Flexible Single Sign-On for SIP: Bridging the Identity Chasm

Pin Nie, Juha-Matti Tapio, Sasu Tarkoma, Jani Heikkinen
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
niepin@cc.hut.fi, jmtapio@verkkotelakka.net
sasu.tarkoma@tml.hut.fi, jani.heikkinen@tml.hut.fi

***Abstract*—Identity federation is a key requirement for today's distributed services. This technology allows managed sharing of users' identity information between identity providers (IDP), and subsequently, the use of federated identities to access service providers (SP). Single Sign-On (SSO) is a core feature provided by these systems.**

The Session Initiation Protocol (SIP) is a signaling framework for session call control. It is becoming a widely accepted layer for applications and services, especially in the telecommunications and multimedia domain.

In this paper, we explore solutions to incorporate SSO process into the SIP framework in order to simplify the services and resources access. Our design leverages the Liberty Alliance specifications and extends the existing SIP standards to support SSO functionality. We also present a prototype implementation at the end of this paper.

Keywords: Identity Federation, Single Sign-On, SIP, SAML, authentication

I. Introduction

In recent years, the emergence of distributed services has given momentum for simplifying client authentication and authorization. Ideally, clients would be required to prove their identities only once and free to access subsequent services without authentication anymore. This feature is well known as single sign-on (SSO) which enables users to access multiple services and resources with one-time login. Essentially, it is a collaboration form enabling access control over multiple software systems.

Single sign-on systems require a special trusted entity, namely the identity provider (IDP), which coordinates and manages the underlying authentication session. One big benefit it brings is a simple form of bootstrapping authentication over multiple sessions for users. With SSO, a user doesn't need to memorize many accounts' information when visiting different services. Instead, the IDP can handle service establishment on behalf of the end user. Consequently, less configuration is required at the user end, security communication efficiency is improved, and seamless service sequence is possible. Up to date, many SSO solutions have been proposed on HTTP-based platforms [1] [2].

SIP (Session Initiation Protocol) [3] [4] is an application-layer signaling protocol for creating, modifying and terminating sessions with one or more participants. The first goal for SIP was to provide a signaling and call setup protocol for IP-based communications that can support a superset of the call processing functions and features presented in the Public Switched Telephone Network (PSTN). Later, additional enhancements and extensions were proposed to build more features and functions on SIP networks, such as SIMPLE (Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions) [5]. So far, SIP protocols family can support a variety of communication services, like VoIP (Voice over IP), SIP conferencing and Instant Messaging (IM).

As the number of SIP-based applications and services grows rapidly, they face the same dilemma that has been solved in SSO for HTTP-based web services, namely complicated authentications. SSO is also preferred especially for mobile services due to the input limits. Meanwhile, SIP is leading a big trend in mobile service creation in IP Multimedia Subsystem (IMS). Consequently, the existing SIP architecture has to be modified to enable SSO. At the same time, the design should consider compliance with existing standards while imposing minimum cost.

In this paper, we build SSO capabilities to the SIP framework on the basis of the Liberty Alliance specifications [6]. We provide extensions for both SIP HTTP mixed situation, e.g. Internet, and SIP native environment. We also consider using terminal-based credentials to bootstrap SSO from device startup.

The rest of the paper is organized as follows: In Section II architecture design and extensions are described. A prototype and evaluation are provided in Section III. In Section IV, we discuss future work to improve our design for better performance. Finally, Section V gives the conclusions.

II. Architectures

Among different Single Sign-On solutions, we chose Liberty Alliance project as the basis of our SIP SSO design. Liberty Alliance is one of the most popular industrial open standards specifying identity management issues. Single Sign-On process is defined in Liberty Identity Federation Framework, known as ID-FF [7], [8], [9]. ID-FF takes the Security Assertion Markup Language (SAML) [10] as the protocol reference, which is an XML standard for exchanging authentication and authorization data between security domains. SAML is also adopted in some other identity federation

projects besides Liberty Alliance, such as Shibboleth [11] and WS-Federation.

We consider two different application scenarios. The first scenario happens in a mixed environment in which both HTTP and SIP services are involved, for example, the Internet. It is the primary scenario where users often operate on multiple services or systems at the same time. The other case takes place when only SIP is applied, for example, the IP Multimedia Subsystem (IMS) [12]. In the following sections, we first examine the mixed environment and provide an alternative design for the SIP native environment.

