Compulsory Service Extensions to SIP-Initiated Communication Sessions

Philipp Marcus, Thomas Mair, and Florian Dorfmeister
Institut für Informatik
Ludwig-Maximilians-Universität München
Munich, Germany
Email: {philipp.marcus, thomas.mair, florian.dorfmeister}@ifi.lmu.de

***Abstract*—The Session Initiation Protocol is an application layer protocol with increasing impact on signaling in mobile networks and mobile applications. However, it lacks the possibility to enforce the abidance of special communication constraints after session parameters have been negotiated. In this work, an approach of compulsory services is presented, which allows peer user agents in communication sessions set up by the Session Initiation Protocol to mutually prove each other that transmitted data has been processed by a trusted third party. This is realized by enforcing a set of compulsory services implemented by trusted third parties. Technically, in the session initiation phase the signaling messages are therefore modified according to a rule base in special policy servers. During data transmission, an involved user agent first passes its message to the compulsory service, receives a corresponding token as acknowledgment and finally transmits this token along with the message to the peer user agent. There, based on the validity of the token, further actions can be taken. The usability is demonstrated by three examples including location verification of mobile users.**

***Keywords*-*SIP-Services*; *Trusted Third Party*; *Public-Key-Cryptography*; *Location Verification*; *Call Recording*.**

I. Introduction

The Session Initiation Protocol (SIP) is an application layer protocol developed by the IETF with focus on creating, modifying and terminating communication sessions with one or more participants [1]. It is designed according to design principles similar to HTTP, and therefore is well suited for IP-based networks. It also allows for extending the protocol with new features that not necessarily must be supported by all clients.

Although SIP is primarily used for Internet telephone calls and video conferencing, due to its flexibility it can also be applied to other domains such as instant messaging. In the context of next generation networks and the IP Multimedia Subsystem (IMS) specified by the 3GGP in TS 23.228, SIP is used as a session control layer to provide a standardized interface to services between mobile users and signaling infrastructure.

The architecture of a typical SIP configuration consists of several entities of which the most important in the context of our work comprise *user agents* (UAs) and the signaling infrastructure elements, i.e., *registrars* and *proxies*. One or more user agents register themselves with a SIP username at a *registrar*, which implements a location service mapping usernames to network addresses. For session initiation, a UA sends an `INVITE` message to the peer user agent. Those messages are typically passed through one or more proxies, which are responsible for routing SIP messages and thereby are able to enforce a call policy.

After the session has been established between the involved user agents, the data transmission phase will follow. Here, none of the signaling infrastructure components has further access to the information exchanged. Hence, although desirable in many application scenarios like the enforcement of mutually agreed on centralized call recording or the mutual provision of reliable information about the residence of UAs, the standard SIP provides no means for enforcing certain properties or the processing of transmitted data.

In order to facilitate the enforcement of specific processing constraints during the data transmission phase, we developed an approach that enables user agents to mutually prove that a given piece of transmitted information has been processed by a set of trusted third party services called *compulsory services* in a predefined manner. The set of involved compulsory services is deduced from policy servers and contained in the SIP-negotiation messages. The services themselves are hosted by external providers.

The remainder of the paper is structured as follows: Section II presents the state of the art in form of related work. Section III then describes the principles of compulsory services comprising the theoretical concepts, the structure of compulsory service assignments in the SIP-negotiation phase, the necessary extensions to SIP and discussion of the data transmission protocol. In Section IV, three example scenarios are presented that profit from compulsory services. Finally, Section V concludes the paper.

II. Related Work

SIP services can be classified into being active during the session initiation or focusing on modifying the data transmission phase. During the session initiation, services can run on signaling infrastructure or on user agent equipment depending on the task the service performs and aim at enforcing parameters for the following data transmission phase according to a set of rules [2]. Device independent services, for example services taking actions when systems are not available or busy, are run on signaling infrastructure, e.g., using SIP-CGI [3]. Services comprising device specific information like location or call waiting alerts are executed on UA equipment. SIP services typically take the following steps:

1) Modification of headers and forwarding of incoming requests based on a set of rules.
2) Receipt of replies to forwarded requests, modifying the replies and returning them to the client.
3) Generating requests to other services by creating and sending appropriate messages.
4) Generation of responses to incoming requests and sending the results to the client.

