A System for Secure IP Telephone Conferences

Axel Treßel
T-Systems Enterprise Services GmbH
Security Solutions
Postfach 91 00
55541 Bad Kreuznach, Germany
axel.tressel@t-systems.com

Jörg Keller
FernUniversität in Hagen
LG Parallelität und VLSI
Postfach 940
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

Abstract

We present a system for secure telephone conferences (stc) over the internet. The system ensures participant authentication via x.509 certificates, such that every participant of a conference is informed about every other participant. Also, all signaling and media data are encrypted, to ensure confidentiality. The system builds upon the open source telephone server asterisk and standard IP softphones. Those software products are used unaltered. Stc client and server processes reside with softphones and server, respectively, to realize secure conferences. Experiments with our prototype show that the additional network and processor load is low, and that the system scales well for more than 10 participants.

1. Introduction

Telephone conferences more and more take the place of conventional business meetings. The reasons are manifold: for example travel budgets for meetings may be shrinking, or travel schedules of the participants cannot be synchronized to meet in one place. In classical telephone conferences, the authentication of participants is obtained by the trust in the public telephone network, i.e. if a known telephone number is dialed everyone trusts in the telephone providers that the call is routed to the correct phone. Confidentiality of the conference also relies on the providers. Everyone assumes that the telephone providers are trustworthy themselves, and that their network is inaccessible to eavesdroppers. This assumption does not necessarily hold. Foreign intelligence services may have access to their local provider's network to perform industrial espionage. Rumor has it that the president of a large German company used cell phones with encryption to securely participate from abroad in telephone conferences with the board, but one by one those cell phones did not connect anymore. One is tempted to assume that the eavesdroppers, realizing that the phone call was encrypted, turned their attack into a denial-of-service.

With the convergence of IP data networks and classical telephone service in Voice over IP (VoIP), the situation changes to the worse. While long-distance telephone calls over the internet are (almost) for free, on the other hand the internet is not as secure as we believe our telephone network to be. First, there are no unique telephone numbers in the internet. This necessitates user authentication. Second, eavesdropping is simple, as long as media data are not encrypted. A simple tool like `ethereal`[1] suffices to see and record the audio data stream. This necessitates strong encryption. While the real time protocol (RTP), used to transmit audio data in VoIP applications, has a secure companion protocol SRTP, the latter is not supported by the majority of softphones, and also does not include a key exchange. Hence, there is a need to supplement VoIP with confidentiality and strong authentication. We concentrate on IP telephone conferences as they seem a relevant application, and because their security carries over to peer-to-peer IP telephony.

We present a system for secure telephone conferencing over IP networks. We build upon the open-source telephone conference server `asterisk` [7] and available softphones like `x-lite`[2], which can remain unchanged. Our system bridges between RTP and SRTP to prevent eavesdropping, realizes secure key exchange for the latter protocol, encrypts also the signalling data, and provides user authentication. The last is used to provide every participant of the conference with a list of all authenticated participants, thus preventing silent participants. The proposed scheme is novel, as far as we know.

The remainder of this paper is organized as follows. In Section 2 we briefly review the architecture of a VoIP conference system, and the steps necessary to secure it. In Sec-

[1] www.ethereal.com
[2] www.xten.net

tion 3 we present our prototype implementation. In Section 4 we conclude and give an outlook onto further work.

2. Securing VoIP conferences

A VoIP connection consists of signalling and media data. Both are transmitted over IP networks. Whether the VoIP connection exists within an organization's internal network, or whether it also spans public networks, there are a number of threats. First and foremost, an attacker could eavesdrop the communication, and confidentiality would be destroyed. Second, an attacker could fake the identity of caller or callee. While this may be difficult in a one-to-one communication, it works well in conferences as long as the attacker is silent. This constitutes an attack on both authentication and confidentiality. Integrity is in danger, too, as packets may be altered along the way. While altering packets with speech data seems difficult and is more a kind of denial of service (receiver cannot understand anything), changing of signalling data could route a call to a different callee, with consequences for identity and confidentiality. Finally, there are pure Denial of Service attacks. The Seminar [1] gives an overview of recent threats. From this we conclude that both media and signalling data are to be encrypted to ensure confidential communication, and that users have to be authenticated.