A. Pipeline Architecture

In the mixed environment with HTTP and SIP, a unified and seamless authentication process is required to combine HTTP-based services and SIP-based services. ID-FF already provides support for HTTP-based SSO. To leverage this existing mechanism and integrate to the standard SIP framework, SIP user agent and proxy are extended to do authentication based on the Liberty Alliance framework. The design is named **Pipeline** for the fact SIP elements concatenate two different communication protocols.

Figure 1 illustrates the message flow during SIP registration in the pipeline approach. A SIP proxy generates response UNAUTHORIZED 401 with its authentication request (AuthnRequest) enclosed when receiving an unauthenticated REGISTER. The user agent (UA) extracts the authentication request from the response and forwards it to the Identity Provider (IDP). Based on a predefined agreement with the service provider (SP), IDP will issue a corresponding assertion for the authentication request and return the artifact (like a pointer) of the assertion to the user agent. By encapsulating the artifact, user agent resends REGISTER to the proxy, which verifies the artifact with IDP for the actual assertion. Note that the single sign-on and the verification stages (dashed lines) are carried out in HTTP protocol. If the verification passes, the proxy returns the response OK 200 and a shared secret will be exchanged between the caller and SIP proxy for subsequent sessions. The verification is defined in the Liberty Single Sign-On and Federation Protocol, with respect to the ID-FF Liberty Artifact Profile. The REGISTER can be replaced by any other SIP request, like INVITE and SUBSCRIBE. The authentication requiring response could be either UNAUTHORIZED 401 and PROXY_AUTHENTICATION_REQUIRED 407, depending on SIP elements, namely registrar or proxy.

Shown in the message sequence diagram above, new logic and process are added. In order to indicate single sign-on capability and convey SAML contents, such as AuthnRequest and artifact, we design a new protocol feature following the extension service instruction in SIP specification [3].

At first, we propose a new option tag in the Proxy-Require header to designate liberty Single Sign-On authentication. It is *"liberty-id-ff"*. "liberty" implies the liberty compliant implementation, and "id-ff" implies the ID-FF compliant Single Sign-On process, which expects artifact header in the SIP request and AuthnRequest header in the SIP response.

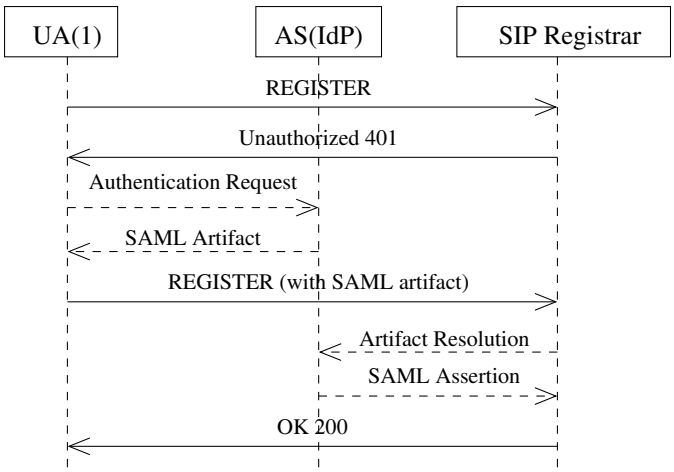

Fig. 1. Pipeline SSO on SIP REGISTER, solid lines are SIP messages and dashed lines stand for HTTP communications.

Second, two new SIP headers are proposed for SSO. They are AuthnRequest header and artifact header. AuthnRequest header contains the SP's authentication request and is added in the SIP response 401 or 407. Artifact header contains the artifact and is added in the SIP requests. It is removed after the artifact resolution once the verification is done.

As a result of HTTP communications, an interface must be built in SIP UA and proxy. On the UA side, it enables UA to extract artifact from HTTP message and encapsulate into SIP request. On the proxy side, it connects SIP registar/proxy with IDP, fulfilling artifact resolution based on the ID-FF specification.