Approaches to the specification of services that are run on UA equipment have been proposed with SIP Call Processing Language (CPL) [2] or Language For End System Services (LESS) [4]. These approaches might be useful for implementing UAs that dynamically adapt to an incoming set of compulsory services during the session initiation.

A hybrid approach is the concept of SIP Back to Back User Agents (B2BUAs) working as a *man in the middle* in the SIP signaling process and therefore incorporating a UAC and UAS at the same time for the transmission of a given message. A B2BUA keeps track of the state of connections it has established and has full control over all signaling messages [1]. Extensive research has been done in developing domain specific languages for B2BUA programming, enabling applications like seamless device handover [5] [6]. Other approaches like DiaSpec focus on the automatic generation of includable source code from telephony service specifications [7]. In comparison, our approach does not aim at modifying the delivered content but rather extending it with a token certifying that a piece of information in a message has been processed by a trusted third party, namely a compulsory service in a predefined manner.

Services in the data transmission phase have especially been in the focus of research and have been developed in the context of lawful interception (LI) [8]. Focused on the interception of voice transmissions, signaling infrastructure typically modifies `INVITE` and `ACK` messages in the SDP information in order to enforce the redirection of the packet through a special recording gateway [9]. This way, the internet telephony service provider in charge is able to apply logging or packet sniffing tools to the data stream. Contrary to our approach, the employment of the recording services can be concealed from the communicating user agents. Another downside of these approaches is that it is not possible to avoid repudiation of received packages, which in contrast can be implemented by our compulsory services.

III. Compulsory Services and SIP-Sessions

In this section, we present our approach to enforcing the interaction of user agents with compulsory services in SIP initiated communication sessions.

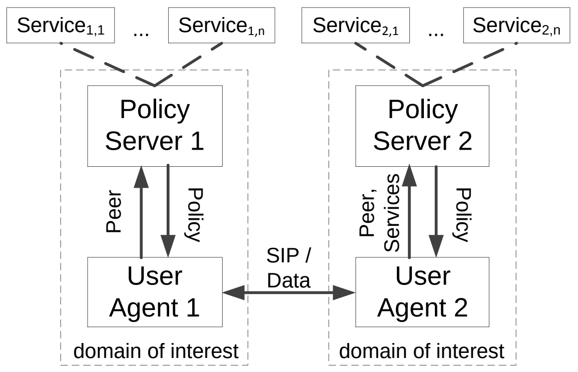

Figure 1. System architecture showing services known by policy servers and two communicating user agents.

A. System Architecture

We propose a system architecture as depicted in Figure 1. A policy server (PS) within an administration domain is associated to $1 \ldots n$ services. Before starting the SIP-negotiation, a user agent queries its PS, which has an integrated set of rules stating which services have been defined as compulsory for the interaction between the two involved user agents. Those services are called *compulsory services* (CS). Conceptually, these services act as trusted third parties, enabling clients to mutually guarantee that data sent to the other user agent has also been sent to and processed by each CS. This is realized by forcing each involved user agent to request *tokens* with a limited time of validity from the assigned compulsory services. These tokens are only issued by the service if the user provides service-specific required data. The tokens are then appended by the UA to data transmissions, which will only be accepted by the receiving user agent if the token is valid. Before initiating a session with a SIP-`INVITE`, a user agent provides its policy server the identities of himself and his peer and receives a set of CSs along with the specification of their communication interface, also including their service access points *SAPs*. The SIP-`INVITE` is then extended by this set and sent to the peer UA, which again contacts his own policy server. The reply sent back to the first UA is extended by the set of CSs that were received from the peer and those received from his own policy server.

As this process runs the risk that two clients might agree on ignoring the tokens and circumvent the enforcement of the compulsory services, our approach is based on the pre-condition that each involved policy server is within the same domain of interest with one of the user agents, i.e., for a given policy server, at least one user agent insists on abiding its claimed compulsory services.