Signalling in VoIP uses either ITU-T's H.323 [2] or IETF's SIP[3] [3] protocols. As we see the latter as more widely deployed because of its simpler structure, we will concentrate on this protocol. SIP runs on UDP and not on TCP. This has to be taken into account when securing SIP connections. According to the RFC 3261, securing can be realized by S/MIME, by SSL/TLS, which however assumes a TCP connection, or by IPSec, which is possible for UDP and TCP connections. As S/MIME still has some problems with inter operability and key management, we concentrate on the latter two protocols. Both TLS and IPSec allow authentication of the communication partners by X.509 certificates.

Certificates are digital documents that contain a user's name and other identification features, together with the user's public key or a hash thereof. Certificates are digitally signed by the certification authority (CA) where the user has registered his key. For authentication, user A can send a short message m to user B. User B encrypts this message with his private key, and sends back the encrypted message m' together with his certificate. Alternatively, user A may already possess B's certificate. User A decrypts the message m' with the public key from the certificate, and if the result is m again, knows that the user B owns the private key corresponding to the public key in the certificate. If the certificate is valid, i.e. if the digital signature of the CA is correct, and the certificate has not been revoked by the CA, then user B is authenticated.

Certificates have the advantage that many organizations already have built a Public Key Infrastructure (PKI) for their employees, including certification authority, certificate revokation, and the like. Thus, certificates are already provided for in many organizations. X.509 version 3 certificates (see e.g. [6]) are the current de facto standard for certificates, therefore we will assume them. As RFC 3261 only mentions IPSec in passing, and leaves many details open, we decided to use TLS.

Media data are transmitted using the Real Time Protocol (RTP) [4] together with the Real Time Control Protocol (RTCP). Both protocols have secure companion protocols SRTP and SRTCP [5]. Those protocols allow symmetric encryption and authentication of packets on the basis of AES and SHA-1, respectively. However, the exchange of the secret key is not part of the protocol and has to be handled separately. For the key exchange, we decided to use the Diffie-Hellman key exchange routine. While this procedure is the standard for exchanging a session key between two communication partners A and B without transmitting the key itself, it can be attacked by a man-in-the-middle attack (see e.g. [6]), where the attacker intercepts all packets between A and B, and plays B's role when communicating with A, and vice versa. A normal measure to prevent such interception is to first establish a secure channel between A and B, for example by transmitting all messages related to the key exchange over a secure TLS connection. As such a connection is already necessary for the signalling data, the additional overhead is minimal.

3. Prototype system

We have realized the security features derived in the previous section in a prototype system called stc (secure telephone conference). We decided to use the x-lite softphone, and the asterisk telephone server. The latter has the advantage that is has a script interface, the asterisk gateway interface (AGI), which allows to couple the server with other applications.

For our system, we first decided that we wanted a central server, although direct connections between all conference participants are possible to transmit media data. However, with n participants, each participant would have to maintain $n-1$ connections, leading to an increase in bandwidth, complicating the key exchange, and necessitating to mix the n speech signals. As the central server decrypts all signalling and media data, it must be trustworthy, and particularly protected.

As we want to use the telephone server and the softphone unaltered, we couple each with a piece of the stc

[3] Session Initiation Protocol

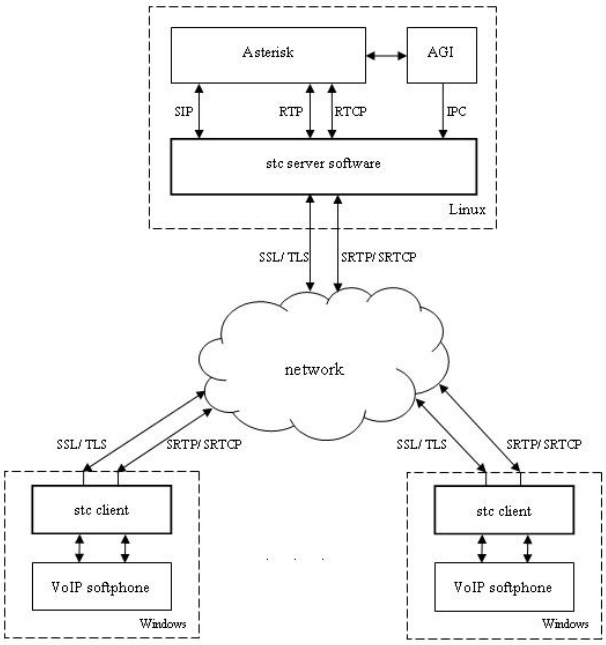

Figure 1. Prototype architecture.

