Analysing Protocol Implementations[⋆]

Anders Moen Hagalisletto, Lars Strand,
Wolfgang Leister, and Arne-Kristian Groven

Norwegian Computing Center
{Anders.Moen,Lars.Strand,Wolfgang.Leister,Arne-Kristian.Groven}@nr.no

Abstract. Many protocols running over the Internet are neither formalised, nor formally analysed. The amount of documentation for telecommunication protocols used in real-life applications is huge, while the available analysis methods and tools require precise and clear-cut protocol clauses. A manual formalisation of the Session Initiation Protocol (SIP) used in Voice over IP (VoIP) applications is not feasible. Therefore, by combining the information retrieved from the specification documents published by the IETF, and traces of real world SIP traffic we craft a formal specification of the protocol in addition to an implementation of the protocol. In the course of our work we detected several weaknesses, both of SIP call setup and in the Asterisk implementation of the protocol. These weaknesses could be exploited and pose as a threat for authentication and non-repudiation of VoIP calls.

1 Introduction

Voice over IP (VoIP) is widely used, and is about to replace the traditional, public switched telephone networks (PSTN) for two main reasons: (1) Providers and customers experience cost savings, especially for long distance calls. Since VoIP use the Internet as carrier, the cost of setting up a phone call needs no more effort than sending an email. (2) Added functionality and flexibility. The VoIP protocols are capable of providing a number of additional services like instant messaging, presence, conferencing, events notification, video calls and other multimedia transmissions and location independence (mobility).

VoIP services cannot rely their security on the telecommunication infrastructure, dedicated lines, physically protected switches, and certified telephony equipment. VoIP services have to be secured by cryptographic techniques. Therefore, the employed protocols and their implementations must undergo a thorough formal crypto-analysis.

A common combination in VoIP is to use the Session Initiation Protocol (SIP) [14] for signalling, e.g., setting up and tearing down calls, and specific protocols for the actual media transfer. The designers of SIP focused on functionality for

⋆ This research is funded by the EUX2010SEC project in the VERDIKT framework of the Norwegian Research Council. The authors would like to thank Jørn Inge Vestgaarden, Lothar Fritsch, Einar W. Høst, Svetlana Boudko, and Truls Fretland for comments on earlier drafts of this paper.

providing specific services rather than security features [15]. However, security issues have been recognised to be an area of further investigation and improvement [1, 6, 4, 10]. Discussions about potential weaknesses and attacks on SIP have, in most cases, been kept on an informal level [16, 4, 13, 18, 17, 12].

Our goal has been to use formal modelling of SIP in order to (1) verify whether the Asterisk implementation of SIP follow the specifications; and (2) perform attacks exploiting weaknesses in the protocol definition, and its implementations. This work is based on previous work [9] where we analysed digest access authentication in the SIP registration process. The same authentication mechanism is used in the call-setup explained in this paper.

There have been other works that analyse SIP and its security configurations formally [7, 2], and the work of the AVISPA project[1]. However, they consider the authentication and registration sub-protocol in combination with the Diameter protocol, rather than the call-setup protocol as presented here.

The rest of the paper is organised as follows: In Section 2, we give a high level overview of SIP, and the tool PROSA that we have used to analyse the call setup. In Section 3, the formal specification of digest access authentication and call setup is described. After discussing whether Asterisk implements the SIP protocol correctly (Section 3.4), we show that a vicious attack on the call setup specification can be performed using the specification obtained previously is presented (Section 4). Finally, in Section 5, we evaluate the approach and point out future extensions of the PROSA tool.

2 Background

SIP is an application layer signalling protocol developed by the IETF. The core functionality of SIP is specified in RFC3261 [14], additional functionality is specified in over 40 RFCs, and nearly 30 pending SIP-related drafts[2]. SIP is used to establish, maintain and tear down multimedia sessions between two or more participants. A session can be an ordinary call between two participants, or an advanced multimedia conference session with several participants. More specific, SIP set up the session context, but does not carry multimedia content.

Fig. 1. Flow graph showing successful session establishment and termination using SIP.