More precisely, a *compulsory service* i was defined as a tuple: $CS_i = (SAP, X, f, D, Cert, g)$ with the following semantics: SAP is the service access point of the CS represented by an URI. The set X represents the domain of data CS_i is interested in to finally create a token. However, in general this data is not passed to CS_i by an user agent genuinely, but is rather pre-processed by the function $f: X \mapsto D$. For example, D can be defined as the set of all hash values of elements in X. The *Cert* element represents a public-key certificate associating a public key $K^+_{CS_i}$ with the *SAP* and also states that CS_i owns the private key $K^-_{CS_i}$. The set of all encrypted elements of D is denoted as $\mathcal{C}(D) = \{x | \exists y \in D : K^-_{CS_i}(y) = x\}$.

The last element g is statically defined as a constant $g : \mapsto \{\texttt{once}, \texttt{periodicTime}(t), \texttt{periodicPacket}(t), \texttt{always}\}$, formalizing when tokens need to be generated for a data packet. $\texttt{periodicTime}(t)$ states that at least every t seconds a valid token has to be used. Similarly, $\texttt{periodicPacket}(t)$ implies that at least every t packets a new token has to be provided. In Section III-C the necessary extensions to SIP are discussed.

A compulsory service CS_i provides an external interface to user agents with a function $CS_i : D \mapsto \mathcal{C}(D) \cup \{\perp\}$, which generates the desired tokens. The process of token generation within CS_i for a request $CS_i(f(x))$ is based on two steps: first, the argument $f(x)$ is sent to a processing function $p : D \mapsto \{\texttt{true}, \texttt{false}\}$, e.g., for logging $f(x)$ to a database or consulting third parties to verify information. Its result is a boolean value. If this value is `true`, the token is generated by computing the digital signature of $f(x)$ using $K^-_{CS_i}$ and returned to the calling user agent. Otherwise, $\perp$ will be the result, symbolizing invalid tokens:

$$CS_i(f(x)) = \begin{cases} K^-_{CS_i}(f(x)), & \text{if } p(f(x)) = \texttt{true} \\ \perp, & \text{otherwise} \end{cases} \quad (1)$$

Finally, the calling user agent transmits all tokens $1, \ldots, n$ alongside the message x to his peer where the validity of each token is checked:

$$K^-_{CS_i}(f(x)) \text{ is valid} \Leftrightarrow K^+_{CS_i}(K^-_{CS_i}(f(x))) = f(x) \quad (2)$$

The formalism describing which services are defined as compulsory for a communication session is discussed in the next section.

B. Assigning Compulsory Services to SIP-Sessions

In this section, the process of rule selection at a policy server in the domain of interest of one of the two involved user agents is presented. Therefore, sets *Users*, *Roles* and *Services* have been defined. The schema is illustrated in Figure 2.

The set *Users* contains all known user agents and *Roles* all role identifiers. Users can be assigned to roles given by an user-role-relation $UR \subseteq Users \times Roles$. The set of all currently assigned roles $r \subseteq Roles$ for a user $u \in Users$ is denoted as $r(u)$. Furthermore, the set of all compulsory services has been defined as *Services* and its power set as $\mathcal{P}(Services)$. The time-dependent role assignment to users at a certain point of time is represented by the set $Sessions \subseteq Users \times Roles$. The actual rule base is defined using the relations $RS \subseteq (Roles \times Roles) \times \mathcal{P}(Services)$ and similarly $\overline{RS}$. RS states, which compulsory services have to be enforced in the communication between two user agents with specific roles, while $\overline{RS}$ contains forbidden ones.

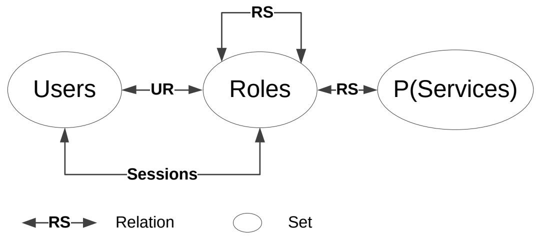

Figure 2. Rule base representing user-role-service-assignments.

