Bringing VoIP Signatures to Mobile Devices

Ronald Marx and Nicolai Kuntze
Fraunhofer Institute for Secure Information Technology SIT
Darmstadt, Germany
{marx, kuntze}@sit.fraunhofer.de

Hagen Lauer
Technische Hochschule Mittelhessen
Gießen, Germany
hagen.lauer@mni.thm.de

ABSTRACT

With the advent of LTE[1] another technology is gaining momentum in mobile communication, viz. Voice-over-IP (VoIP). While LTE provides state-of-the-art security features such as confidentiality, integrity, and authenticity, non-repudiation of voice communication over LTE is not addressed. At the same time the utilization of mobile phone devices for business communication is already common practice. However, applications supporting specific business requirements, such as the confirmability of statements made in a phone call, are missing.

VoIP Signatures is a recognized concept to achieve both non-repudiation for VoIP calls as well as integrity protection against the manipulation of recorded conversations. We present the integration of this concept with a VoIP client running on a mobile device. The implementation concept has been optimized for applicability on standard smartphones with regard to given characteristics such as processing power limitations. The implementation concept is introduced and the performance has been evaluated, resulting in a promising statement regarding the applicability of our approach on future mobile devices providing Voice-over-LTE functionality.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Security, Design, Performance

Keywords

Voice-over-IP, signatures, security, non-repudiation, integrity

[1]Long Term Evolution

IPTComm '13, October 15 - 17 2013, Chicago, IL, USA

http://dx.doi.org/10.1145/2554666.2554669

1. INTRODUCTION

At the end of 2012 the International Telecommunication Union (ITU) estimated the number of mobile subscriptions to 6.8 billion, which equals 96 percent of the world population [7].

In line with this development, mobile communication has also become an essential part of most businesses. Nowadays, a considerable amount of business communication is made by mobile phone. For instance in UK: already since 2007 minutes for business calls over cellular exceeded those over fixed line [12]. Mobile voice communication enables businessmen to easily stay connected to their customers and business partners. In real-time businesses such as stock exchange, instantaneous communication is a key factor for success, since a delay in forwarding buying or selling decisions can result in heavy monetary loss. Further examples of businesses exist in which placing a phone call is considered more convenient, efficient, or productive than writing an email. One reason for this effect is the opportunity for immediate mutual discussion and, if required, clarification of given statements and orders.

Unfortunately, one drawback of using mobile voice communication for business purposes is the lack of evidence and documentation of the agreements made. Non-repudiation of the content of a voice communication would help to dispose this effect. To fulfil this requirement, the following measures have to be taken.

- (1) Protection of the integrity of voice conversations,
- (2) authentication of speakers[2], and
- (3) electronic signatures over voice conversations.

Combining these three features would allow for the conclusion of engaging business contracts by the use of a mobile phone.

The initial architectural and technological concepts for implementing non-repudiation for Voice-over-IP (VoIP) conversations have already been introduced in [5]. Thus, this paper will outline these concepts just briefly and will focus on the concepts' application and implementation on mobile devices.

This paper will introduce a secure solution for achieving non-repudiation of VoIP calls on mobile devices. The paper is organized as follows. Section 2 specifies requirements regarding signatures on VoIP communication. Section 3 provides some basic background on Session Initiation Protocol

[2]An approach for the mutual authentication (based on electronic ID cards) of the parties of a VoIP conversation was presented in [11].

(SIP), Session Description Protocol (SDP), and Real-time Transport Protocol (RTP). In Section 4 we introduce the concept of how to integrate VoIP Signatures with a mobile device, followed by a description of the implementation steps in Section 5. Section 6 finally concludes the paper and provides a brief outlook on further research activities.

2. REQUIREMENTS TOWARDS VOIP SIGNATURES

In the following section requirements towards signatures on VoIP communication are presented from the perspective of security, legal framework, and functionality.

2.1 Security and Legal Requirements

The central requirements for achieving non-repudiation by signing VoIP are related to security, as also discussed in [6] and [8]. Of the well known trinity of information security requirements – Confidentiality, Integrity, and Availability – , integrity is the central one to achieve non-repudiation for digital, packet-based, natural language communication. I.e, in order to provide non-repudiation, it must be ensured that a communication was not changed at any point in time, be it during transmission or after (or even during) the required process of archiving. Furthermore, integrity also must be applied to any relevant meta-data created or used during a call. In particular this comprises data that authenticate the communication partners, or at least the partner who digitally signs the set of communication-related data.