We illustrate the operation of SIP with a scenario where Alice calls Bob. A session including call setup and tear down is shown in Fig. 1. While a session

[1] Automated Validation of Internet Security Protocols and Applications (AVISPA) project: `http://avispa-project.org/library/sip.html`

[2] IETF Session Initiation Protocol Charter: `http://www.ietf.org/html.charters/sip-charter.html`

generally may traverse several SIP proxies, we restrict our scenario to only one SIP proxy. The SIP proxy has three roles: (*i*) it acts as registration server, (*ii*) handles call setup, and (*iii*) routes the SIP messages and in some cases the media stream. In the call setup Alice calls Bob sending an 'INVITE' message (1, 2). A receipt for each such hop is returned by the 'Trying' message (3, 4). If Bob's phone is connected then it starts to ring and propagates the 'Ringing' message back to Alice (5, 6). When Bob answers the call, he sends an 'OK' message routed back to Alice through proxy S. Bob answers the phone in message (7), and the call content (voice) is transferred using the Real-time Transport Protocol (11). Alice terminates the call (12) and a 'BYE' message is sent. Before the 'INVITE' message (1) is sent, a SIP proxy may challenge Alice to authenticate, as formalised in Section 3.1.

For the purpose of this paper we restrict our work to how the widely used open source telephony platform Asterisk[3] [11] implements SIP. Asterisk is a private branch exchanges (PBX), whose functionality is to connect phone calls. Asterisk also supports a range of other common telephony services like voice mail, conference calls and telephone menus.

2.1 The method

In order to gain initial knowledge of the behaviour of the SIP implementation in Asterisk, we record traces from real phone traffic going through the Asterisk server on a real-world Asterisk configuration. This is done by using the network monitoring tool Wireshark[4]. Traces retrieved from Wireshark can be presented both textually and as interaction diagrams. The process of obtaining a formal specification of SIP was roughly as follows: From the core IETF standards we derived an as accurate descriptions of SIP as possible. These resulting specifications were typically incomplete, interaction diagrams showing the transmissions and message content were lacking. Traces of the call setup with real softphones were then used to supplement the incomplete specifications, with details of message credentials. Hence, based on the Asterisk traces and SIP standard we constructed the formal models manually, and analysed the specifications in the the protocol analyser PROSA [8].

The analysis process is depicted in Fig. 2. In this process both IETF documentation and the traces are used to obtain a formal description (1) which typically enhances the understanding of the implementation (4). The formal specification is then validated by PROSA (2). If there are any errors or unreasonable elements found at this stage, the formal specification is revised (3). A correct protocol specification will then be subject to hand-crafted or automatically generated attacks. The correctness of manually constructed attacks will then be checked by validation in a similar way as for non-compromised protocol specifications, thereafter simulated in PROSA. Finally, the output of the analysis is a report that either confirms the initial security requirements or points

[3] Asterisk homepage: `http://www.asterisk.org/`
[4] Wireshark homepage: `http://www.wireshark.org`

Fig. 2. Workflow for analysis of implementations

at weaknesses that break some security goals (5). If no attacks are found then the protocol is considered preliminary secure (6). If an attack is found then the protocol is not secure and a revision is made (7). Since the formal specifications are derived from a concrete implementation, the report gives feedback onto the implementation (8).

2.2 The protocol analyser PROSA

PROSA [8] is a tool developed for the specification, static analysis and simulation of security protocols. PROSA consists of three main modules: (a) a specification language based on temporal epistemic logic; (b) a static analysis module; (c) a simulator for executing intended protocols and attacks on protocols.

The static analysis module consists of algorithms for *automated refinement* of both protocol specifications and attack descriptions. The automated refinement results in an explicit specification that contains local assumptions, i.e. pre- and postconditions, for each transmission clause. Refined specifications can then be *validated*. The validation process of a trace specification is performed in two steps in PROSA: First, a tool-supported refinement of the specification is generated. This will always give a longer specification that contains information about the agents beliefs and construction of credentials, like the generation of nonces, timestamps, assumptions about keys, and cryptographic operation like encryption, decryption and hashing. Secondly, the refined specification is validated to check whether a participant in the protocol setting possesses any beliefs that have not been legally obtained through communication or cryptography.

