Work in progress: A secure and lightweight scheme for media keying in the Session Initiation Protocol (SIP)

Vijay K. Gurbani  
Acatel-Lucent, Bell Labs  
1960 Lucent Lane,  
Naperville, IL 60566 (USA)  
vkg@research.bell-labs.com  

Vladimir Kolesnikov  
Acatel-Lucent, Bell Labs  
600-700 Mountain Ave.,  
Murray Hill, NJ 07974 (USA)  
kolesnikov@research.bell-labs.com

ABSTRACT
Exchanging keys to encrypt media streams in the Session Initiation Protocol (SIP) has proved challenging. The challenge has been to devise a key transmission protocol that preserves the features of SIP while minimizing key exposure to unintended parties and eliminating voice clipping. We first briefly survey the two IETF SIP media keying protocols – SDES and DTLS-SRTP – and evaluate them against a core feature set. We then introduce a novel simple and lightweight scheme to significantly increase the security of SDES SIP keying with minimal overhead costs. Our proposed key exchange involves only one symmetric key operation by sender and receiver and is secure against the Man-in-the-middle attack unless the attacker is able to intercept both the SIP signaling and media plane traffic. Our key exchange scheme is much simpler than DTLS-SRTP; in fact, compared to SDES, it includes only one additional simple step. At the same time, it provides significantly better security than SDES and is only slightly weaker than the non-PKI version of DTLS-SRTP.

Categories and Subject Descriptors
K.6.5 [Security and Protection]: [Authentication, Unauthorized access]

General Terms
Security

Keywords
SIP, key exchange, media, security, SDES, DTLS

1. INTRODUCTION
The Session Initiation Protocol (SIP [17]) is an Internet protocol to set up, maintain, and terminate multimedia sessions. While SIP is used to rendezvous the session participants, the session itself is conducted using separate protocols. The Session Description Protocol (SDP, [11]), which is transported in SIP is used to describe endpoint capabilities, exchange the voice or video codecs and network identifiers — IP addresses and port numbers — where the media will flow. The media itself, i.e., the actual contents that comprise the voice or video session, use the Real-Time transport Protocol (RTP. [18]).

Because the protocols for initial rendezvous, capability description, and eventual media stream are different, it becomes a challenge to provide security for the system as a whole. As an example of this challenge, consider that signaling in SIP can be protected by hop-by-hop use of Transport Layer Security (TLS [6]), yet the media often flows end-to-end using plaintext RTP. Furthermore, the protection afforded to the signaling messages is such that confidentiality, message authentication and replay protection are ensured on a per hop channel, but the intermediary that forwards the signaling onwards have unhindered access to the plaintext that comprise the signaling messages.

Today, while secure keying techniques (e.g., DTLS-SRTP) are available and standardized, SIP implementations predominantly use (weakly secure) SDES key transmission for securing media-plane communication (see Table 1 for samples collected at SIP interoperability events.) This state of affairs is due to the implementation complexity and increased computation and communication costs associated with the public-key based proposals, such as DTLS-SRTP and ZRTP[24].

Our Contributions and Outline of the Work
We close this security/efficiency gap, by proposing a new media keying protocol that involves only one symmetric key operation by sender and receiver and is secure against man-in-the-middle (MiTM) attack unless the attacker is able to intercept both the SIP signaling and media plane traffic. To match its efficacy against the standardized SIP media keying protocols, we first analyze the two media keying protocols – Security Descriptions (SDES [1]) and DTLS-SRTP [14] for their suitability in a SIP network. We chose to focus on these two protocols primarily because they are standardized by the Internet Engineering Task Force (IETF) and as such will witness large-scale deployment in SIP networks.

To analyze a protocol's ability to successfully key media SIP streams, we list a feature set against which the particular media keying protocols, including our novel contribution, will be evaluated. We will see that our key exchange scheme is much simpler than DTLS-SRTP; in fact, compared to SDES it includes only one additional simple step. At the same time, it provides significantly better security than SDES and is only slightly weaker than the non-PKI
version of DTLS-SRTP.

The paper is structured as follows: Section 2 presents the required background on SIP and SRTP. Section 3 identifies the core feature set that the keying protocols should support. Sections 4 and 5 review SDES and DTLS-SRTP protocols, respectively, and evaluate them on the core feature set. We present our novel keying method, analyze its security, and subject it to the same core feature set evaluation in Section 6. Section 7 provides related work; we conclude in Section 8.

2. THE SESSION INITIATION PROTOCOL

A SIP ecosystem consists of user agents, proxy servers, redirect servers, and registrars. Of special interest to us with respect to this paper are user agents and proxy servers.

2.1 Establishing a SIP Session

There are two types of SIP user agents: a user agent client (UAC) and a user agent server (UAS). A UAC and a UAS are software programs that execute on a computer, an Internet phone, or a personal digital assistant (PDA). A UAC originates requests (i.e., start a multimedia session) and a UAS accepts and acts upon a request. Proxy servers are used to route requests and responses between a UAC and a UAS.

SIP invests a great amount of trust in the proxies, as we will see later in this paper. In the canonical SIP trapezoid [17], Alice wishes to establish a session with Bob. Her SIP request to establish a session traverses through her proxy to Bob’s proxy. Bob’s proxy performs a lookup service to determine where Bob can be located, and forwards Alice’s request to Bob. If Bob responds in the affirmative, the response backtracks the path taken by the request to reach Alice. Note that the media session is established directly between Alice and Bob, and does not go through the intermediary proxies.