Due to the special features of voice communication, i.e., a bidirectional, full-duplex interactive conversation, only both channels together provide the necessary context to fully understand the content of the conversation and to make use of the inherent security that is provided by interweaved, natural language conversations. To ensure that, once recorded, parts of the conversation are not exchanged with each other or dropped in order to manipulate the content of the conversation, the envisaged system provides a feature we call cohesion. Cohesion is the temporal sequencing of the communication and its direction. These two characteristics need to be protected in a way that makes later tampering practically unfeasible. This could be achieved by sufficiently strong cryptographic methods. Cohesion as a feature related to time entails a subsidiary requirement, namely the secure assignment of a temporal context to a conversation. Each conversation has to be reliably associated with a certain point in time, which must be as close as possible to the conversation's start and the initiation of the signing (note that assignment of a signing time is a legal requirement for qualified electronic signatures according to the European Signature Directive [16] and pertinent national regulations). Drift of the time base should be mitigated during a signed conversation.

Finally, cohesion also refers to qualitative aspects of the communication channel. A signatory is well advised not to sign a document which is illegible or ambiguous. In the digital domain this relates to the presentation problem for electronically signed data [10]. In analogy, the quality of the VoIP channel must be maintained to a level that ensures understandability to both partners during the time span in which the conversation is signed.

To invoke a telephony conversation as a lawful evidence, it must have been recorded while taking place. Providing a functionality to securely archive a once recorded and signed voice conversation is a task that can be easily fulfilled by either party of the conversation. However, naming or defining particular methods for the secure archiving of VoIP communications is not in the scope of this paper. One possible method has been described in [5], also having been combined with the VoIP signatures procedure applied in our approach. Considerations of long-term archiving aspects for signed digital data can be found in [9]. Since these can be applied at the transport layer, confidentiality features for VoIP conversations are as well out of scope of this paper.

2.2 Functional and Usability Requirements

Further requirements have to be considered regarding the efficiency of the system design and implementation. First, it is highly desirable, both from a security as well from an efficiency viewpoint, to sign and secure the VoIP conversation as "close" as possible to its transmission, and conceptually close to the actual VoIP stream. Simplicity of the implementation should minimise the effect on existing systems and infrastructures, e.g., client-side requirements should be minimised. A tight integration is required to enable the utilisation of existing infrastructures without, or with only minor, changes. An efficient use of memory, bandwidth, storage space, and computational resources can be achieved by basic conceptual design decisions. Furthermore, scalability of the concept to a large number of concurrent calls is a necessity in real business environments. This means that centralised signature creation infrastructures should be avoided if possible. Finally, any architecture that copes with VoIP security issues needs to appropriately take packet loss into account, in particular in view of the cohesion requirement.

3. SESSION INITIALIZATION AND MEDIA TRANSFER IN VOIP

VoIP communication in the area of PSTN and mobile networks is mostly based on the SIP/RTP protocol family. An applicable solution requires therefore to be integrated into these protocols and also to take advantage of the protocol features. This section introduces the relevant features of the underlying communication protocol as far as they are relevant for the concept presented in the latter chapter.

3.1 SIP and SDP

The Session Initiation Protocol (SIP) [14] defined by IETF is a signaling protocol mainly designed to manage creation, modification and termination of Multimedia-over-IP sessions such as VoIP calls. The SIP session setup procedure consists of (at least) a SIP three way handshake (INVITE request, 200 OK response, ACK request). These SIP messages are exchanged between the SIP clients of the caller (sending INVITE and ACK) and the callee (sending 200 OK) to establish a connection-oriented state of communication.

Note that SIP itself does not define any information regarding the type or characteristics of media that are going to be exchanged between the communicating parties. These details are additionally negotiated between the clients in an end-to-end offer-answer manner by the use of SDP (Session Description Protocol) [3]. Therefore, SDP media descriptions are carried in the message bodies of the SIP INVITE request (media details offered by calling party) and the corresponding 200 OK response (media details answered by called

party). The most relevant information described comprise the type of media (such as audio and/or video), codec choice (such as G.711 or iLBC audio codecs), and the IP address and UDP port number to which the other party should send its outgoing media stream(s).