3 Formal specification of SIP call-setup

SIP defines distinct functionality for registration, call setup and modification, call control and mid-call signalling. We refer to these parts as 'sub-protocols' since each of these parts requires its own sequence of message exchanges. We take a closer look at the sub-protocols *digest access authentication*, *call setup* and *call teardown*. *Call teardown* denotes the explicit event of terminating a call, specified by message 12-15 in Fig. 1. These are specified in a form commonly used in the literature. A protocol clause where "agent A sends a message M to the agent B" is of the form:

$$(P) \quad A \longrightarrow B \quad : \quad M$$

Messages in the protocols consist of basic entities as follows:

A, B, C, D, T, S, I, $I(A)$	agent terms
K_{AB}	symmetric key shared by A and B
N_A	nonce generated by A
W_A^Y	string containing the text Y related to A
X_A	miscellaneous entities related to A

We use three composition operators in the notation: concatenation of message content denoted by "," (comma), hashing $\mathsf{H}[M]$, and encryption denoted by $E(K : M)$, where K denotes a key and M a message content. The particular agent term $I(A)$ reads that "the intruder I impersonates as A".

3.1 Authentication

According to RFC3261 [14], there are three ways to configure SIP authentication: plain-text authentication, weak authentication, and strong authentication. Plain-text authentication sends the authentication credentials unprotected. Weak authentication is an adaptation of the HTTP digest access authentication [5] that requires a shared secret between the two participants. Strong authentication uses certificates in the same way as web browsers and servers use them. Weak authentication using a digest is by far the most common method. A typical application of digest access authentication is given by a challenger S requesting a client A to authenticate as described in the following protocol skeleton:

$$\begin{aligned}
&(\mathrm{D}_1) \quad S \longrightarrow A : N_S \\
&(\mathrm{D}_2) \quad A \longrightarrow S : W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N_A, W^{\mathrm{qop}}, \\
&\qquad\qquad \mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N_S, X_{\mathrm{nc}}, N_A, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{meth}}, W_A^{\mathrm{URI}}]]
\end{aligned}$$

Agents A and S share the symmetric key K_{AS}^{pwd}. Initially, the challenger S sends a nonce N_S to the client A. The client responds by sending the basic entities in plain text, except the password, and then the response itself. The entities involved in digest access authentication are as follows:

W_A^{uname}	username of A
W^{realm}	a protective domain
K_{AS}^{pwd}	shared password between A and S
W^{meth}	main method of message (like HTTP)
W_A^{URI}	Digest URI for client A
N_S	nonce of the challenger S
X_{nc}	nonce counter
N_A	A's nonce
W^{qop}	quality of protection

The authentication is one-way: A is authenticated to S, guaranteed by the secrecy of the shared key K_{AS}^{pwd}. Agent S can be certain that the message comes from A, since A is the only agent except S who possesses the key. Integrity of the message entities involved is provided by the fact that the hash could only be generated by A and freshness of the message is provided by the challenger nonce N_S.

3.2 The teardown sub-protocol

The trace described in Fig. 1 is one out many possible traces. *Teardown* of sessions can be performed at any stage in the session. Therefore the final four messages 12-15 in Fig. 1 and ($\mathrm{T}_{14} - \mathrm{T}_{17}$) in Fig. 4, can be considered as a teardown sub-protocol run by three agents. Instances of the teardown protocol might be running in parallel with the call setup protocol, which implies that a 'BYE' message received to any participant causes the host session of the call setup to be terminated. A SIP compliant specification where SIP methods are propagated correctly results in the following specification of teardown, extracted from the Wireshark protocol dump:

$$
\begin{array}{lll}
(\mathrm{TD}_1) & C \longrightarrow T & : W^{\mathrm{BYE}}, D, W_D^{\mathrm{URI}}, W_C^{\mathrm{Contact}}, W_C^{\mathrm{URI}}, N_C^{\mathrm{callid}} \\
(\mathrm{TD}_2) & T \longrightarrow D & : W^{\mathrm{BYE}}, C, W_C^{\mathrm{URI}}, N_D^{\mathrm{callid}} \\
(\mathrm{TD}_3) & D \longrightarrow T & : W^{\mathrm{OK}}, W_D^{\mathrm{Contact}}, W_D^{\mathrm{URI}}, N_D^{\mathrm{callid}} \\
(\mathrm{TD}_4) & T \longrightarrow C & : W^{\mathrm{OK}}, W_D^{\mathrm{Contact}}, W_D^{\mathrm{URI}}, N_C^{\mathrm{callid}}
\end{array}
$$

where C denotes the role of the agent initiating the teardown, D denotes the responder agent, while T denotes the proxy server. Instances of the teardown protocol are started by the call setup protocol, while the call-setup is terminated by the teardown protocol.