2.2 RTP and SRTP

In SIP, the media is transported end-to-end using RTP, which exchanges packets in cleartext. A profile called Secure RTP (or SRTP [4]) was subsequently developed to provide confidentiality, message authentication, and replay protection to the cleartext RTP traffic. Conceptually, SRTP can be viewed as a “bump in the stack” implementation that resides between the RTP layer and the transport layer. SRTP intercepts RTP packets and then forwards an equivalent SRTP packet on the sending side, and intercepts SRTP packets and passes an equivalent RTP packet up the stack on the receiving side [4].

To achieve the goals of confidentiality, message authentication, and replay protection, SRTP defines extensions to the RTP packet format to encrypt the RTP payload. Each SRTP stream requires the sender and receiver to maintain cryptographic state information (the “cryptographic context”). The cryptographic context provides all the necessary parameters such as the chosen cipher, its mode of operation, and the block size; the master key; session keys; etc. SRTP uses two types of keys: session key and a master key. The session key is used directly in a cryptographic transform (i.e., payload encryption or message authentication) and the master key is a random bit string provided by the keying protocol from which session keys are derived in a cryptographically secure manner. The master key, salt, and other parameters in the cryptographic context are provided by keying mechanisms — such as SDES or DTLS-SRTP — external to SRTP. SRTP is increasingly being used in SIP; however, its wide-spread adoption has been slow (see Table 1.)

The cryptographic context itself is selected by a 32-bit numeric field carried in the fixed RTP header called Synchronization source (SSRC), which is used to identify the source of a RTP stream. Some keying protocols provide this to SRTP, while in others the SSRC is obtained dynamically when SRTP packet arrives at a receiver (the SSRC field is part of the fixed RTP header that is used without any change in SRTP; the only difference being that in SRTP the integrity of the RTP header is protected by a message authentication code.) Since SSRC is a random 32-bit number, the chance of independent RTP streams generating the same SSRC, while small, does exist. However, the two keying protocols handle such collisions appropriately.

While DTLS-SRTP is able to agree on the master key, salt and other parameters independently at the peers, some amount of information to tie the media stream to the signaling channel to prevent a third party from inserting false media packet can be provided by the signaling layer. To accomplish this, DTLS-SRTP can transport the fingerprints of the public certificates exchanged between the peers as an a=fingerprint attribute in SDP. As we will observe in Section 4, SDES transports the entire cryptographic parameters, including the master key and salt in an a=crypto SDP attribute.

3. IDENTIFYING A FEATURE SET

We now establish a core feature set that we consider immutable. That is, when we analyze the key exchange protocols, we will analyze them with a view towards how they support (or do not) this core feature set in a transparent manner (i.e., the feature behavior should not be modified to conform to the machination of the specific media keying protocol.) This core feature set includes features that are intrinsic to how SIP works as a protocol as well as features that use SIP as a service enabler. Some of the features in our set overlap with those outlined in Wing et al. [23]; however, we go further by including in our set those features that are deemed out of scope (e.g., shared-key conferencing) or not discussed at all (e.g., legal interception) in Wing et al. [23].

We consider eight features important enough to be supported by a key exchange protocol. Of these eight, six are described in Wing et al. [23]. These are: forking, the Heterogeneous Error Response Forking (HERFP) problem, minimizing media clipping, re-targeting, placing calls from the Internet to the public-switched telephone network (PSTN), and shared-key conferencing. Shared-key conferencing, while described in Wing et al. [23] is deemed out of scope in their analysis; we include it in our analysis. There are an additional two features that are not mentioned in Wing et al. [23]; these are legal intercept and session recording. We define them below.

3.1 Security Model

Before proceeding with the feature set, it is important to understand the guarantees and limitations of the security services provided by the keying techniques. First, we stress that we analyze security against very strong adversary, so-called Man-in-the-Middle (MtTM) who fully controls the communication channel between the parties. Such
3.3 Feature: Session Recording

Session recording is a critical operational requirement in many businesses, especially where voice is used as a medium for commerce and customer support [22]. SIP does not — at the protocol level — provide any explicit support for session recording. In fact, if Alice is talking to Bob, either can decide to record the session on their local endpoints, assuming that the local endpoint is capable of recording and storing media (in the most general case, recording is simply duplicating arriving and departing media packets and storing them in a persistent store while maintaining the temporal ordering between the packets.)

In order to comply with the legal procedures and regulatory environments pertinent to business practices and country codes, traditional switched networks evolved to support legal interception of the media traffic by law enforcement or business or enterprise for other reasons (e.g., recording calls at a call center for training or at a financial brokerage firm for non-repudiation.) In an end-to-end key exchange model, this operational requirement becomes harder to enforce because the service provider will not have access to the master key.

Table 1: Support for SRTP in SIP

<table>
<thead>
<tr>
<th>SIPit number (date)</th>
<th>Total unique implementations</th>
<th>Number supporting SRTP</th>
<th>Number using SDES</th>
<th>Number using DTLS-SRTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 (April 2006)</td>
<td>73</td>
<td>10</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>19 (October 2006)</td>
<td>90</td>
<td>12</td>
<td>predominant &quot;</td>
<td>0</td>
</tr>
<tr>
<td>20 (April 2007)</td>
<td>90</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>21 (November 2007)</td>
<td>70</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22 (April 2008)</td>
<td>80</td>
<td>32</td>
<td>predominant &quot;</td>
<td>0</td>
</tr>
<tr>
<td>23 (October 2008)</td>
<td>50</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>24 (May 2009)</td>
<td>43</td>
<td>16</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>25 (September 2009)</td>
<td>42</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>26 (May 2010)</td>
<td>42</td>
<td>23</td>
<td>23</td>
<td>0</td>
</tr>
</tbody>
</table>

"Exact number unknown, SIPit 19 archives state "Keying was predominantly sdes."