3.2 RTP

Once a session is established, the Real-time Transport Protocol (RTP) [15] is typically used for media transport in case of a VoIP call. Therefore, the digitalized voice stream is split up into isochronous sequences, which are subsequently encoded and sent over the IP network as a virtual media stream. A virtual stream consists of a series of consecutive RTP packets, each carried in a single UDP datagram. Each RTP packet is assigned to a distinct media stream by its unique SSRC (Synchronization Source) identifier that comes as a part of the RTP packet header. The RTP header also includes a sequence number which is incremented for every subsequent packet of a media stream to allow for media replay in the correct order, and for packet loss recognition. Note that an RTP media stream is by definition unidirectional, i.e. in case of a VoIP call, two RTP streams are typically exchanged (one media stream from caller to callee, and a second stream vice versa). Within this work, beside for media data transport, RTP is also used to carry XML-based signing information for the sake of implementation simplification. If a confidential exchange of information is required, the data have to be encrypted and transported via Secured Real-Time Transport Protocol (SRTP) [2], providing media encryption and integrity, coming as a lightweight extension to RTP.

4. VOIP SIGNATURES AND GENERAL IMPLEMENTATION CONCEPT

Our work focuses on the integration of VoIP Signatures with an existing type of open-source SIP-based VoIP client running on a mobile platform. The general concept of VoIP Signatures has originally been introduced in [5]. It provides security features such as integrity and non-repudiation for VoIP calls that are going to be recorded and archived. The concept is based on the logical splitting of a VoIP stream into defined intervals (e.g., an interval might represent a one-second portion of the voice stream). The splitting is performed in real-time while the call is ongoing. An interval consists of a series of consecutive RTP packets, for which hash values are computed. The hashes are stored in a meta data set with a defined structure together with the sequence numbers of the corresponding RTP packets, and the data set is electronically signed. The signature of an interval is added to a new meta data set together with the RTP sequence numbers and hashes that describe the next interval. The new data set is electronically signed and the signature will be included in the subsequent data set, and so forth. Thus, consecutive intervals are logically chained so that an attacker cannot cut out, exchange, reorder or manipulate parts of the media stream. The meta data sets are exchanged between the communicating parties and are mutually verified, which prevents false pretenses regarding the course of the call or packet loss effects. The original concept takes into account the bidirectional nature of a VoIP call by considering intervals of both communication directions in the interval chaining. Thus, also the chronology with regards to the content of the VoIP call is preserved by applying VoIP Signatures. As an example, this concept could be leveraged to provide evidence of a verbal agreement between a bank officer and a customer giving investment orders.

Figure 1 shows the overall architecture of a VoIP client equipped for VoIP Signatures, running on a mobile platform. Orange elements (i.e. *SIP Client* and *RTP stream In/Out*)) are expected to be parts of the existing client. Cyan elements (i.e. *Provider* and *Interface/Listener* incl. related interface blocks) are interfaces or components that act as an abstraction layer between the VoIP client and the VoIP Signatures architecture, to which the green elements (i.e. *Archiver/Signer* and *Action Stack*) belong.

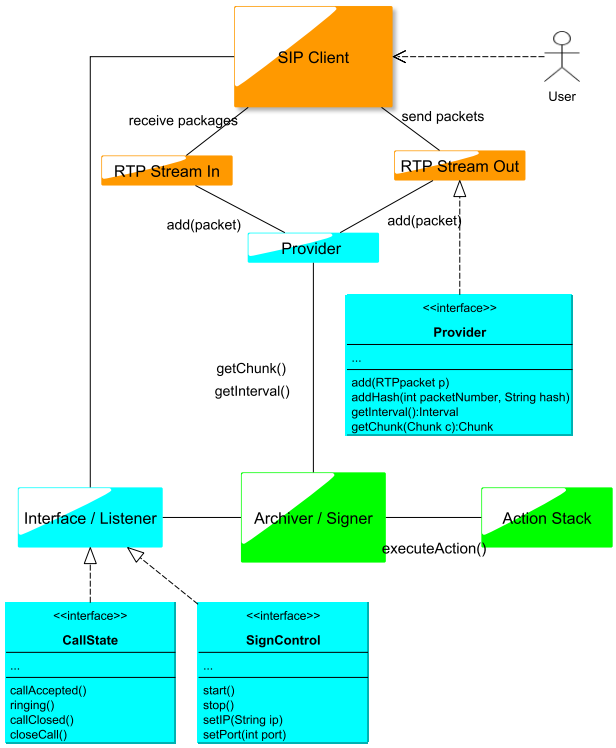

Figure 1: General implementation concept and main components

The components shown are briefly described in the following.