In complex scenarios, each involved user agent, ua_1 and ua_2, has its own policy server PS_1 or PS_2 within its domain of interest. In all cases, ua_1 generates a SIP-`INVITE` according to the rules imposed by PS_1. For this purpose PS_1 provides a set of compulsory services S_{ua_1} that will be integrated into the handshake message. Additionally, ua_2 has the policy server PS_2 within its domain of interest. The latter will will be provided with S_{ua_1} as well as ua_1 and will return a set S_{ua_2} of desired compulsory services and a boolean value indicating if all services in S_{ua_1} are acceptable. In accordance to the extensions made to the original SIP-`INVITE` by ua_1, ua_2 will extend the reply with his set of compulsory services S_{ua_2} induced by the rule base of his own policy server PS_2, too.

A policy server PS_i within the domain of interest of a user agent ua_i computes the set of compulsory services S_{ua_i} according to its rule base by evaluating:

$$\begin{aligned} S_{ua_i} = \{cs \in Services | \exists S \in \mathcal{P}(Services) \\ . \exists r_1, r_2 \in Roles : r_1 \in r(ua_1) \\ \wedge r_2 \in r(ua_2) \wedge (r_1, r_2, S) \in RS \wedge cs \in S\} \end{aligned} \quad (3)$$

The elements in the set of services S_{ua_k} received in a handshake message from a peer are tested for policy conflicts using $\overline{RS}$. In case of conflicting policies, the session initiation is canceled.

Eventually, each involved user agent knows the union set of all enforced compulsory services $S_{ua_1,ua_2} = S_{ua_1} \cup S_{ua_2}$, which is fixed for the duration of the session.

The next section describes how SIP-messages are extended in order to allow for the transmission of S_{ua_1,ua_2} to the user agents.

C. Integration into SIP

In order to agree on a set of compulsory services between two user agents, the SIP messages at the initiation of a session need to carry information about these services. This guarantees that the user agents are able to reject the session establishment in case one of the user agents does not accept the compulsory services required by the other party.

The inclusion of additional information into a SIP-message may be achieved by different means. Possibilities are to define an additional header field, to send the information within the body of the message or to use multipart bodies when the SIP message already contains a body part (e.g., an SDP body) [10]. Making use of the message body to include the additional information is reasonable if the additional information is extensive and contains structured information. If a user agent encounters an unknown header field, the header will simply be ignored. This solution has the advantage that a user agent that is not able to process new header fields may be detected to be incompatible. If a user agent is not able to decode multipart messages it will issue a `415 Unsupported Media Type` response to signal its inability to process the message. In this work, to add the information, adding the required compulsory services to the body of the SIP-messages has been chosen. This requires only little modification of established user agents and allows for great flexibility when implementing a suitable container format for the description of the compulsory services. The definition of the container format is omitted here, but it will include all the information contained in the definition of a compulsory service as stated in Section III-A.

Each user agent follows the same steps when a session is established as depicted in Figure 3. The calling user agent ua_1 requests the set of compulsory services S_{ua_1} from its policy server PS_1 by providing it with the identity of the desired peer. It then includes the received services S_{ua_1} in the SIP-`INVITE` message, which will be forwarded to the called user agent ua_2. The called user agent first examines if the certificates of the compulsory services are valid and then checks if it is able to provide the necessary information for all services. If one of these checks fails, the `INVITE` request will be rejected by generating a `406 Not Acceptable` response. If the services pass this validation, user agent ua_2 – analogous to ua_1 – queries PS_2 for the required compulsory services for a session with ua_1 and whether the services contained in S_{ua_1} are acceptable. If so, it includes S_{ua_2} into the `200 OK` response or sends a `406 Not Acceptable` otherwise. User agent ua_1 will perform a validation of the services required by ua_2, too. If the validation fails, the session will be terminated by an immediate `BYE` request. Otherwise the session will continue with the data transmission phase as described in Section III-D.