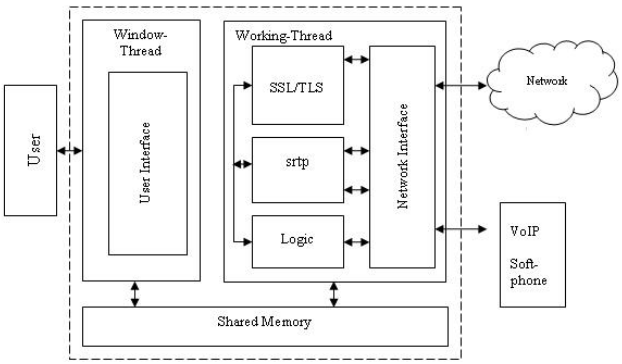

Figure 2. Structure of the stc client software.

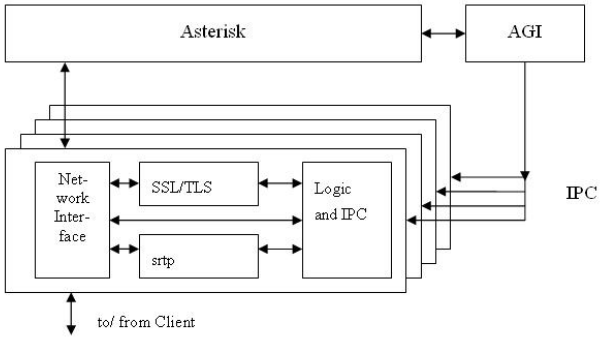

Figure 3. Structure of the stc server software.

software. Figure 1 depicts the architecture of the prototype. The central server on top, with a Linux operating system, contains the asterisk telephone server, and the stc server software. Both are coupled via the AGI. The clients at the bottom, running with the Windows operating system, each contain a softphone and the stc client software. Asterisk and the softphones communicate with their corresponding stc software counterparts via SIP, RTP, and RTCP. They do not know that any encryption is taking place. The stc server communicates with the stc clients only via TLS and SRTP/SRTCP. Thus, signalling and media data are unencrypted only within computers, and are always encrypted when travelling on the network.

Figure 2 depicts the structure of the stc client software. It consists of two threads. One is responsible for the user interface, and the other deals with the network and the softphone. It first initiates a TLS connection to the stc server, and both parties are authenticated via X.509 certificates. Then the Diffie-Hellman key exchange is performed. Now the stc client opens a UDP port towards the softphone and accepts SIP messages. It forwards those messages through the TLS connection to the server, and SIP messages arriving from the server to the softphone. If the stc client notices that the softphone initiates a VoIP connection, it opens RTP and RTCP connections towards the softphone and SRTP and SRTCP connections towards the server. Packets arrriving via RTP are encrypted and forwarded via SRTP, and vice versa. The TLS connection is also used to transmit the certificate data of all conference participants to each stc client.

Figure 3 depicts the structure of the stc server software. It consists of several processes: one for each connection, and one for the coordination of all other processes. The server software works similarly to the client software, with the addition that the stc server software is coupled to the asterisk server. The asterisk server informs the stc server when a conference is started. Then the participant data is forwarded to all clients.

The stc server software was written in C and compiled with the GNU C compiler. The stc client software was written in C++ and compiled with Microsoft Visual Studio. We used the OpenSSL library for the TLS functions and the libsrtp for the encryption of the media data.