Data for this table gathered from SIPit official website at https://www.sipit.net/SIPitSummaries. A 0 in column 4 or 5 signifies no support for that particular keying protocol. It is not the case that the number of implementations supporting SDES and DTLS-SRTP add up to the number supporting SRTP; in some cases, implementations were using unspecified means to key the SRTP stream.

Second, flooding attacks are far easier to mount against the SIP protocol itself than they are against some of the keying techniques [9]. DTLS-SRTP in particular only performs a pair-wise key exchange with the peer that is interested in establishing a session (i.e., responds with a 200 OK response message.) Thus, the only way an attacker can mount a flooding attack at the keying layer is by causing the initial request to fork to many endpoints, each of which returns a 200 OK response to the sender. This will cause the sender to enter a pair-wise key exchange session with multiple endpoints simultaneously. Note that an attacker that simply causes a swarm of manufactured 200 OK responses to be sent to an arbitrary victim does limited harm to the victim because such a response will not match any pending SIP transaction in the victim’s transaction state table, causing the victim to simply throw away the response at the cost of a search across the transaction table. Thus we limit our discussion on flooding attacks as well, unless a certain feature requires specific discussion for such an attack.

3.2 Feature: Legal Interception

In order to comply with the legal procedures and regulatory environments pertinent to business practices and country codes, traditional switched networks evolved to support legal interception of the media traffic by law enforcement or by business or enterprise for other reasons (e.g., recording calls at a call center for training or at a financial brokerage firm for non-repudiation.) In an end-to-end key exchange model, this operational requirement becomes harder to enforce because the service provider will not have access to the master key.

4. SECURITY DESCRIPTIONS

4.1 Security Descriptions Overview

Conceptually, SDES is the simpler of the two key management protocols. Simply put, it arranges for the SRTP master key, salt, and other parameters to be transported in...
the SIP signaling messages (thus pedantically, it is not a key exchange protocol as much as a key transport protocol.)

SDES defines a new SDP attribute called “crypto” that is used to signal and negotiate cryptographic parameters for SRTP media streams. This attribute transports the encryption and authentication algorithms, master key and salt of the sender (i.e., the receiver should use the said master key and salt to derive session keys for decryption), and the lifetime of the master key (i.e., maximum number of SRTP and SRTPC packets that use this master key.)

In its simplest form, the UAC inserts this parameter in the SDP of the INVITE request and sends it to the UAS; the UAS inserts this parameter in the 200 OK response and transmits it to the UAC. Consequently, SDES provides distinct keys for each media stream in each direction. The example below shows the “crypto” attribute in an INVITE from Bob to Alice (only pertinent SIP headers shown):

```
INVITE sip:bob@example.com SIP/2.0
To: Robert <sip:bob@example.com>
From: Alice <sip:alice@example.org>;tag=0ij8z
Content-type: application/sdp

v=0
o=alice 2890844526 2890844526 IN IP4 a.example.org
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
      inline:NzB4d1BINUAvLEw6UzF3WSJ+PsdfCgdUJShpX1Zj|2^20
```

The “crypto” attribute above identifies the encryption and authentication algorithm (AES_CM_128_HMAC_SHA1_80) and specifies the master key, salt, and the lifetime of the master key (\(2^{20}\)). The master key and salt are concatenated and base 64 encoded (\(NzB4d1BINUAvLEw6UzF3WSJ+PsdfCgdUJShpX1Zj\)). The sender of the “crypto” attribute uses the master key to derive the session key for encryption and the receiver uses it to derive the session key for decryption.

Evidently, if the SIP request or response containing the “crypto” attribute is transmitted in the clear, a malicious eavesdropper can gain access to the master key. Thus, the cryptographic keys and other parameters should be secured on a hop-by-hop link using TLS. While this prevents unauthorized eavesdroppers from gathering the cryptographic keys, it does not afford complete privacy or confidentiality to the media session because the intermediaries at the end of the hop-by-hop TLS link will have access to the cleartext cryptographic keys.

MiTM attack remains a problem for SDES — if an adversary is able to inject itself as a next hop in the intermediary chain, it will have complete access to the cryptographic parameters. From this point of view, SDES may be considered the least secure of the keying protocols we consider. Note that the use of TLS-secured channels across the intermediary chain does not guarantee secure and private delivery of session keying material. This is because, as of this writing, guidelines on SIP certificate issuance are in the process of being standardized [10] and until a certificate can be issued specifically for a SIP service, any other certificate (e.g., one issued for the use of web services) may suffice. Thus, an adversary may be able to obtain a legitimate certificate from a certificate authority and then insert itself in the intermediary chain by techniques such as DNS cache poisoning. We discuss additional subtle vulnerabilities of SDES in Section 6.

### 4.2 Suitability for Feature Set

We now discuss how SDES supports the feature set we outlined in Section 3.