SIP is also able to modify the parameters of an active session, which is achieved by the reinvite mechanism of the

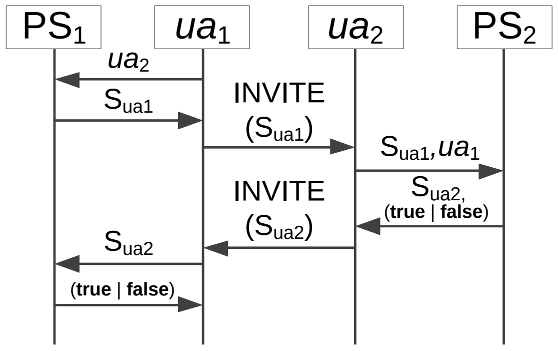

Figure 3. Communication during the the SIP session establishment.

protocol. This will be used when a compulsory service is no longer reachable or other problems with the services occur. The re-invitation will renegotiate all the compulsory services for each user agent and terminate the session in the same manner as an initial `INVITE` in case the services do not pass the verification steps at each user agent. This gives the services' administrators the ability to change or remove a service without interrupting the data transmission phase and without termination of the session. If the re-invitation was not successful, i.e., one of the compulsory services is not reachable or the user agent is not able to provide all the necessary information for one of the services, the session is terminated by issuing a `BYE` request.

D. Data Transmission Protocol

With the extended session initiation being completed, user agents ua_1 and ua_2 can start their communication as shown in Figure 4. Before actually sending a message x to its peer, however, each user agent has to contact all services contained in S_{ua_1,ua_2} first in order to have its message's content processed and signed by the respective compulsory services. Consequently, the unavailability of any CS leads to the abortion of a session. Since different services might require different representations of x, the requesting user agent ua_1 will locally perform a service-specific pre-processing function f_{CS_i} and provide CS_i with $f_{CS_i}(x)$. Depending on whether a service's processing function p_i works as an idempotent call or not, making use of a reliable communication protocol (e.g., TCP) between user agents and CS_i is required. This is in order to prevent inconsistencies caused by the repeated processing of a message $f_{CS_i}(x)$ due to a lossy channel. CS_i replies to requests with a token $K^-_{CS_i}(f(x))$ that ua_1 will then forward to ua_2 alongside the other services' tokens and the original message x.

We propose two alternative solutions for this kind of extended data transmission: The first one is to define a minimal container format bundling the set of required to-

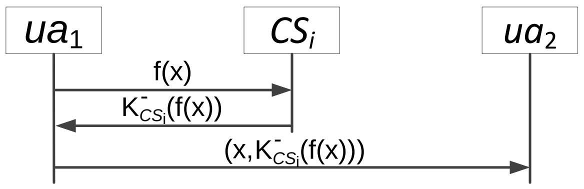

Figure 4. Compulsory service integration in the data transmission phase.

kens $K^-_{CS_i}(f(x))$ and the original message x into a single packet. This tight coupling of a message's content and its associated tokens within one packet is necessary in order to avoid any inconsistencies that might otherwise arise from the loss of packages containing messages or tokens when using unreliable communication channels. Another possible solution to this problem is to embed this kind of compulsory service related information into existing protocols. In case of RTP this could for example be accomplished by assigning a new dynamic payload type `PT` in the RTP header [11] [12]. This allows to avoid defining a wholly new message format by making a minor extension to a standardized protocol a multitude of clients is already speaking. Hence, making use of the latter approach is recommended whenever it seems feasible. Either way, just as desired, only those clients that can either handle such a specific container format or can cope with an existing protocol's extension will be able to correctly process and validate incoming packets as well as give adequate responses.

Upon receiving a message from ua_1 user agent ua_2 will decompose the received packet and then try to validate the contained tokens in order to locally prove that service CS_i has processed and approved the content of x. If any of the tokens required by S_{ua_1,ua_2} are found to be absent or illegal, the discovering user agent will terminate the current session or at least reject the fraudulent message(s).