- SIP Client: VoIPS Signatures reuses the SIP-related parts of the target VoIP client (including SIP stack and UI).
- RTP Stream In/Out: The RTP stack of the target VoIP client is also reused. Access to the RTP streams is required because, amongst others, data such as RTP sequence numbers have to be processed by components of the VoIP Signatures architecture.
- Provider: This component is an interface between the RTP-related units of the VoIP Client and the signing functions. It supports the signing procedure by providing necessary information (such as RTP packet sequence numbers) to the signing units.

- Interface/Listener: These connect the user interface of the VoIP client with *Archiver* and *Signer*. On the one hand, controls to start and stop signing and archiving is provided to the VoIP client (*SignControl* function, to be implemented at *Signer* side). On the other hand call state and session information are provided to *Archiver* and *Signer* (*CallState* function, to be implemented at the client side).

- Action Stack: As the mobile phone can act as both, signer and archiver, the action stack component includes both stacks, one for signing, and one for archiving. It contains the state objects required to "speak" the VoIP signatures protocol, in both the signing and the archiving VoIP terminal.

- Archiver/Signer: *Archiver* and *Signer* are closely related to each other. However, their respective functionalities can be clearly distinguished. The duties of the *Archiver* is to request, to receive and subsequently to verify the VoIP signatures. The *Signer* element is in charge of signing both incoming and outgoing RTP media streams. Depending on which role the VoIP client is acting as either the *Archiver* or *Signer* element is used.

In the following section the interaction of *Archiver* and *Signer*, which are running on two different mobile devices, is described.

4.1 Signing Information Transport and Startup Process

Figure 2 shows the mutual data exchange of signing information between *Archiver* and *Signer*. If one party of a VoIP conversation wishes to receive signing data to be stored together with the RTP media, it will activate the signature function. As a consequence the *Archiver* component is triggered, which is implemented with the VoIP Signatures client extension. Utilizing RTP as a transport protocol, the *Archiver* now sends a *greeting* data element to the corresponding *Signer* function implemented with the VoIP client of the communication partner (see top portion of Figure 2). The *greeting* data element already includes a set of basic information about the call, such as the start time, the identities of the communication parties, the public key to verify the signatures and the codec used (see the general XML code of the greeting element given in Listing 1). The public key is part of a key pair which is generated at the beginning of a call. Note that the key can be set to different sizes, but it should be big enough to allow long term storage.

Listing 1:

```
<g>
 <t>Time</t>
 <a>Party A</a>
 <b>Party B</b>
 <c>Codec<c>
 <k>Session-Key</k>
 <s>Signature</s>
</g>
```

Upon receipt, the *Signer* signs the data set, includes the signature into the *greeting* data element and sends it back to the *Archiver*, who in turn verifies and stores the signature.