**Forking:** In SDES key leakage occurs as a result of forking; the master key from the initiator of the request will be replicated to all of the forked branches. One way to deal with this is to re-key the media stream after the initial session has been successfully established with one forked branch, thereby making obsolete the old key available at the remaining branches.

**HERFP remains a problem for SDES because a higher-class response that intends to negotiate the “crypto” parameters gets masked by a lower class response.**

**Media Clipping:** Media clipping also remains a problem with SDES. Each party selects their own keys for the encryption of the traffic they generate and send these keys to the other party. Consider the case where Bob establishes a session with Alice, and in that session description, he provides his cryptographic keys. Alice accepts the session and provides her cryptographic keys for decryption in the 200 OK and starts speaking, thus causing SRTP packets to go directly from her user agent to Bob’s user agent. Due to the hop-by-hop nature of her 200 OK signaling response, the SRTP packets, which take a direct route, may get to Bob’s user agent first. However, Bob does not have Alice’s cryptographic key to decrypt the packet, causing playout delay or clipping to occur.

**Re-targeting:** Re-targeting in SDES suffers from the same key leakage problem of forking. When an intermediary proxy re-targets a request, it cannot, obviously, change the cryptographic keys. Furthermore, the initiator of the request will not know that re-targeting has occurred until he or she establishes a session and exchanges some media packets with the recipient (that is, only when Bob talks to Alice’s delegate, Carol, does he know that he is not talking to Alice.)

**Conferencing:** SDES is not suitable for general conferencing since the definition of the “crypto” attribute is limited to a two-party unicast media stream where each source has a unique cryptographic key.

**Calls to Other Networks:** There is nothing intrinsically prohibitive about supporting calls to other networks in SDES. However, SDES can only secure communications within the portion of the network that supports it. That is, if SDES is negotiated by a UAC and a PSTN gateway, the media is protected using SRTP between the UAC and the PSTN gateway. When sessions continue to the PSTN from the gateway, SDES will be unable to secure the portion of the session that continues to the PSTN (or any other network.)

**Legal Interception:** Insofar as legal interception can be supported by provisioning known cryptographic keys in endpoints, SDES will support it. Unlike DTLS-SRTP that negotiates the keys in the media layer, an endpoint that uses SDES can be provisioned with a key known to the operator of the service.

**Session Recording:** Because SDES transports the cryp-
tographic keys in signaling, it is conceivable to route the signaling messages through a recording server such that it has access to the SRTP master key of each endpoint in a session.

However, there is a subtelty that comes into play here. Because the keys are delivered to the recording server in the initial request to establish a session, the recording server can act as a MiTM and inject or modify any encrypted media packets (note that while a SIP proxy also has access to the keys, the difference is that proxies are trusted in SIP whereas a recording server may not be.) A better solution would be to provide the keys to the recording server at the end of the session (through a SIP BYE request), but the SDES specification [1] does not contain any such provisions.

5. DTLS-SRTP

We start with reviewing DTLS-SRTP and then in Section 5.3 discuss its suitability for supporting the basic features described in Section 3.

As we build the presentation from the top down, note that DTLS-SRTP is a DTLS-based extension of SRTP, designed to combine the performance and security flexibility benefits of SRTP with the key and association management of DTLS. DTLS-SRTP can be equivalently viewed as a key management method for SRTP, or as a new RTP-specific data format for DTLS. We now briefly discuss DTLS, to give the necessary background for the discussion of the main aspects of DTLS-SRTP.

5.1 DTLS Overview

DTLS — Datagram TLS [15] — is an adaptation of the established and well-understood TLS to the datagram transport. The design goal of the authors of DTLS was only minimal deviation from TLS, for the simplicity of analysis (in relation to the complex TLS), and minimization of the risks of introducing errors or vulnerabilities. For the purposes of this survey paper, the differences introduced by DTLS can be largely ignored, and a reader familiar with TLS may assume that DTLS is a faithful implementation of TLS executed over datagram transport. For completeness, we give a brief overview of DTLS, and make several comments on its inherited and introduced vulnerabilities.

**DTLS message exchange**

In this section, we omit some cryptographic details, such as agreement on suites, etc. For concreteness, we show the case with mutual authentication using RSA. (If the Client is not authenticated, “request cert”, ClientCertificate and ClientCertificateVerify messages are not sent. Then the Client avoids the expense of the computation, and the Server does not perform corresponding verifications.) Further (not included in the diagram), an optional cookie is exchanged prior to the core execution to mitigate DoS attacks. That is, server only proceeds to the crypto-intensive part of the handshake, if the client is able to replay the cookie sent to the claimed IP address.

**DTLS core description**

The description below depicts the cryptographic core of the DTLS exchange; it is not a complete description of the DTLS protocol itself.

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.random →</td>
<td>(1)</td>
</tr>
<tr>
<td>← S.random</td>
<td>(2)</td>
</tr>
<tr>
<td>← pkS, certS, “req. cert”</td>
<td>(3)</td>
</tr>
<tr>
<td>Verify certS</td>
<td></td>
</tr>
<tr>
<td>pkC, certC →</td>
<td>(4)</td>
</tr>
<tr>
<td>Choose rand. r</td>
<td></td>
</tr>
<tr>
<td>EncpkS(r) →</td>
<td>(5)</td>
</tr>
<tr>
<td>sigpkC(H(prev.msgs)) →</td>
<td>(6)</td>
</tr>
<tr>
<td>Verify certC</td>
<td></td>
</tr>
<tr>
<td>Verify sigpkC</td>
<td></td>
</tr>
<tr>
<td>Decrypt</td>
<td></td>
</tr>
<tr>
<td>EncpkS(r)</td>
<td></td>
</tr>
</tbody>
</table>

where messages (1)–(6) are:

1. ClientHello
2. ServerHello
3. ServerCertificate
4. ClientCertificate
5. ClientKeyExchange
6. ClientCertificateVerify

The session key is set to PRF(r, “master secret”, ClientHello.random+ServerHello.random). Note, prev.msgs includes all previously exchanged messages, and, in particular, EncpkS(r), C.random and S.random. SignpkC is the public key signature.