Dependent on the service-specific value of g_{CS_i}, both the sending and the receiving user agent know when to request and accordingly expect a fresh token from service CS_i. The constant value of g thereby indirectly synchronizes the two user agents' expectations of when an updated token signed by CS is required. Each time one of the agreed on compulsory services demands the usage of a new token, the sending user agent has to repeat the procedure described at the beginning of this section in order to acquire a fresh token. Otherwise the peer client will reject incoming packages until a valid token arrives or even quit the session.

As already stated before, it has been defined as a pre-condition that each user agent ua_i at least abides by the rules imposed by the policy server PS_i that it is assigned to. This is required since the policy servers have no means for supervising the correct usage of compulsory services once the peer-to-peer communication session between the two user agents has been established. Hence, full abidance of the whole set of negotiated services has been achieved by having ua_1 and ua_2 controlling the rules introduced by PS_1 and PS_2 respectively.

IV. Example Scenarios

In this section, three example scenarios and use cases are presented that can be realized using compulsory services.

Location Verification: Especially for mobile user agents the current location of the peer user agent could be of high interest but may be tampered with. Location Verification is the process of proving positions claimed by users [13]. In the context of SIP-sessions this can be realized by implementing a location prover as a compulsory service. In general, a location prover is able to verify the claimed location of a given user agent or entity, e.g., by employing trusted infrastructure like ultrasound sensors in rooms. Assume two communicating user agents that intend to exchange information x along with a verified position l, describing the residence of the sending UA. This results in a message structure $m = (x, l)$. Furthermore, let CS be a location verification compulsory service that has been assigned to the session. Given the message $m = (x, l)$, the pre-processing function f has to be defined as: $f(m) = l$, i.e., only the location l will be transmitted to CS. There, the processing function p will return `true` if the claimed location could be verified and `false` otherwise.

The correctness of l is checked and confirmed with a token $K^-_{CS}(l)$ or with $\perp$ if l does not correspond to the residence of ua_i. According to the delay that is caused by CS when contacting the location proving infrastructure, CS is defined with an appropriate periodicity g. Finally, after checking the correctness of the received token, the peer user agent applies the same processing steps when sending messages to ua_i.

Non-Repudiation Receive: In order to prevent two communicating UAs ua_1 and ua_2 from repudiating the receipt of a message m, a non-repudiation compulsory service CS is assigned to the session. The service has a pre-processing function $f(m, K^-_{ua_i}) = K^-_{ua_i}(hash(x))$, which is used to digitally sign the hash value of a previously received message m. The processing function p is then implemented as the logging of $K^-_{CS}(hash(m))$ to an attached database. This way, CS is able to generate a log of all messages that have been received by the user agents ua_1 and ua_2. The generated token $K^-_{CS}(K^-_{ua_i}(hash(m)))$ is finally sent to the peer user agent by piggy-backing it on a message x in order to acknowledge the receipt of m, thereby preventing the UA from repudiating it later.

Interception and Recording: The assignment of compulsory services to SIP-Sessions can also be used for complete logging of communication sessions on behalf and on the agreement of the user agents ua_1 and ua_2. In many countries, mutual agreement on the recording of calls is claimed by law. The interception or recording of all pieces of transmitted information is generally more suitable for non real time applications, e.g., instant messaging based on SIP. The periodic requests for attached compulsory services $S_{ua_1,ua_2} = \{CS_1, \ldots, CS_n\}$ introduce a delay d given by the maximum round trip time (RTT) between user agents and compulsory services:

$$d = max\left(\{t | \exists s \in S_{ua_1,ua_2} : t = RTT(s, ua_1) \vee t = RTT(s, ua_2)\}\right) \quad (4)$$

An appropriate compulsory service CS would define a pre-processing function f corresponding to the identity function, i.e., $f(x) = x$. The processing function p here returns `true` if the received data x could be recorded. This is finally confirmed by a ticket $K^{-}_{CS}(x)$, which is sent along with x to the receiving user agent. There, if the token is found to be invalid, stating that x has not been correctly recorded, the processing of x can be stopped.