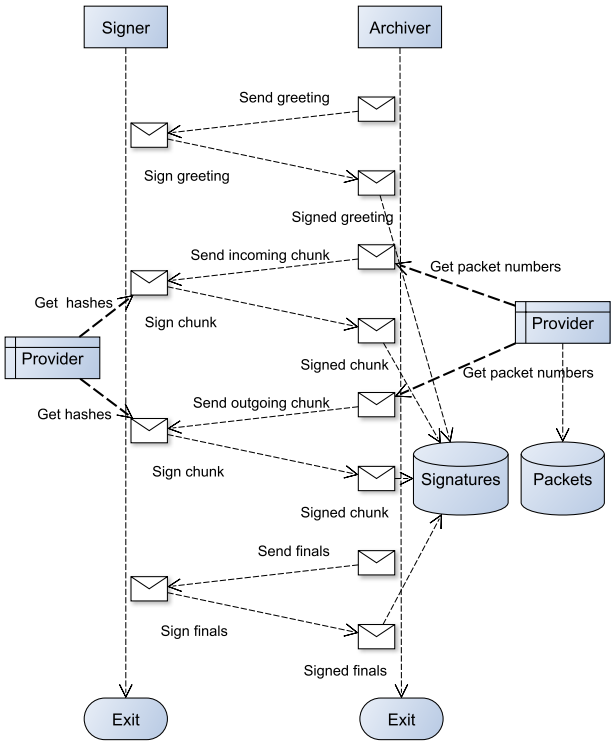

Figure 2: Archiver and Signer interaction

4.2 RTP Stream Signing

After having received a number of RTP packets of the VoIP stream, the *Archiver* prompts the *Signer* to sign the media data. Therefore it sends a *chunk* data element to the *Signer* (see middle portion of Figure 2), including the sequence numbers of the received RTP packets. The *Signer* then fills in the signatures of each packet from the packet numbers and removes the numbers from the list that he could not find (due to loss or possible manipulations). Subsequently, the data element is forwarded back to the *Archiver*. Listing 2 shows the XML data structure of a *chunk* element.

Listing 2:

```
<c>
 <p>Previous signature</p>
 <t>Time</t>
 <d>direction (in/out)</d>
 <n>Packet numbers</n>
 <h>Hashes of packets</h>
 <s>Signature</s>
</c>
```

Note that both communication directions are considered by VoIP Signatures to be able to prove and sign the interactive course of the conversation. Therefore the terms *incoming* or *outgoing chunk* shown in Figure 2 refer to the direction of the media packets from the point of view of the *Archiver*.

Figure 3 shows the interworking of *Archiver* and *Signer* in case of a chunk (either outgoing or incoming) to be signed. Orange elements (i.e. all elements shown in the first (topmost) row level within the figure, plus *Provider* in the second row level) are sent by the *Archiver*, containing the information that has to be signed. Blue elements (i.e.

all elements shown in the third row level plus *Hash* in the lowest row level) are created by the *Signer* as a response to the request of the *Archiver*.

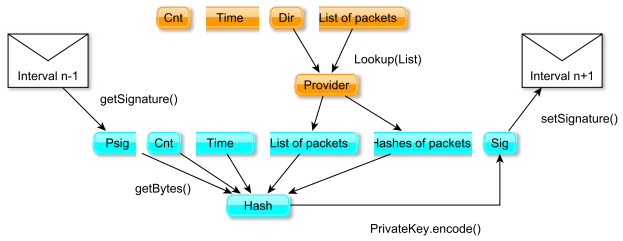

Figure 3: Sources of data required for chunk signing

Once the call has been terminated, a last interaction is performed between *Archiver* and *Signer* to sign the end of the conversation. The respective XML data structure is given in listing 3.

Listing 3:

```
<f>
 <p>Previous signature</p>
 <t>Time</t>
 <r>CallEnd-Reason</r>
 <c>Certificate</c>
 <k>Session-Key</k>
 <s>Signature from certificate</s>
</f>
```

Note that chunk and final messages contain a previous signature filled in by the *Signer*. The previous signature refers to the signature of the last sent chunk message and it is used to pinpoint the exact order of messages. By adding the signature of the previous message to the new message, hashing and signing it, the messages are chained to each other. This makes it impossible to remove, manipulate or exchange messages without breaking this chain.

Summarizing the respective action of both *Archiver* and *Signer* required for their interworking, Figure 4 shows the step-by-step processing in both the *Archiver* (left flow-chart) and the *Signer* (right flow-chart) in a simplified manner.