Here pkC, pkS, certC, certS are public keys and certificates of the client and server respectively. Certificates include the public keys, but we wrote them out separately to be explicit. Note that these parameters are transmitted in the clear and are publicly known.

**DTLS security**

First, we would like to point out that DTLS-SRTP and its use in SIP are not vulnerable to variants of Man-in-the-Middle (MiTM) attack on TLS derivatives, described in [3], even though, by design, no improvements were introduced in the DTLS derivation. The reason is that the attack is applicable only in a few settings, namely, where TLS is used to establish a tunnel over which a second-factor (e.g. password) authentication is performed.

Given that DTLS-SRTP is run in mutual authentication mode, it provides good protection against active attacks. In addition to TLS DoS attacks, DTLS suffers from the standard resource consumption attack, and an amplification attack. In our opinion, both of these are of mild severity, and are further mitigated by the cookie exchange described above.

DTLS-SRTP depends on a PKI to prevent MiTM attacks. Additionally, to remove/reduce reliance on PKI, DTLS-SRTP endpoints exchange the fingerprint of the certificates in SIP signaling channel; when key exchange is performed in the media channel, each side compares the other sides fingerprint to the received key. A MiTM attack would effectively need to control both the media and signaling to mount a successful attack.

---

**Table 5.1**: DTLS-SRTP Process

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ClientHello</td>
</tr>
<tr>
<td>2.</td>
<td>ServerHello</td>
</tr>
<tr>
<td>3.</td>
<td>ServerCertificate</td>
</tr>
<tr>
<td>4.</td>
<td>ClientCertificate</td>
</tr>
<tr>
<td>5.</td>
<td>ClientKeyExchange</td>
</tr>
<tr>
<td>6.</td>
<td>ClientCertificateVerify</td>
</tr>
</tbody>
</table>

**Diagram 5.1**: DTLS-SRTP Process Flowchart
5.2 DTLS-SRTP Overview

While DTLS provides the key to the communicating parties, DTLS-SRTP specifies its usage in the following data exchanges.

DTLS-SRTP is defined for point-to-point media sessions, in which there are exactly two participants. Each DTLS-SRTP session contains a single DTLS association, and either two SRTP contexts (if media traffic is flowing in both directions on the same host/port quartet) or one SRTP context (if media traffic is only flowing in one direction). All SRTP traffic flowing over that pair in a given direction uses a single SRTP context. A single DTLS-SRTP session only protects data carried over a single UDP source and destination port pair in a single direction.

The general pattern of DTLS-SRTP is as follows. For each RTP or RTCP flow, the peers do a DTLS handshake on the same source and destination port pair to establish a DTLS association. Which side is the DTLS client and which side is the DTLS server is established via an out of band mechanism (SIP). The keying material from that handshake is fed into the SRTP stack. Once that association is established, RTP packets are protected (becoming SRTP) using that keying material.

Between a single pair of participants, there may be multiple media sessions. There must be a separate DTLS-SRTP session for each distinct pair of source and destination ports used by a media session. However, for efficiency, it is recommended that such sessions share a single DTLS session and hence amortize the initial public key handshake. This is done by deriving separate DTLS-SRTP master keys for each DTLS-SRTP session from the same DTLS output.

Credentials and Authentication

The security of entire data exchange run by DTLS-SRTP is dependent on the integrity of the public key certificates possessed by the communicating parties. Ideally, they will be maintained by a PKI; however this solution has potential high costs associated with it.

An alternative natural approach is to delegate some of the responsibility to the SIP layer. For example, as described in [14], parties may exchange hashes of their public keys in the SIP layer. Then, if the SIP layer is secured, this provides sufficient guarantees; if it is not, this serves merely as an additional hurdle for the attacker, and the combined protocol is still vulnerable to attacks.

More specifically, when Alice wishes to set up a secure media session with Bob, she sends an offer in a SIP message to Bob. This offer includes, as part of the SDP payload, the fingerprint (i.e. secure collision-resistant hash) of Alice’s certificate. Alice should utilize existing SIP security mechanism, and send this message to her proxy over an integrity-protected channel. If all the channels on the way to Bob are integrity-protected, a polynomial time adversary will not be able to compromise the security of DTLS-SRTP.

5.3 Suitability for Feature Set

In this section, we go over features discussed in Section 3 and analyze their support by DTLS-SRTP.

Forking: Key exchange and session establishment occurs in DTLS-SRTP in the media plane. Therefore, each responder would establish independent key with the initiator, and key leakage will not occur. Further, as also noted in [7] (Appendix A.24), since key exchange is executed in the media path, error messages are also communicated along this path, and proxies will not need to take action based on error messages. Thus, Heterogeneous Error Response Forking Problem (HERFP) is not applicable here either. In summary, threats associated with forking, as described in Section 3 are not applicable in DTLS-SRTP.