Figure 4 displays a screen shot of the user interface of the stc client software. The green light on the right side and the status line on the bottom indicate that a secure connection to the stc server has been established. The configuration button can be used to select the X.509 certificates, the stc server's IP address, and similar parameters. The window in the center displays all conference participants according to

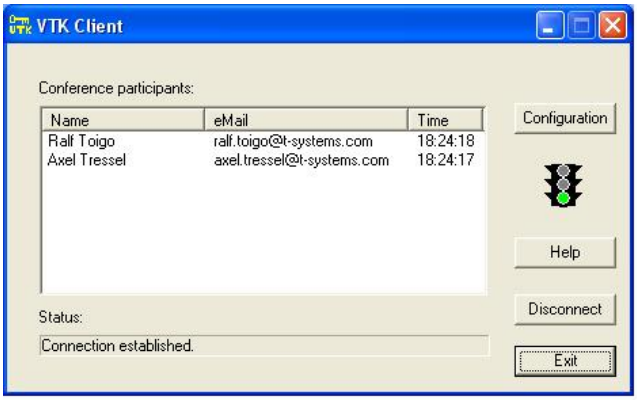

Figure 4. User interface of the stc client software.

data in their certificates. When a secure connection is established upon start of the client software, a particular conference can be chosen as usual by dialling the number of the asterisk conference room on the softphone. Thus, the stc software is completely transparent for the softphone and the asterisk server.

The software was tested in practice. When we tried to eavesdrop on the encrypted connections, we were not able to hear anything beyond noise, no matter which of the available speech codecs we used. Thus, the speech data is really encrypted. As the encryption is done with the AES algorithm, which is a symmetric encryption algorithm using 128-bit keys, chosen by NIST as the successor of the DES standard, eavesdropping would require to break into this algorithm. Thus, the communication can be considered confidential.

The stc client software only consumes a small amount of memory and processor performance (less than 1% processor power) even on a notebook. We were not able to notice any change in the audio quality when switching form unencrypted VoIP communication to the stc system.

When measuring server performance, we concentrated on load originating from established connections, as authentication and other additional tasks only happen upon initiating a connection to the server. The stc server software was run on a Toshiba Tecra S2 notebook with an Intel Pentium-M 750 centrino-processor at 1.86 GHz and 2 GB of RAM. Running asterisk and stc server with 4 clients, both the processor load and the memory consumption were less than 10%. Thus we conclude that the system could be run with more than 20 clients on such a platform.

4. Conclusions

We have presented a system for secure telephone conferences over insecure IP networks. The system extends the usual softphone — asterisk server combination by a client and a server software. The system has been tested in practice and found to work satisfactorily. The software is even able to work in environments where network address translation (NAT) is performed.

Yet, handling two software products (softphone and stc client) may be disturbing for the average user. Therefore, one may think to integrate the stc client and the softphone by using a software development kit available from several softphone producers. Another step could be to integrate the stc client into an IP telephone device. As those often contain an embedded linux system, such an extension is possible but would require access to the source code of the phone. Finally, the system could be extended to also securely connect to phones via ISDN or GSM.

References

[1] Information Systems Audit and Control Association (ISACA). Voice over IP security — after hour seminar, May 2005. http://www.isaca.ch/files/AHS_VoIP_Sec.pdf.

[2] International Telecommunications Union (ITU-T). Recommendation H.323 — packet-based multimedia communications system, July 2003. http://www.itu.int/rec/T-REC-H.323-200307-I/en.

[3] Internet Engineering Task Force. RFC 3261 — SIP: Session initiation protocol, 2002. http://www.ietf.org/rfc/rfc3261.txt.

[4] Internet Engineering Task Force. RFC 3550 — RTP: A transport protocol for real-time applications, 2003. http://www.ietf.org/rfc/rfc3550.txt.

[5] Internet Engineering Task Force. RFC 3711 — the secure real-time transport protocol (SRTP), 2004. http://www.ietf.org/rfc/rfc3711.txt.

[6] B. Schneier. *Applied Cryptography*. Wiley, 2nd edition, 1996.

[7] J. VanMeggelen, J. Smith, and L. Madsen. *Asterisk. The Future of Telephony*. O'Reilly Media, 2005.