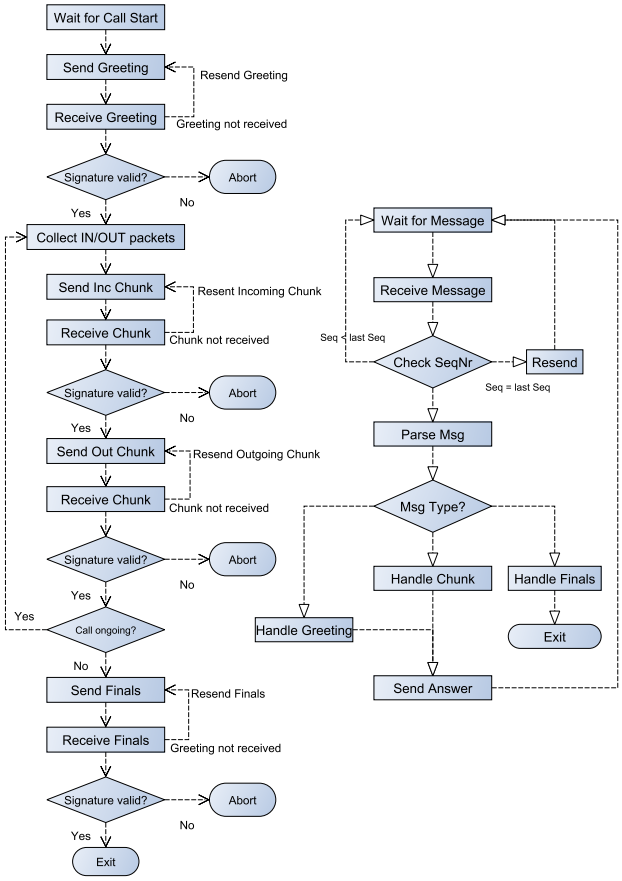

Figure 4: Processing steps performed by *Archiver* (left) and *Signer* (right)

5. IMPLEMENTATION ON ANDROID

This section comprises the implementation steps that are necessary for the realisation of the concept that has been introduced in Chapter 4.

5.1 Integration in SipDroid

For implementation we favoured Android over other operating systems for mobile devices because it is a widely-used platform for mobile applications and it offers comprehensive and free support for developers. Furthermore, many VoIP and SIP clients for Android are available, most of them are closed source though. LinPhone[3] and SipDroid[4] are open source, well documented and under continuous development. LinPhone uses a complex streaming API from Oracle's Java Media Framework to manage its RTP streams, while SipDroid uses its own RTP implementation – SipDroid provides easier access to RTP streams. In the end we decided for SipDroid mainly because its implementation of the RTP streaming is easier to extend.

To integrate our new architecture and extend SipDroid, we essentially need to alter RTP streams and SIP signaling. SipDroid uses an external, but well known and well documented SIP stack called MjSip. MjSip follows a layered approach and offers API's (Figure 5) to each layer:

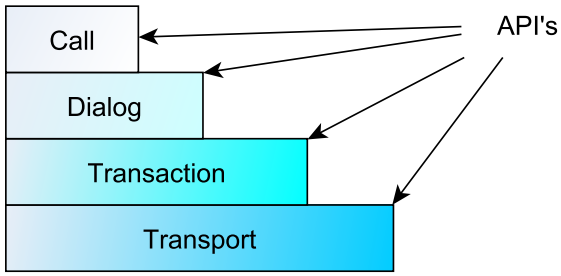

Figure 5: Structure of MjSip's API

- Transport layer: It is the lowest layer, responsible for SIP message transport. It comprises transport proto-

[3]http://www.linphone.org
[4]https://code.google.com/p/sipdroid

cols like TCP and UDP and provides an interface for sending and receiving SIP messages.

- Transaction layer: This layer handles protocol-specific SIP transaction mechanisms such as message re-sends, timeouts and request-response message construction. SIP messages generated by the transaction layer are handed down to the transport layer for sending, and messages received by the transport layer are handed up to the transaction layer for processing.
- Dialog layer: This layer manages several SIP sessions in which the considered SIP client is involved, and also distinguishes the respective communication partners. Each SIP session consists of one dialog.
- Call layer: This is the top and most abstract layer. It offers APIs (listeners) to abstract call-related states and events such as ringing, occupied, hangup, or call accepted.

The specific SIP client extensions introduced within this work are designed to interact with the SIPDroid call layer. Therefore we make use of the following methods offered by the call layer and its listener.