Media Clipping: Again, since keying occurs in the media plane, user agent applications are in full control over how to send the data (encrypted or not), depending on whether DTLS has completed and keys were derived. Therefore, the problem of early media clipping, as described in Section 3 is easily avoidable by client applications.

DTLS-SRTP signals its intent such that both peers must support the extension before SRTP media flows between them. In this respect, it does not result in any leak of privacy by first sending plaintext RTP.

Re-targeting: First, we observe that the keys will not be leaked to unintended recipients since key exchange is executed end-to-end in the media plane. Further, authenticated DTLS-SRTP will always detect an exception in case of re-targeting, since the credentials won’t match. Because DTLS-SRTP relies on certificates, the initiator will have received the certificate of the responder and will be able to identify the person to whom the call has been re-targeted.

As an aside, to our knowledge no proposed protocol supports cryptographic delegation of authorization from Bob to Carol. Such an authorization, for example, may be a simple specially formatted message signed by Bob, associating Carol’s public key, delegation period, and possibly other relevant information. When Carol answers the call, this message can be attached to her PKI chain to convince Alice that Carol is an authorized representative.

Conferencing: DTLS-SRTP does not support establishment of a single key shared between more than two endpoints. However, participants can still establish DTLS-SRTP sessions individually with a conference bridge.

When Alice participates in a conference, DTLS-SRTP allows her to establish secure media to the conference bridge or entity acting as the bridge in the case of three-way calling when a participant bridges someone into the call. Alice has no control over whether or not media from her is encrypted as it is sent from the bridge to other participants. Alice also has no control over who the other participants are and therefore to whom the media is sent (aside from being able to choose not to participate herself).

Calls to Other Networks: As mentioned in the discussions on forking and re-targeting, one endpoint may not be within a VoIP network and the SRTP terminates at a gateway to another network, such as a switched cellular network or PSTN. The same gateway may not be used for every call between the same two endpoints. For such calls, DTLS-SRTP only provides establishment of SRTP keying material between the participant on the VoIP network and an undetermined endpoint.

Legal Interception: DTLS-SRTP exchanges keys end-to-end in the media stream. Unlike SDES, it does not transport the keys in signaling thus making legal interception (or informed recording of conversation as in the case of call centers or financial transactions) harder to support.

Session Recording: DTLS-SRTP does not provide a general recording solution since it does not specify the exact
means by which the key can be shared with a recording server.

6. OUR MEDIA KEYING SCHEME

As discussed in the introduction, while DTLS-SRTP and ZRTP provide strong security in establishing session keys, they are still not widely deployed due to the complexity of implementation and significant computation and communication costs. For simplicity and efficiency reasons, media key is often chosen and transmitted by Alice to receiver Bob via trusted SIP framework intermediaries. This method provides adequate security for low-value media streams.

In this section, we propose a simple and efficient way to significantly increase the security of SIP key transmission, with minimal additional costs. We do not resort to more expensive public key cryptography. Our proposal involves sending an extra key, and evaluating one Pseudo-Random Permutation (PRP), such as AES. We believe that our protocol presents a desirable trade-off between security, costs, and deployment complexity. It can be built directly from SIP key transmission by adding two simple steps.

We start the presentation with discussion of some of the weaknesses of SIP key transmission.

Key Transmission weaknesses.

We assume that both Alice and Bob are properly authenticated to the SIP network. We mention obvious vulnerabilities resulting from corrupt SIP server node(s) – in this scenario all security is lost since adversary sees the session key.

However, there are subtle attacks by a relatively weak adversary who does not have access to privileged SIP nodes. These attacks are due to forking, which may result in the SIP network sending Alice’s key to more than one of Bob’s devices.

The first attack may occur in the scenario where adversary is in possession of one of the Bob’s devices B1. Then, adversary is informed of the session key by the SIP network and can interact with Bob pretending to be Alice. (We note that this attack is prevented in our protocol.) Note that this is a different and stronger attack than an (unavoidable) possibility of adversary in possession of B1 pretending to be Bob to unsuspecting Alice.

In the second attack scenario, adversary does not have access to Bob’s devices, but controls a portion of the media-plane network. He is able to redirect the messages between honest Alice’s device A1 and Bob’s two devices B1 and B2, all of which use the same key k. Even though adversary does not know the shared key all three devices share, adversary may be able to route the messages to create unintended transactions, even if channels are protected with k. For example, Bob’s devices may talk to each other thinking they are talking to Alice. Or, outside of VoIP scope, both Bob’s devices would initiate a transaction (e.g. a money transfer), and result in duplicate transaction execution1.

Finally, even if SIP servers are trusted — and it is reasonable to trust the intermediaries not to abuse the knowledge of all session keys — hiding the keys from them, among other advantages, reduces the servers’ liability, consequences of compromise, and makes system recovery easier.

DTLS-SRTP is a secure Key Exchange (KE); using SIP with DTLS-SRTP avoids all possibilities of attacks2, including the above weaknesses. However, this solution involves the use of PKI. While conceptually relatively simple, PKI systems are expensive to deploy and manage, and it is best to avoid them. As suggested in DTLS-SRTP, a natural way to eliminate the logistical complexity of PKI is for the participants to transfer hashes of their certificates in the secured SIP layer. This way, the adversary on the insecure media plane channel would not be able to substitute the certificates, and thus the certificates can be trusted. However, this approach fails to provide security against SIP-layer adversaries, who in fact can substitute the certificates and enable man-in-the-middle attacks. We believe this is a reasonable compromise between security and deployment and running costs. The above non-PKI version of DTLS-SRTP is (ever so slightly) more secure but also significantly more costly than our key agreement protocol described next.