V. Conclusion and Future Work

We presented an approach that enables SIP user agents to mutually force each other after session initiation to prove that sent data has been additionally processed by a trusted third party in advance, which has been denoted as a compulsory service. During session initiation using an extension of SIP the compulsory services are negotiated, each pre-defined by a pre-processing function and a processing function. A user agent aiming for sending information to its peer first transmits the pre-processed information to all compulsory services and receives individual tokens certifying this interaction. These tokens are finally passed to the peer user agent alongside the original information. During the initiation of a session, the involved user agents extend signaling messages according to a set of compulsory services that are in each case deduced from an attached policy server. This policy server is defined within the same domain of interest of the user agent. We outlined how SIP signaling messages can be extended with a set of compulsory services and discussed aspects of communication between user agents and compulsory services as well as the interaction between two user agents. Finally three applications of compulsory services in the fields of location verification, avoidance of repudiation of message receipt and agreement upon centralized call recording have been presented.

Future work has to concentrate on developing a description language for compulsory services, on optimizing the piggybacking mechanisms for user tokens and evaluating aspects of efficiency.

References

[1] 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), Internet Engineering Task Force, Jun. 2002, http://www.ietf.org/rfc/rfc3261.txt (retrieved: April, 2012).

[2] J. Lennox and H. Schulzrinne, “Call Processing Language Framework and Requirements,” RFC 2824 (Informational), Internet Engineering Task Force, May 2000, http://www.ietf.org/rfc/rfc2824.txt (retrieved: April, 2012).

[3] J. Lennox, H. Schulzrinne, and J. Rosenberg, “Common Gateway Interface for SIP,” RFC 3050 (Informational), Internet Engineering Task Force, Jan. 2001, http://www.ietf.org/rfc/rfc3050.txt (retrieved: April, 2012).

[4] X. Wu and H. Schulzrinne, “Programmable End System Services Using SIP,” in *IEEE International Conference on Communications, 2003*, ser. ICC’03, vol. 2. IEEE, 2003, pp. 789–793.

[5] P. Zave, E. Cheung, G. Bond, and T. Smith, “Abstractions for Programming SIP Back-to-Back User Agents,” in *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, ser. IPTComm ’09, ACM. ACM, 2009, pp. 11:1–11:12.

[6] V. Hilt and M. Hofmann, “Approaches to Implementing Services in SIP Networks,” *Bell Labs Technical Journal*, vol. 9, no. 3, pp. 39–44, 2004.

[7] W. Jouve, N. Palix, C. Consel, and P. Kadionik, “A SIP-Based Programming Framework for Advanced Telephony Applications,” in *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5310, pp. 1–20.

[8] W. Chen, C. Gan, and Y. Lin, “NTP VoIP Platform: A SIP VoIP Platform and Its Services,” in *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, ser. ACOS’06, World Scientific and Engineering Academy and Society (WSEAS). World Scientific and Engineering Academy and Society (WSEAS), 2006, pp. 756–761.

[9] M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol,” RFC 4566 (Proposed Standard), Internet Engineering Task Force, Jul. 2006, accessed 12-April-2012. [Online]. Available: http://www.ietf.org/rfc/rfc4566.txt(retrieved:April,2012)

[10] G. Camarillo, “Message Body Handling in the Session Initiation Protocol (SIP),” RFC 5621 (Proposed Standard), Internet Engineering Task Force, Sep. 2009, http://www.ietf.org/rfc/rfc5621.txt (retrieved: April, 2012).

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550 (Standard), Internet Engineering Task Force, Jul. 2003, http://www.ietf.org/rfc/rfc3550.txt (retrieved: April, 2012).

[12] H. Schulzrinne and S. Casner, “RTP Profile for Audio and Video Conferences with Minimal Control,” RFC 3551 (Standard), Internet Engineering Task Force, Jul. 2003, http://www.ietf.org/rfc/rfc3551.txt (retrieved: April, 2012).

[13] N. Sastry, U. Shankar, and D. Wagner, “Secure verification of location claims,” in *Proceedings of the 2nd ACM Workshop on Wireless Security*, ser. WiSe ’03. ACM, 2003, pp. 1–10.