- *onCallIncoming(Call call, NameAddress caller, String sdp, Message invite)*: This method is used to launch the signing process (in case that the callee is referred to be the signer). At this point it is required to create a key pair to sign incoming messages. Java's Crypto API offers both hash and RSA keygen algorithms. The given call object delivers all the necessary information for a proper *greeting* parsed from the SDP messages. In a further step the *accept()* method is called. This method is invoked as a result of a user action (when the user answers to call). From this point incoming packets in the signing channel are considered.
- *onCallRinging(Call call, Message resp)*: This method is invoked by the caller (in case that the caller is the archiver). This method is only called when the communication partner is available and has received a notification for an incoming call. At this point the archiving process is launched, and the signing process is started by sending a greeting message.
- *onCallAccepted(Call call, String sdp, Message resp)*: Just like *onCallRinging()* this method is invoked by the caller and it is used to notify the user of an established call and to start the archiving process. The archiver can now check the buffers and start sending messages to sign intervals from these buffers.
- The methods *hangup()*, *onCallClosing(...)* and *onCallClosed(...)* are used to indicate the end of the signing process. Invoking any of these methods results in a final message to be sent. After the message has been processed, both parties quit their signing process.

In a next step an additional RTP stream is established, comprising the signing data. To create the corresponding RTP channel for signing the audio data, MjSip offers easy access to the SIP session setup. Therefore, the *addMediaDescriptor()* method is applied to add a new media descriptor to the SDP portion. Note that, alternatively, the signing data portions could also be sent within the same channel (i.e., to the same UDP socket) like the media data. In this case, a different SSRC for the signing and the audio packets would be used to separate them.

In a last step, the RTP packages received and sent are added to the buffers of the signing system: The idea is to add a provider object to incoming and outgoing data streams. SipDroid implements an audio receiver as well as an audio sender and launches them in separate threads in methods such as *onCallIncoming()* or *onCallAccepted()*. Subsequently, both the audio receiver and sender get access to this buffer and deposit either the RTP packets (archiver) or hashes of RTP packets (signer). The signing/archiving thread fetches the packets through these two buffer objects.

5.2 Performance evaluation

Network communication and the signature process are most likely to create a delay in the extended software. While network delay cannot be influenced by implementation, the delay caused by signing data is a major factor for the performance of our VoIP signatures extension. A great amount of measurements was taken in order to evaluate the performance which will be presented in this section. Please note that the measured software is a prototypical implementation that has not been fully optimized for performance, yet. Moreover, the performance measured was found to depend, amongst others, on the type of the mobile device used to sign the voice stream.

We performed two sets of measurements. First, with a Samsung Nexus S running Android 4.1.2 and second, with a Samsung Galaxy S running Android 4.2.2. Phones were communicating with a peer on which the same extended SIPDroid software was run. The time measured is the delay for performing the chain signature of 2048-bit on the RTP chunks, including building and sending out the packet. This is done in variable intervals. The faster a signer processes the data the shorter are the intervals. However, intervals can not become longer than one second. Note that the interval length has not been set to a fixed value to allow for the adaptation to processing power conditions on different mobile phones. Communication between peers is realized through a IEEE 802.11g Wi-Fi router. The router also connects to the SIP server, which is also located in the same subnet.

Using the aforementioned test-bed set-up an average delay of *20.37 ms* was measured for the Nexus S, and *23.55 ms* for the Galaxy S. The measured values have a standard distribution of *24.66 ms* in terms of the Nexus S, and *24.33 ms* in terms of the Galaxy S. The results come as mean values from 1000 measurements.

Figure 6 shows the distribution of the measurement results among different delay value ranges for both phone types. The ranges were chosen in a way that qualified statements can be easily derived from the graph. E.g., running the VoIP signature extension on a Nexus S phone, in more than 52 percent of all measurement runs, the delay resulting from the signing process did not exceed 15 ms. On a Galaxy S phone, a delay lower than 20 ms was achieved in more than 56 percent of all runs. For both phones it can be pointed out that a delay not exceeding 50 ms was measured in 90 percent (Galaxy S) and 92 percent (Nexus S) of all runs, respectively. These values show that the extended software adds a reasonable and tolerable delay, which does not hinder

the application of VoIP signatures on mobile devices.