Since Proxy-Require header is always checked at the beginning (during the request validation), our single sign-on procedure would be processed between SIP elements if the new headers are supported and interfaces are implemented. Otherwise, the SIP request would be abandoned and an error response 420 (Bad Extension) is replied. As a result, a client must switch to another authentication method, e.g. SIP digest. Therefore, our design also provides backward compatibility for existing authentication methods in case our SIP SSO is not supported by the proxy.

At the end of SSO, a shared secret should be agreed between SIP UA and proxy to provide security provisioning for subsequent messages. Any symmetric security protocol can be employed here. For example, the response 200 OK from the SIP proxy initiates a shared secret to SIP UA. By holding this secret, subsequent SIP calls from the UA can be accepted by the proxy bypassing the authentication. To provide stronger security association, a more complex computation is recommended to bind with secret. Usage of a hash function with a timestamp and a nouce is a popular example. Furthermore, expiration settings are required as well. As a result, session hijack, replay attack, and eavesdropping are prevented.

There are two benefits in Pipeline architecture. First, there

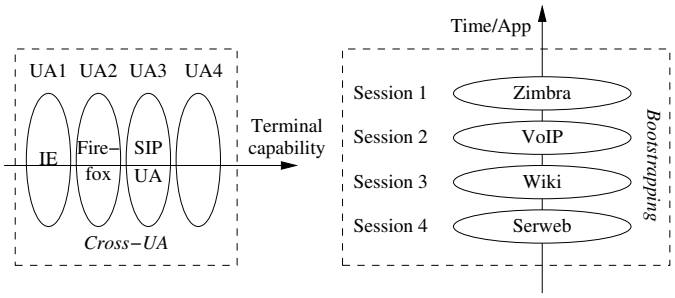

Fig. 2. Identity Agent provides a common interface and Single Sign-On provides bootstrapping authentication

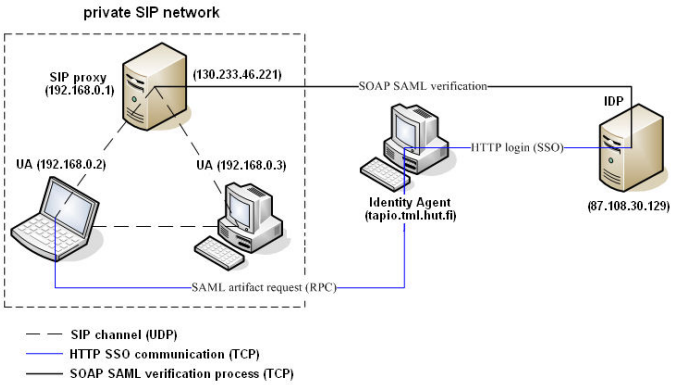

Fig. 3. Organization of the SIP Single Sign-On testbed

is no need to modify existing IDP servers. The IDP server keeps all HTTP features and logic modules intact. Second, SIP UA and other HTTP-based agents (e.g. web browser) in the terminal can apply a unified authentication service with a common interface. The user profile is shared and the workload on the user is minimized. In our implementation of the common interface, we propose **Identity Agent** to communicate with the IDP on behalf of SIP elements. It bootstraps the single sign-on for the terminal, and is responsible for providing authentication service to the browser, SIP UA and other client applications. Figure 2 illustrates the idea of this interface.

B. SIP Native Architecture

In contrast to the Pipeline architecture for HTTP and SIP mixed environment, **SIP Native** architecture considers SSO only in the SIP domain. Specifically, both SAML assertion request and artifact resolution stages are executed in SIP protocol. In this model, one primary task is to carry SAML protocols with SIP framework. An IETF draft on SIP SAML profile and binding [13] has been proposed. However, it relies on HTTP URI in the SAML HTTP-URI-based SIP Binding. In our assumption of a pure SIP world, it does not work.

We propose to use SAML SIP-URI-based Binding (SSUB) and SAML SOAP Binding over SIP (SSSB) to solve the problem. SSUB binding extends the ”SAML URI Binding” specified in the SAML binding standard [14]. The general form of SIP URI follows [3]:

sip:user:password@host:port;uri-parameters?headers

Assume that an IDP’s SIP address is ”sip:idp@tml.tkk.fi”, SAML assertion URI uses the parameter ”assertion”, the assertion is issued for the caller ”niepin”, a request for the assertion can be sent to following SIP address:

sip:idp@tml.tkk.fi;transport=tcp&assertion=id?
from=niepin%40tml.tkk.fi

SSSB binding keeps SAML SOAP binding [14] and uses SIP to transport SOAP messages. An IETF draft [15] proposed a new SIP request method, named **SERVICE** to carry SOAP message as its payload. Since SAML negotiation may contain a lot of data that often results in large messages to transport, connection-oriented protocol like TCP should be used to transmit. The parameter is specified in the field ”transport”. More parameters can be appended.