3.3 Formalising call setup permitting arbitrary teardown

Since we do not know which of the agents actively closes a session, we model that each agent starts a passive session of the teardown sub-protocol. In the example shown in Fig. 1 Alice actively closes the session, why she plays the role of B of the teardown sub-protocol. Consequently, the Asterisk server acts in the role of S. Generally, both Alice and Bob might actively tear down a session by starting an instance of teardown in the role of A.

Combining the Wireshark protocol dump with the SIP specification we get Fig. 3. In order to model this we go beyond standard protocol notation and introduce the clauses ($\mathrm{Q}_0 - \mathrm{Q}_3$), as seen in Fig. 3. The call setup involves the additional SIP methods and primitives:

W^{INVITE}	The INVITE method that indicates a request for phone call
W^{ACK}	An acknowledgement method
W^{PAR}	Proxy Authentication Required
W^{Trying}	100 Trying, receipt to a previous SIP message
W^{OK}	200 OK method notifies successful registration
W^{Ringing}	The responder's phone is ringing
W^{BYE}	Tearing down session (hang up phone)
$N_A^{\mathrm{callid}}, N_B^{\mathrm{callid}}$	Call identifiers for A (and S) and B (and S) respectively

Clause (Q_0) reads "agent A locally starts a session of the teardown protocol, such that agent A plays the responder role D, agent B plays the initiator role C, and the server S plays the server role". The notation $\mathsf{Role}(A, C)$ means that the

$$
\begin{array}{lll}
(\mathrm{Q}_0) & A & : \mathsf{start}(\mathsf{teardown}, \mathsf{Role}(B,C), \mathsf{Role}(A,D), \mathsf{Role}(S,T)) \\
(\mathrm{P}_1) & A \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{Q}_1) & S & : \mathsf{start}(\mathsf{teardown}, \mathsf{Role}(A,C), \mathsf{Role}(B,D), \mathsf{Role}(S,T)) \\
(\mathrm{Q}_2) & S & : \mathsf{start}(\mathsf{teardown}, \mathsf{Role}(B,C), \mathsf{Role}(A,D), \mathsf{Role}(S,T)) \\
(\mathrm{P}_2) & S \longrightarrow A & : W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(\mathrm{P}_3) & A \longrightarrow S & : W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_4) & A \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N_A', W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{2}{l}{\mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N_A', X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(\mathrm{P}_5) & S \longrightarrow A & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_6) & S \longrightarrow B & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{Q}_3) & B & : \mathsf{start}(\mathsf{teardown}, \mathsf{Role}(B,D), \mathsf{Role}(A,C), \mathsf{Role}(S,T)) \\
(\mathrm{P}_7) & B \longrightarrow S & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_8) & B \longrightarrow S & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_9) & S \longrightarrow A & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{10}) & B \longrightarrow S & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_{11}) & S \longrightarrow A & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{12}) & A \longrightarrow S & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{13}) & S \longrightarrow B & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
 & A \longleftrightarrow B & : \text{Media session}
\end{array}
$$

Fig. 3. Call setup with flexible teardown based on RFC 3261.

agent A plays the C role in the given protocol, similar to procedure calls with parameter substitution in ordinary programming languages. The first local clause (Q_0) then says that A initially starts listening for a possible 'BYE' message from B. The server propagates the 'BYE' and 'OK' methods involved in the teardown sub-protocol. There are two cases: Clause (Q_1) starts a session where the server S is waiting for a 'BYE' from A, while in (Q_2), the server starts a similar session waiting for B sending a 'BYE'-message.