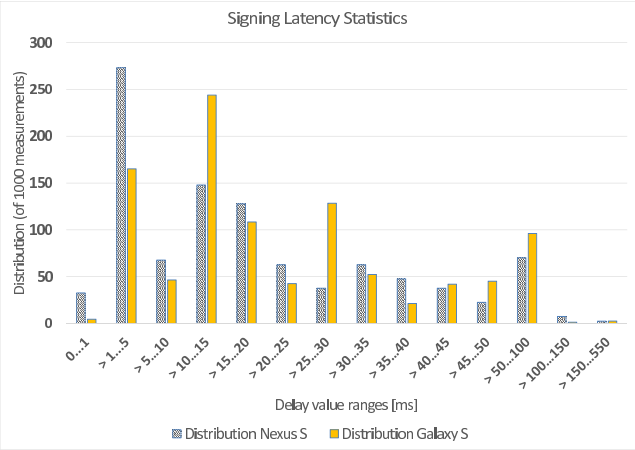

Figure 6: Distribution of achieved signing latencies

6. CONCLUSION AND OUTLOOK

This paper demonstrates the feasibility to sign VoIP communication in real-time on a mobile device. Therefore a proven concept (namely VoIP Signatures) has been applied that provides non-repudiation to VoIP calls with regard to the integrity of a call recording, and hence its evidentiary value. An integration concept has been carefully designed to address specific conditions given on typical mobile devices, such as processing power limitations. The results gained from the performance evaluation show that VoIP Signatures can be successfully implemented, and run on a mobile Android device with satisfying performance. This provides the basis for non-repudation and content integrity for VoIP calls on mobile phones. Our future work in this field will address the extension of the VoIP Signatures concept towards different media such as video streaming, and its multimedia application on mobile devices. Furthermore, the consumption of battery power through the proposed implementation could be examined, and the results could help to further optimize the signatures approach for use on mobile devices.

7. ACKNOWLEDGMENTS

The authors would like to thank their colleague Dr. Frank Gerd Weber for his support and his valuable contributions.

8. REFERENCES

[1] H. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP Authentication. In *Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security*, IAS '08, pages 237–242, Washington, DC, USA, 2008. IEEE Computer Society.

[2] M. Baugher et al. Secure Real-Time Transport Protocol (SRTP). IETF Request for Comments: RFC 3711, Mar. 2005.

[3] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006.

[4] C. Hett. Security and Non-Repudiation for Voice-over-IP conversations. Thesis, Technical University Darmstadt, Germany, July 2006.

[5] C. Hett, N. Kuntze, and A. U. Schmidt. A secure archive for Voice-over-IP conversations. In D. S. et al., editor, *Proceedings of the 3rd Annual VoIP Security Workshop (VSW06)*. ACM, 2006.

[6] C. Hett, N. Kuntze, and A. U. Schmidt. Security and Non-Repudiation for Voice-Over-IP Conversations. Poster presentation at the ISSA 2006 From Insight to Foresight Conference, Sandton, South Africa, 5th-7th July 2006, 2006.

[7] International Telecommunication Union. ICTFacts and Figures - The world in 2013, Feb. 2013.

[8] N. Kuntze, C. Rudolph, A. Alva, B. Endicott-Popovsky, J. Christiansen, and T. Kemmerich. On the creation of reliable digital evidence. In G. Peterson and S. Shenoi, editors, *Advances in Digital Forensics VIII*. Springer, 2012.

[9] T. Kunz, S. Okunick, and U. Pordesch. Data Structure for Security Suitabilities of Cryptographic Algorithms (DSSC)-Long-term Archive And Notary Services (LTANS). Technical report, IETF Internet-Draft, 2008.

[10] P. Landrock and T. P. Pedersen. WYSIWYS? - What you see is what you sign? *Inf. Sec. Techn. Report*, 3(2):55–61, 1998.

[11] R. Marx. VoIP authentication and non-repudiation by means of the new German ID card and VoIP Signatures. In *IIT Real-Time Communications*. Illinois Institute of Technology, Oct. 2011.

[12] Ofcom. Communications market report 2012. Technical report, Ofcom, July 2012.

[13] P. Rose. Technical forensic speaker recognition: Evaluation, types and testing of evidence. *Computer Speech & Language*, 20(2):159–191, 2006.

[14] J. Rosenberg et al. Session Initiation Protocol (SIP). IETF Request for Comments: RFC 3261, June 2002.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF Request for Comments: RFC 3550, July 2003.

[16] The European Parliament and the Concil of the European Union. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. *Official Journal of the European Communities*, L 12:12–20, 2000.