However, in SIP native architecture, IDP must provide authentication service on the top of SIP framework. Actually, Identity Provider, a.k.a Authentication Server, is a special SIP proxy in our model. It provides authentication service to UAs and is trusted by other proxies. The underlying single sign-on function and ID-FF profile remains the same, but with a different protocol binding.

III. Evaluation

We implemented the SIP Single Sign-On based on the Pipeline architecture to demonstrate the benefits of our design. A general test was conducted to evaluate the feasibility.

A. Testing Platform

In order to simulate the SIP SSO with Pipeline architecture, we used four machines, two SIP user agents, one SIP Proxy and one IDP server. For clear evaluation of the interface, we separated Identity Agent to a different machine. Figure 3 shows the deployment of our testbed.

Hardware and software of each elements in our testbed are listed below:

- UA-1 (192.168.0.2): AMD Athlon(tm) 64 X2 Dual, Core Processor 3800+ 2.0GHz, 1GB RAM; MS Windows XP Professional, Version 2002 Service Pack 2
- UA-2 (192.168.0.3): Intel(R) Pentium(R), D CPU 2.8GHz, 1GB RAM; MS Windows XP Professional, Version 2002 Service Pack 2
- SIP proxy (192.168.0.1): Intel(R) Pentium(R), D CPU 2.8GHz, 1GB RAM; Linux 2.6.16-1-686-smp SMP i686 GNU/Linux
- IDP server (87.108.30.129): Intel(R) Xeon(TM) CPU 3.00GHz, 3GB RAM
- Identity Agent (tapio.tml.hut.fi): Mac G4 1.3GHz, Linux Debian

	Unsigned	Signed
avg ops/min	98 ops	74 ops
Speed	1.68 ops/sec	1.23 ops/sec

TABLE I
SEQUENTIAL TEST OF SIP SINGLE SIGN-ON

B. Measurements

Our first step measurement derives from two research questions:

1) *Is it possible to extend Liberty single sign-on solution to the SIP framework?*
2) *What is the baseline performance of SIP single sign-on?*

To answer the first question, the real experience of seamless access with both SIP and HTTP applications appeals to us. We integrated multiple services in our single sign-on implementation, including web applications and SIP proxy. Web services include Serweb, Mediawiki, Zimbra Collaboration Suite and NEON Indoor location. SIP proxy is a modified copy of JAIN-SIP-PRESENCE-PROXY, which can provide instant messaging, presence and voice call services. According to our test, the user firstly logs into the IDP, then can browse integrated web services and enjoy call services without any more authentication. The first time to visit a service is a bit slow, because the web application server (or SIP proxy) needs to verify user's identity from the IDP and negotiates a shared secret with the client. After that, subsequent accesses are smooth, like usual. Related resources and documentation are available on our website [16].

Concerning the second question, we make use of existing SIP applications from two open source projects, NIST sip proxy [17] and SIP Communicator [18]. Both are developed in Java language, providing ordinary performance. We used 4096 bits long key for cryptography in signing and ssl (HTTPS connection). Signing is a recommended process on AuthnRequest at the SIP proxy, a.k.a digital signature. It can protect the IDP from DoS attack. In the sequential test, table I shows the average performance of SIP SSO from the user's perspective. The operation is the session establishment and iterates for 60 seconds. The test was repeated 10 times to provide enough samples to minimize possible errors.

In order to get the peak performance, we separate request sending process and artifact fetching process. INVITE sending speed is controlled in milliseconds. A new thread of artifact fetching process is created every time when a new AuthnRequest is received. All threads work in parallel and do not wait the completion of others.