6.1 Description of our solution

Our proposed solution achieves most of the security goals achieved by the above non-PKI version of DTLS-SRTP, but without the computation and communication complexity associated with its public-key operations.

We present a generic version of the protocol, based on Pseudorandom Permutation Generators (PRPG). Further, we do not fix the domains for the randomly drawn keys, messages, and values. We only require that they are “large enough”, according to the current suggested key lengths. Today, we envision using the AES encryption as the PRPG, in which case the domains of k and r may be k, r ∈ {0, 1}128.

PROTOCOL 1. (Secure SIP Key Transmission)

Setup: Initiator Alice wishes to securely connect to responder Bob. Both Alice’s and Bob’s devices are authenticated to the corresponding SIP servers.

1. Alice chooses a random key k and transmits string (Alice, Bob, k) to the responder Bob via the SIP framework, as it is done in the SIP key transmission method. We stress that the transmitted key will not be the session key that is used for communication.

2. Upon the receipt of the key, Bob chooses a random nonce r and sends back string (Alice, Bob, r) to the initiator Alice, together with the media stream.

3. The session key, which can be immediately used to encrypt the media, is the PRPG F evaluated with the seed k on the data r. Namely, the session key is sk = F_k(r)

This protocol flow is illustrated in Figure 1.

Note that, in particular, this protocol prevents more than one instance of Bob from obtaining the session key due to

1We note that while the session encryption may be such that such message manipulation is difficult (e.g. using special counters), key security should not be delegated to the session, but achieved in the key exchange/transmission phase. This would allow for better modularity and more easily understandable protocols. Further, standard proofs of security are done in this modular world.

2The crypto core of TLS was formally proven secure [16]. However, the complete protocol suite has not been fully analyzed, and there are potential weaknesses in certain parts of the protocol, leading to possible attacks, such as the recent TLS renegotiation attack.
As noted above, here we only present the main points of why Theorem 1 holds. Indeed, a polynomial-time adversary who observed (or even modified in transit) \( r \), but had not obtained \( k \) (this is the media-plane-only adversary considered in the theorem), will not be able to distinguish \( sk \) from a random string of the same length. This follows immediately from the security properties of PRPG \( F \), namely, from the fact that the output of \( F \) evaluated with a random and unknown key on any adversarially chosen message, is indistinguishable from a random string. Further, adversary will not be able to mismatch honest players (i.e. forge an unintended communication channel), since honest Alice instance \( A_i \) selects random \( k_i \), and each of the honest Bob’s devices \( B_j \) (who receives some \( k_i \)) independently chooses \( r_j \); both \( k_i \) and \( r_j \) are unique with overwhelming probability. Therefore, even if the same \( k_i \) is delivered to several Bob instances (e.g. due to forking) and arbitrary \( r \) values are delivered to Alice instances, the keys output by each player instance will be either all independently random, or there may be (at most) two equal keys, which would correspond to a successful completion of the KE protocol. (We note that a media-plane MiTM may “connect”, i.e. cause output of the same session key, of a different Bob’s device that Alice expects, e.g., based on the IP address. We note that this can be avoided by additional signaling in the SIP layer, but we do not consider this a KE vulnerability. All we guarantee here is that if Alice establishes a channel, it is with a single Bob’s device.)

Next, we show that our protocol is secure against a SIP signaling-plane-only adversary. We note that this corresponds to the setting where a SIP server may be corrupted, but the attacker is unable to consistently monitor the general Internet traffic of the parties.

For formal proof of security in one of the attack scenarios covered by the next theorem, we would need to rely on a slightly stronger than PRPG notion of security, ideal cipher. (See Footnote 3 for high-level description of its security properties.) We envision using AES as the instantiation of ideal cipher, as its design aims to satisfy the required properties.

**Theorem 2.** Let \( F \) be an ideal cipher. Assume that adversary is unable to observe or interfere with (only) the protocol message sent in the media plane. Then our protocol is a secure key exchange protocol.

We give intuition for the proof of Theorem 2. Indeed, a polynomial-time adversary who observed \( k \), but had not obtained \( r \), cannot distinguish \( sk \) from random. This is because \( F \) is a (known to the adversary) permutation, which, applied to a random input, produces random output. We note that a SIP signaling plane attacker (i.e. a rogue server) modifies \( k \) in transit does not gain any advantage, if a “good” pseudorandom function (e.g. AES) is used\(^3\).

\(^3\)Strictly speaking, PRPG does not guarantee any security properties if executed on related keys. Resilience to related key attacks is modeled by assuming stronger properties on the underlying function, in our case, AES. This assumption is referred to as the ideal cipher assumption. While it is sometimes considered too strong in theoretical cryptogra-
Further, adversary cannot forge an intended connection among the players, since all Bob’s devices choose an independently random \( r \), which results in all of them computing independent session keys. Alice receives \( r \) generated by one of the Bob’s devices, and outputs either the same corresponding session key, or an independently random key, in case \( k \) was modified (here we use the ideal cipher assumption). Note that SIP-layer adversary may misrepresent the identity of Alice to Bob, hoping to cause Bob to believe he is talking to Carol, while he is in fact talking to Alice. We address this by including the names of both players, in order (initiator, responder) in both protocol messages. This way, Alice will not accept Bob’s response which includes Carol’s name.