3.4 Deviations from the SIP specification

The trace in Fig. 4 shows that the Asterisk implementation of SIP diverges from the specification described in Fig. 3 in three ways:

1. Alice's phone starts to ring (message T_7) *before* Bob is authenticated to the server. The meaning of an incoming 'Ringing' message received by Alice is that Bob has received an 'INVITE' message, and she is ready to start a call if Bob answers the call. Hence, in order to follow the SIP RFCs message (T_9) should come *before* message (T_7). Hence Alice is fooled to believe that Bob's phone is ringing, which is not the case. Therefore we simulated scenarios where the responder Bob was disconnected from the network just after receiving the 'INVITE' message. Alice still received a Ringing message. This behaviour is also confirmed in experiments using soft phones.

$$
\begin{array}{lll}
(\mathrm{T}_1) & A \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_2) & S \longrightarrow A & : W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(\mathrm{T}_3) & A \longrightarrow S & : W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_4) & A \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N'_A, W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{2}{l}{\mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N'_A, X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(\mathrm{T}_5) & S \longrightarrow A & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_6) & S \longrightarrow B & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_7) & S \longrightarrow A & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_8) & B \longrightarrow S & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_9) & B \longrightarrow S & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{10}) & B \longrightarrow S & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{11}) & S \longrightarrow B & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{12}) & S \longrightarrow A & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{13}) & A \longrightarrow S & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
 & A \longleftrightarrow B & \text{Media session (RTP or SRTP)} \\
(\mathrm{T}_{14}) & A \longrightarrow S & : W^{\mathrm{BYE}}, B, W_B^{\mathrm{URI}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{15}) & S \longrightarrow A & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{16}) & S \longrightarrow B & : W^{\mathrm{BYE}}, A, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{17}) & B \longrightarrow S & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}}
\end{array}
$$

Fig. 4. A trace of call setup and teardown using Asterisk as server S.

2. The acknowledgement received by Bob in message (T_{11}) arrives *before* Alice sends the message in clause (T_{13}). However, at this point Bob is mislead to believe that Alice has acknowledged the Ringing request from Bob.
3. After teardown initiated by Alice (T_{14}), the 'OK' message from S to Alice (T_{15}) is sent *before* the related 'OK' message is sent from Bob. This breaks the specification, since (T_{15}) would implicate that Bob has received the 'BYE' message, which is not the case in the implementation.

This implementation can lead to unexpected results: An obvious attack targets the Ringing method: An intruder I could act as an eavesdropper until clause (T_7), then take over Bob's session entirely, kick Bob out of the call, and for the rest of the trace masquerade as Bob. Persons that are used to the particularly quick response (immediate ringing) from Asterisk based VoIP would not be alerted when the intruder I impersonates as Bob later in the session.

4 Attack on the call setup

An attack on the registration protocol of SIP has been found recently [9], yet in the following, we consider attacks that do not rely on successful registration attacks. We assume that the attacker I is as powerful as the Dolev Yao attacker [3] who controls the entire network, can intercept any message, impersonate as any other agent, and inject whatever entity it knows into SIP messages. Cryp-

Fig. 5. Hijacking the initiator and the responder.

tography is assumed to be perfect, no brute force attacks on the underlying cryptographic algorithms are considered in this paper.

In the following we describe how it is possible for an attacker to hijack both the initiator and responder roles. In the initial part of the attack, described in Fig. 6, the intruder only passively listens in the authentication sub-protocol. In the protocol clauses ($P_{1.1.a}$) through ($P_{1.4.b}$) the intruder acts as a passive man-in-the-middle, obtaining information from plain text entities. From the knowledge gained during the initial eavesdropping, an attacker can perform a combined attack on both the caller and the callee.

In the attack, as shown in Fig. 5, the attacker Ivory (denoted I in Fig. 6) begins by eavesdropping the initial four messages concerned with establishing the call and authenticating the caller Alice to the server. Then Ivory tears down Alice's session prematurely by using a 'BYE' message, and thereafter terminates Bob's session before he has entered the media session. Before the media session, the attacker has taken over the call, and can start a conversation with agent Frank (denoted F in Fig. 6).

The attacker tears down the session after pretending to be Alice. The server S cannot discover that the two local sessions at each calling party are teared down. The attack effectively breaks the authenticity of the participants, and therefore results in an identity management problem. The consequence could be that the intruder I could set up an arbitrary call, that Alice is billed for. Another consequence of this attack could be that the logs, that telephony providers are obliged to carry out by legislation, are incorrect. Hence, the attack shows that non-repudiation is broken as well.