The concurrent test executed 60 seconds and was repeated under different INVITE sending speeds to find the endurance variation and maximum threshold of our SIP SSO system. It ran separately in signed and unsigned AuthnRequest cases to find out the overhead imposed by the signing process. In unsigned tests, we increased the INVITE sending speed by reducing the interval from 500 milliseconds to 100 milliseconds, while in signed tests it is from 600 milliseconds to 200 milliseconds, due to the extra computing overhead of the signing process. By recording the timestamp of INVITE request, 407 response with AuthnRequest, re-INVITE with artifact and the response OK 200, the speed of each phase during SSO can be roughly measured. The results are shown in Fig. 4.

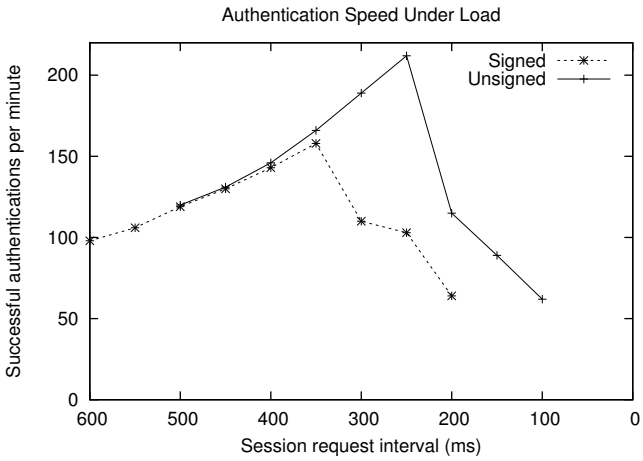

Fig. 4. SIP Single Sign-On stress loading performance

Figure 4 shows that the signed and unsigned curves merge at the beginning when the INVITE sending interval is big enough. Hence, the SIP proxy is capable to consume all incoming requests and complete sessions. When the speed grows, two curves reach their summits at the values slightly over 150 (Signed) and 200 (Unsigned), and drop down sharply. We believe the drop is caused by the code flaws when overloaded, such as faulty buffering policy or memory leaking.

C. Results Analysis

Based on the measurements above, we have the answers to our questions. Liberty single sign-on solution is feasible in SIP framework with our modified design. It does simplify the user authentication process and provide the same security level, which is guaranteed by Liberty alliance specifications.

Our SIP SSO design provides reasonable performance. It can achieve much better benchmarks with some improvements. First, if the key length is shortened from 4kb to 2kb or 1kb, the speed would increase a lot. It is because the cryptography in signing and ssl requires intensive CPU computing and consumes much time. In our script test, 2kb long key offers as seven times speed as 4kb length on the same machine. In practice, 1kb key is applied most and breakable at the current state of arts. 2kb is recommended.

Second, if our SIP SSO is used in SIP registration phase instead of session establishment, The load can be greatly reduced because all subsequent sessions benefit from the registration at the beginning. Usually, one registration covers many sessions before expiration.

Third, it is better to integrate the single sign-on function directly into the SIP stack, other than the application logic. It would provide much faster response.

IV. Discussion

The most intuitive advantage of our Single Sign-On solution is the release of "passwords fatigue" on the user, when operating on multiple services. In our trial, we first log into the IDP and access all ID-FF enabled services including making SIP call without any further authentication. We believe this happy experience would motivate people to try more services in a consecutive switching. While unit-by-unit login in the old fashion will be obsolete in the future.

To further simplify the authentication process in SIP domain, we plan to extend our single sign-on design in two aspects. First, it can be expanded by sharing the account on a group of devices. By taking advantage of the Contact header list in SIP, the initiator could append all trusted devices' SIP addresses into the list. The initiator is treated as the owner of the account and all other grouped device are considered as guests. The initiator can specify the expiration for each guest in the corresponding contact header. The authorization policy of the guest is also configurable in the preferences of the resource.

Second, we believe the ultimate form of SSO is to integrate with local account on the operating system. Once the user logs in to the system, authentication is completed for SSO enabled services. In mobile device, it is considered to bind with the credential in SIM card. As a result, Single Sign-On will be a bootstrapping service on the user end. GBA/GAA (Generic Bootstrapping Architecture and Generic Authentication Architecture) [19], accepted in 3GPP (3rd Generation Partnership Project), is a good example. It also provides a interface [20] to cooperate with Liberty Alliance Project (LAP). Consequently, our SIP SSO design may achieve wider application.