We stress that we use the ideal cipher assumption only for proving claims related to active adversary in the SIP layer. All other claims are proven only assuming \( F \) is a PRPG.

Finally, we caution the reader that colluding SIP servers and the media-stream attackers succeed easily. It is sufficient for the SIP server to leak the key \( k \) to the media-stream MiTM to break into the conversation. However, at the same time, attacking a non-PKI DTLS-SRTP version described in this section requires only slightly stronger resources. There, a SIP server simply substitutes the transmitted hash to enable the media-stream MiTM to perform the attack.

In conclusion, we proposed a simple, secure, and very efficient amendment of the protocol for key transmission. In particular, our proposed amendment reduces the trust assumptions on the SIP servers, and prevents instances of the responder sharing the session key due to forking.

6.3 Suitability for Feature Set

We now discuss the applicability of our approach to the features outlined in Section 3.

- **Forking:** In DTLS-SRTP, keying material is exchanged completely in the the media stream. In our proposal, the key exchange is distributed between the signaling stream and the media stream: the random key \( k \) is sent in the signaling stream and the nonce \( r \) flows in the media stream. When forking occurs, \( k \) remains constant for all the forked branches, but each branch contributes a unique \( r \), thus deriving a separate session key and preventing key leakage to parties not part of the session. Similarly, HERFP does not pose a problem since there is no key negotiation done in signaling; the \( k \) carried in signaling is not subject to negotiation. If an endpoint does not support the interpretation of \( k \), it will simply ignore it (following the accepted practice of handling unknown headers and attributes in Internet protocols.)

- **Media Clipping:** Media clipping does not pose a problem in our approach. Key derivation is complete when \( A \) (see Figure 1) receives nonce \( r \). Since \( B \) will send the first media packets, it can encrypt them using the session key (thus, no plaintext RTP packets will be sent.) Furthermore, since the nonce \( r \) is different for each endpoint the request forked to, HERFP does not pose a problem for our approach.

- **Re-targeting:** We do not formally address re-targeting. However, we briefly sketch the possibilities to handle its simple forms.

  When Alice’s UA has a good user interface (e.g. a computer, or a phone with a display), SIP layer may inform Alice that the call was sent to Carol rather than Bob, and that would imply Carol is authorized to receive Bob’s calls. Further, Alice’s UA may store the pre-shared key she had shared with Bob (if this is a repeat call), and determine by itself that re-targeting has occurred. In either of these cases, Alice is notified of an exception, and may take corresponding action. Finally, as with forking, we note that keys will not be leaked to unintended recipients.

7. RELATED WORK

We have presented and proved secure a novel key exchange method that involves only one symmetric key exchange operation by the sender and receiver. We provide security guarantees which are much stronger than that of SDES, and are nearly as strong as that of DTLS-SRTP. At the same time, our computational costs are comparable to that of SDES, and are much less expensive than DTLS-SRTP and ZRTP, which use public-key encryption. Table 2 provides a feature evaluation summary of our key exchange method with the analysis we performed also for DTLS-SRTP and SDES. We note that our method compares well against both SDES and DTLS-SRTP.

8. CONCLUSIONS

We have presented and proved secure a novel key exchange method that involves only one symmetric key exchange operation by the sender and receiver. We provide security guarantees which are much stronger than that of SDES, and are nearly as strong as that of DTLS-SRTP. At the same time, our computational costs are comparable to that of SDES, and are much less expensive than DTLS-SRTP and ZRTP, which use public-key encryption. Table 2 provides a feature evaluation summary of our key exchange method with the analysis we performed also for DTLS-SRTP and SDES. We note that our method compares well against both SDES and DTLS-SRTP.

9. REFERENCES


Table 2: Feature Evaluation Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>SDES</th>
<th>DTLS-SRTP</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>MITM</td>
<td>Attacks possible</td>
<td>Mitigated through certificate fingerprints</td>
<td>Adversary controls media plane: secure; adversary controls signaling plane: secure; adversary controls both planes: attacks possible</td>
</tr>
<tr>
<td>Forking</td>
<td>Key leakage occurs</td>
<td>No key leakage</td>
<td>No key leakage</td>
</tr>
<tr>
<td>HERFP</td>
<td>Remains problematic</td>
<td>Not a problem</td>
<td>Not a problem</td>
</tr>
<tr>
<td>Media clipping</td>
<td>Remains problematic</td>
<td>Not a problem</td>
<td>Not a problem</td>
</tr>
<tr>
<td>Retargeting</td>
<td>Remains problematic</td>
<td>Detects retargetting</td>
<td>Detects retargetting</td>
</tr>
<tr>
<td>Conferencing</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>PSTN calling</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Legal Intercept</td>
<td>Supported (keys can be provisioned in endpoints)</td>
<td>Not supported (key exchange is end-to-end)</td>
<td>Not supported (key exchange is end-to-end)</td>
</tr>
<tr>
<td>Session Recording</td>
<td>Supported (keys available to the recording server)</td>
<td>Not supported (no mechanism in protocol to share keys)</td>
<td>Not supported (no mechanism in protocol to share keys)</td>
</tr>
</tbody>
</table>