5 Discussion and Conclusions

Our method reveals concrete attacks that indicate where improvements in the protocol are necessary. We discovered that the well-known SIP implementation Asterisk deviates from the SIP specification, and found a severe call hijack attack, where intruders can completely take over a phone call.

Our work provides the designers of VoIP protocols with a set of tools for protocol analysis. The PROSA language and framework can be used to formally specify and analyse protocols and their specific implementations in a rigorous way. The use of traces, and the analysis with PROSA, can help to detect differences between protocol specification and implementation. The behaviour of implementations is therefore analysed, and treated as specification for a variant of the employed protocols. Attacks could be designed and tested rapidly compared to the traditional approaches as described in [13] and [4].

References

1. J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka. Security Mechanism Agreement for the Session Initiation Protocol (SIP). RFC 3329 (Proposed Standard), January 2003.
2. Wafaa Bou Diab, Samir Tohme, and Carole Bassil. VPN analysis and new perspective for securing voice over VPN networks. *ICNS*, 0:73–78, 2008.
3. Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
4. David Endler and Mark Collier. *Hacking Exposed VoIP: Voice over IP Security Secrets and Solutions*. McGraw-Hill Osborne Media, Nov 2006.
5. J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard), June 1999.
6. D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis. SIP Security Mechanisms: A state-of-the-art review. In *Proceedings of the Fifth International Network Conference (INC 2005)*, pages 147–155, July 2005.
7. Prateek Gupta and Vitaly Shmatikov. Security Analysis of Voice-over-IP Protocols. In *Computer Security Foundations Symposium, 2007. CSF '07. 20th IEEE*, pages 49–63, 2007.
8. Anders Moen Hagalisletto. *Automated Support for the Design and Analysis of Security Protocols*. PhD thesis, University of Oslo, December 2007.
9. Anders Moen Hagalisletto and Lars Strand. Formal modeling of authentication in SIP registration. In *Second International Conference on Emerging Security Information, Systems and Technologies SECURWARE '08*, pages 16–21, Aug 2008.
10. D. Richard Kuhn, Thomas J. Walsh, and Steffen Fries. Security Consideration for Voice over IP Systems. Sp 800-58, National Institute of Standards and Technology (NIST), Jan 2005.
11. J. Meggelen, J. Smith, and L. Madsen. *Asterisk: The Future of Telephony*. O'Reilly Media, Sep 2005.
12. David Persky. VoIP Security Vulnerabilities. Technical report, SANS Institute, 2007.
13. Thomas Porter. *Practical VoIP Security*. Syngress, Mar 2006.
14. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916.
15. S. Salsano, L. Veltri, and D. Papalilo. SIP security issues: The SIP authentication procedure and its processing load. *Network, IEEE*, 16:38–44, 2002.
16. Henry Sinnreich and Alan B. Johnston. *Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol*. John Wiley & Sons, Inc., New York, NY, USA, second edition, August 2006.
17. Jianqiang Xin. Security issues and countermeasure for VoIP. Technical report, SANS Institute, 2007.
18. Ruishan Zhang, Xinyuan Wang, Xiaohui Yang, and Xuxian Jiang. Billing Attacks on SIP-Based VoIP Systems. USENIX, Aug 2007. First USENIX Workshop on Offensive Technologies (WOOT '07).