Meanwhile, we also need to keep up with other related work. For example, SIP-SAML Profile and Binding [13], the first IETF draft introducing SAML assertion concepts to SIP networks, is a big help and important reference to our design. Since SAML v2.0 is coming up, our design based on SAML v1.1 has to be updated.

V. Conclusions

This paper explored the solutions to introduce Single Sign-On to the SIP domain with the liberty ID-FF specifications as the starting point. In addition to the architectural design, two new SIP SAML bindings were introduced. Tests and measurements were carried out to show the feasibility of the proposed architecture. Our demo shows the significant improvement on users' impression of services access in the presence of SSO system. In other words, the efficiency of the services is enhanced. Applications at the upper layer are becoming more friendly and easier to operate. In addition to JAIN SIP Proxy, we have also implemented SSO extension for OpenSER SIP proxy. Future work would include evalution of this extension, investigation of native SIP SSO functionality and combination with local account in operating system.

VI. Acknowledgements

We would like to thank Pekka Laitinen, Prof. Antti Yla-Jaaski, Prof. Jukka Manner, Annu Myllyniemi, Hannu Flinck, Hannes Tschofenig and Gang Lian for their valuable comments and suggestions on the paper. The research is funded by NOKIA NEON programme.

References

[1] Henri Mikkonen and Mika Silander, "Federated Identity Management for Grids," in *International conference on Networking and Services, 2006. ICNS '06*. Helsinki, Finland: IEEE computer society, July 2006, pp. 69–69.

[2] A. Myllyniemi, "Identity Management Systems: A Comparison of Current Solutions," At http://www.tml.tkk.fi/Publications/C/22/papers/Myllyniemi_final.pdf.

[3] J. Rosenberg, H. Schulzrinne, G. Camarillo, J. Peterson, A. Johnston, and E. Schooler, "RFC 3261: SIP: Session Initiation Protocol," June 2002, status: IETF Standard Track.

[4] Henry Sinnreich and Alan B. Johnston, *Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol*. New York, USA: John Wiley and Sons, Inc., 2001.

[5] "SIP for Instant Messaging and Presence Leveraging Extensions (simple)," At http://www.ietf.org/html.charters/simple-charter.html, IETF simple working group.

[6] "Liberty specs tutorial," At http://www.projectliberty.org/liberty/content/download/423/2832/file/tutorialv2.pdf, 2004, Liberty Alliance specification.

[7] Scott Cantor, John Kemp, Jeff Hodges, and Peter Thompson, "Liberty ID-FF Architecture Overview," 2005, Liberty Alliance specification.

[8] Scott Cantor, John Kemp, and Darryl Champagne, "Liberty id-ff bindings and profiles specification," 2005, Liberty Alliance specification.

[9] Scott Cantor and John Kemp, "Liberty id-ff protocols and schema specification," 2005, Liberty Alliance specification.

[10] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language(SAML) V2.0," March 2005, oASIS standard specification.

[11] M. A. C. for Education (MACE), "Shibboleth (Internet2)," At http://shibboleth.internet2.edu/.

[12] Miikka Poikselka, Aki Niemi, Hisham Khartabil, and Georg Mayer, *The IMS: IP Multimedia Concepts and Services*. New York, USA: John Wiley and Sons, Inc., 2006.

[13] H. Tschofenig, J. Peterson, J. Polk, D. Sicker, and J. Hodges, "SIP SAML Profile and Binding," Oct 2006, status: IETF Draft Standard.

[14] Scott Cantor, Frederick Hirsch, John Kemp, Rob Philpott, and Eve Maler, "Bindings for the OASIS Security Assertion Markup Language(SAML) V2.0," March 2005, oASIS standard specification.

[15] N. Deason, "SIP and SOAP," June 2000, status: IETF Draft Standard.

[16] "Neon wiki website," At https://trustinet.hiit.fi/neon/index.php/, 2007.

[17] "Nist sip project," At http://snad.ncsl.nist.gov/proj/iptel/, 2007.

[18] "Sip communicator project," At http://sip-communicator.org/, 2007.

[19] "Generic Authentication Architecture (GAA); Generic bootstrapping architecture," 3GPP, TS 33.220 3436, Dec 2006.

[20] "Liberty Alliance and 3GPP security interworking," 3GPP," TS 33.980, Sep 2006.