altogether.

Besides the signaling and media controlling function, there are also a number of service scenarios that are incompatible with the end-to-end key exchange model in the sense that either an intermediate entity operating in the media plane requires access to the plaintext media, or a key management server is used to distribute the secret keys to the entities authorized to encrypt/decrypt the media.

A first example is the deferred media delivery, a typical use case of which is the voice mail. In this scenario the communicating parties may be unable to perform a key exchange since, for instance, the callee may be outside of the network coverage. As a result, either the participants establish separate key exchange sessions with the voice mail system, which gives the voice mail system access to the plaintext, or a key management server is used to issue a ticket (that contains the secret key) to the caller and allow the callee to subsequently retrieve the ticket.

Other common examples are the transcoding and the call recording, where an intermediate entity in the media plane needs to have access to the plaintext media being exchanged. These are operations that occur with the knowledge of the participants in the multimedia session. One possible solution would be the

participants to willingly disclose the negotiated shared secrets to the authorized entities, using for instance the mechanism described in [24], where a participant uses the PUBLISH method protected by TLS or S/MIME to send the shared secrets (used in both send and receive directions, as well as their associated parameters including the corresponding SSRCs), encoded with an extended version of SDES, to a third party. The trust model of such an approach requires however a closer consideration because it involves a participant trusting the other participant to disclose the shared secrets to a trusted third-party entity, which however may not be known to the first participant.

Legal interception is yet another example of an operational requirement that is incompatible with the end-to-end key exchange model. Unlike call recording, legal interception must be transparent to the participants the communication between whom is being wiretapped. In IMS for instance, where the signaling infrastructure is under the control of the operators, a signaling entity can easily act as a man-in-the-middle for the end-to-end key exchange protocol or eavesdrop the keys in case they are exchanged in clear (e.g. using SDES). However, even if a SAS authentication mechanism is not used, there is at least a theoretical possibility that the endpoints could retrieve the peer's public key fingerprint, the initial message hash value or the encryption key and compare it orally, or verify the certificate subject name against the peer's identity.

A cleaner alternative to the "shared secrets disclosure" and "transparent man-in-the-middle" approaches described so far could be the Kerberos-like ticket-based model proposed in 3GPP ([25]). This model introduces a trusted third party that distributes secret keys to the entities that are authorized to use them; the difficulty to deploy it however arises from the fact that all the relevant entities that need to access a certain secret key must trust the same key distribution server.

Finally, a key escrow scheme may be used, where network entities do not actively participate in the encrypting and decryption of the data but can determine the secret key used and tap into the data traffic whenever they are authorized to do so. This could be achieved, for instance, if the home network could enforce (e.g. by checking the public key's fingerprint) that a subscriber sends to the other party a public Diffie-Hellman key g^K , where K is derived from the AKA IK and CK keys, which are known to both the subscriber and his home network. By exponentiating the public Diffie-Hellman key received by the subscriber from his peer with the key K that belongs to the subscriber, the home domain will obtain the Diffie-Hellman secret used by the subscriber and his peer for the current session and thus will be able to perform legal interception if authorized to.

6. CONCLUSIONS

The paper has highlighted a number of aspects that are essential to key exchange protocols aimed to protect the media traffic, like the interdependency between the signaling, the media and the key exchange planes. The paper also discusses the strengths and weaknesses of the different multimedia key exchange protocols with regard to a number of aspects mainly related to the support for SAS authentication and vulnerability to DoS attacks. Finally the paper describes the compatibility of the end-to-end media protection paradigm with a number of communication scenarios

¹ E.g. 180, 181, 182, 183. Error codes higher than 183 are currently not specified

and suggests a key escrow that could find applicability in the IMS architecture.

7. REFERENCES

- [1] J. Rosenberg et al., SIP: Session Initiation Protocol, RFC3261, June 2002
- [2] H. Schulzrinne et al., RTP: A Transport Protocol for Real-Time Applications, RFC3550, July 2003
- [3] M. Handley, V. Jacobson, C. Perkins, SDP: Session Description Protocol, RFC4566, July 2006
- [4] C. Bormann, Ed., Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP and uncompressed, RFC3095, July 2001
- [5] T. Koren et al., Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering, RFC3545, July 2003
- [6] J. Rosenberg, H. Schulzrinne, An Offer/Answer Model with the Session Description Protocol (SDP), RFC3264, June 2002
- [7] M. Baugher et al., The Secure Real-time Transport Protocol (SRTP), RFC3711, March 2004
- [8] D. McGrew, S. Fluhrer, Attacks on Additive Encryption of Redundant Plaintext and Implications on Internet Security, Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000. Waterloo, Ontario, Canada, August 2000. Proceedings, pg. 14-28
- [9] F. Andreasen, M. Baugher, D. Wing, Session Description Protocol (SDP) Security Descriptions for Media Streams, RFC4568 July 2006
- [10] J. Arkko, et al., MIKEY: Multimedia Internet KEYing, RFC3830 August 2004
- [11] J. Arkko, et al., Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP), RFC4567, July 2006
- [12] D. Ignjatovic, MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY), RFC4738, November 2006
- [13] M. Euchner, HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY), RFC4650 September 2006
- [14] P. Zimmermann, A. Johnston, J. Callas, ZRTP: Media Path Key Agreement for Secure RTP, draft-zimmermann-avt-zrtp-06, March 2009
- [15] D. McGrew, E. Rescola, Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP), draft-ietf-avt-dtls-srtp-07, February 2009
- [16] D. McGrew, E. Rescola, Short Authentication String Extension for DTLS, draft-mcgrew-tls-sas-00, March 2007 (expired)
- [17] D. Wing, Ed., et al., Requirements and analysis of Media Security Management Protocols, RFC5479, April 2009
- [18] T. Ylonen, C. Lonvick, Ed., The Secure Shell (SSH) Authentication Protocol, RFC4252, January 2006
- [19] D. Wing, DTLS-SRTP Key Transport (KTR), draft-wing-avt-dtls-srtp-key-transport-03, March 2009
- [20] J. Peterson, C. Jennings, Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP), RFC4474, August 2006
- [21] J. Elwell, Connected Identity in the Session Initiation Protocol (SIP), RFC4916, June 2007
- [22] T. Dierks, E. Rescola, The Transport Layer Security (TLS) Protocol, RFC5246, August 2008
- [23] J. Hautakorpi, Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments, draft-ietf-sipping-sbc-funcs-08, January 2009
- [24] D. Wing, et al., Secure Media Recording and Transcoding with the Session Initiation Protocol, draft-wing-sipping-srtp-key-04, October 2008
- [25] 3GPP Technical Specification Group services and System Aspects, IMS media plane security (Release 8), 3GPP TR 33.828 v1.0.0, March 2009