$(P_{1.1.a})$	$A \longrightarrow I(S)$	:	$W^{\text{INVITE}}, A, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{1.1.b})$	$I(A) \longrightarrow S$	:	$W^{\text{INVITE}}, A, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{1.2.a})$	$S \longrightarrow I(A)$	:	$W^{\text{PAR}}, W_A^{\text{uname}}, W^{\text{realm}}, N_S, A, B, N_A^{\text{callid}}$
$(P_{1.2.b})$	$I(S) \longrightarrow A$	:	$W^{\text{PAR}}, W_A^{\text{uname}}, W^{\text{realm}}, N_S, A, B, N_A^{\text{callid}}$
$(P_{1.3.a})$	$A \longrightarrow I(S)$	:	$W^{\text{ACK}}, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{1.3.b})$	$I(A) \longrightarrow S$	:	$W^{\text{ACK}}, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{1.4.a})$	$A \longrightarrow I(S)$	:	$W^{\text{INVITE}}, A, B, N_S, W_A^{\text{Contact}}, W_A^{\text{URI}}, X_{\text{nc}}, N'_A, W^{\text{qop}}, N_A^{\text{callid}}$ $\mathsf{H}[\mathsf{H}[W_A^{\text{uname}}, W^{\text{realm}}, K_{AS}^{\text{pwd}}], N'_A, X_{\text{nc}}, N_S, W^{\text{qop}}, \mathsf{H}[W^{\text{INVITE}}, W_B^{\text{URI}}]]$
$(P_{1.4.b})$	$I(A) \longrightarrow S$	:	$W^{\text{INVITE}}, A, B, N_S, W_A^{\text{Contact}}, W_A^{\text{URI}}, X_{\text{nc}}, N'_A, W^{\text{qop}}, N_A^{\text{callid}}$ $\mathsf{H}[\mathsf{H}[W_A^{\text{uname}}, W^{\text{realm}}, K_{AS}^{\text{pwd}}], N'_A, X_{\text{nc}}, N_S, W^{\text{qop}}, \mathsf{H}[W^{\text{INVITE}}, W_B^{\text{URI}}]]$
$(T_{2.3.2})$	$I(S) \longrightarrow A$	:	$W^{\text{BYE}}, B, W_B^{\text{URI}}, N_A^{\text{callid}}$
$(T_{2.3.3})$	$A \longrightarrow I(S)$	:	$W^{\text{OK}}, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{2.5})$	$S \longrightarrow I(A)$	:	$W^{\text{Trying}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_A^{\text{callid}}$
$(P_{2.6.a})$	$S \longrightarrow I(B)$	:	$W^{\text{INVITE}}, A, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.6.b})$	$I(S) \longrightarrow B$	:	$W^{\text{INVITE}}, A, B, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.7.a})$	$B \longrightarrow I(S)$	:	$W^{\text{Trying}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.7.b})$	$I(B) \longrightarrow S$	:	$W^{\text{Trying}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.8.a})$	$B \longrightarrow I(S)$	:	$W^{\text{Ringing}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.8.b})$	$I(B) \longrightarrow S$	:	$W^{\text{Ringing}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(T_{2.4.2})$	$I(S) \longrightarrow B$	:	$W^{\text{BYE}}, A, W_A^{\text{URI}}, N_B^{\text{callid}}$
$(T_{2.4.3})$	$B \longrightarrow I(S)$	:	$W^{\text{OK}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.9})$	$S \longrightarrow I(A)$	:	$W^{\text{Ringing}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_A^{\text{callid}}$
$(P_{2.10})$	$I(B) \longrightarrow S$	:	$W^{\text{OK}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_B^{\text{callid}}$
$(P_{2.11})$	$S \longrightarrow I(A)$	:	$W^{\text{OK}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_A^{\text{callid}}$
$(P_{2.12})$	$I(A) \longrightarrow S$	:	$W^{\text{ACK}}, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(P_{2.13})$	$S \longrightarrow I(B)$	:	$W^{\text{ACK}}, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_B^{\text{callid}}$
	$\mathbf{I(A) \longleftrightarrow F(B)}$	:	**Media session**
$(T_{2.2.1})$	$I(A) \longrightarrow S$	:	$W^{\text{BYE}}, B, W_B^{\text{URI}}, W_A^{\text{Contact}}, W_A^{\text{URI}}, N_A^{\text{callid}}$
$(T_{2.5.2})$	$S \longrightarrow I(B)$	:	$W^{\text{BYE}}, A, W_A^{\text{URI}}, N_B^{\text{callid}}$
$(T_{2.2.3})$	$I(B) \longrightarrow S$	:	$W^{\text{OK}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_D^{\text{callid}}$
$(T_{2.2.4})$	$S \longrightarrow I(A)$	:	$W^{\text{OK}}, W_B^{\text{Contact}}, W_B^{\text{URI}}, N_A^{\text{callid}}$

Fig. 6. Formal attack when hijacking the initiator and the responder.