Advancement Towards Secure Authentication in the Session Initiation Protocol

Lars Strand
Norwegian Computing Center / University of Oslo
Oslo, Norway
Email: lars.strand@nr.no

Wolfgang Leister
Norwegian Computing Center
Oslo, Norway
Email: wolfgang.leister@nr.no

***Abstract*—The Digest Access Authentication method used in the voice over IP signaling protocol, SIP, is weak. This authentication method is the only method with mandatory support and widespread adoption in the industry. At the same time, this authentication method is vulnerable to a serious real-world attack. This poses a threat to VoIP industry installations and solutions. In this paper, we propose a solution that counters attacks on this wide-spread authentication method. We also propose a two-step migration towards a stronger authentication in SIP. We add support for a Password Authenticated Key Exchange algorithm that can function as a drop-in replacement for the widely adopted Digest Access Authentication mechanism. This new authentication mechanism adds support for mutual authentication, is considered stronger and can rely on the same shared password used by the digest authentication. A long-term solution is to replace the authentication scheme in SIP with a security abstraction layer. Two such security frameworks are introduced, discussed and evaluated: the Generic Security Services Application Program Interface and the Simple Authentication and Security Layer, which both enable SIP to transparently support and use more secure authentication methods in a unified and generic way.**

***Index Terms*—SIP, authentication, Digest Access Authentication, PAKE, SASL.**

I. INTRODUCTION

Considering the growing market share for Voice over IP (VoIP) technologies, VoIP services need to be stable and secure for the benefit of both users and service providers. Authentication methods are an important part of this and need to be thoroughly examined. We base our current work on a conference article [1], where we analyzed and implemented an attack on the Digest Access Authentication used in the Session Initiation Protocol (SIP) and proposed a correction to mitigate this attack. Since there is a need for better authentication methods in SIP, we add support for a security abstraction layer in SIP [2] and propose a migration strategy towards a secure authentication in SIP [3].

The importance of analyzing and improving the SIP authentication methods comes from the fact that there has been a steady increase in the number of VoIP users since 2002, as well as a decrease in the number of PSTN installations [4]. With two billion users worldwide having access to the Internet by the end of 2010 [5], the VoIP growth potential is huge. For example, at the end of 2009, 29.1 % of the private land-line phone market in Norway used VoIP.

VoIP is the emerging technology that will eventually take over from the traditional Public Switched Telephone Network (PSTN) [6] due to VoIP's improved flexibility and functionality, such as improved sound quality ("HD sound") using wideband codecs like G.722 [7], instant messaging (IM), presence, mobility support, and secure calls. VoIP also reduces maintenance and administration costs since it brings convergence to voice, video and data traffic over the IP infrastructure.

Although there exist several competing network protocols that are capable of delivering VoIP, the Session Initiation Protocol (SIP) [8] and the Real-time Transport Protocol (RTP) [9] developed by the IETF have become the *de facto* industry standard. These two protocols fulfill two different functions – SIP is used for signaling, e.g., responsible for setting up, modifying and tearing down multimedia sessions, while RTP transports the actual media stream (voice). Although the SIP protocol is flexible and rich in functionality [10], several vulnerabilities and security attacks have been found [11]–[13].

Securing a SIP-based VoIP system has proven challenging and the reasons are multi-faceted:

- The scale and complexity of the SIP protocol specification, with primary focus on functionality rather than a sound security design [14].
- SIP usage of intermediaries, expected communication between nodes with no trust at all, and its user-to-user operation make security far from trivial [8, page 232].
- A large number of threats against VoIP systems have been identified [15]. Several security mechanisms for countermeasures have been proposed, but no single security mechanism is suited to address all these security threats concerning VoIP and SIP [16], [17].
- Since the SIP and RTP protocols share the same infrastructure as traditional data networks, they also inherit the security problems of data communication.
- VoIP services have strict requirements to the network performance with respect to Quality of Service since it is a duplex communication with low tolerance for latency, packet loss and saturation. Introducing strong security mechanisms might affect network performance [18].

PSTN is a mature and stable technology providing 99.999% uptime [19], and users will expect VoIP to perform at similar service level. But with an increasing number of VoIP users, VoIP will become a target for attackers looking for financial gain or mischief. A clear threat taxonomy is given by the "VoIP Security Alliance" [15] and is discussed by Keromytis [20].

In VoIP, authentication tries to validate the identity of the communication peers and to bind that identity to a subject (peer). It must be stressed that the user's phone is authenticated rather than the user herself. In VoIP terminology, a subject could be a User Agent (UA), such as a phone, identified by a phone-number/username and IP-address/hostname pair, denoted as an Address-of-Record (AoR). The authentication in VoIP is therefore the assurance that a communicating entity, the UA, is the one that it claims to be [21]. Equally important for the UA is to establish the identity of the communicating peer, i.e., the SIP server. If the client does not authenticate the SIP server, it might risk to communicate and send content to a hostile SIP server.

SIP supports several security services, and the RFC specification documents recommends their use. These security services can provide protection for authentication, confidentiality, and more. Yet, only one such security service is mandatory: the SIP Digest Access Authentication (DAA) method [8, page 193]. In the EUX2010sec research project [22], we revealed, in close collaboration with our project partners, that most VoIP installations only use the mandatory, Digest Access Authentication (DAA) method [23]. DAA is primarily based on the HTTP Digest Access Authentication [24], and is considered to be weak and vulnerable to serious real-world attacks [25].

One contribution of this paper is to present and analyze the seriousness of a vulnerability we presented in our earlier work – the registration attack [25]. We implement a real-world attack, and propose a solution to the DAA that will counter this vulnerability. Further, we introduce an authentication method based on the Password Authenticated Key Exchange (PAKE) [26], which provides mutual authentication based on a shared secret, and can function as a drop-in replacement of the digest authentication currently used. However, a more flexible authentication method is desired. Different security requirements may require different authentication mechanisms. Instead of adding support for many different authentication mechanisms in SIP, we introduce support for a security abstraction layer. Two such security frameworks are introduced, discussed and evaluated. The Generic Security Services Application Program Interface (GSS-API) [27] and Simple Authentication and Security Layer (SASL) [28], which both enables SIP to transparently support and use more secure authentication methods in a unified and generic way.

The rest of the paper is organized as follows: Related work and the current state of authentication in SIP is given in Section II, and show our method in Section III. We explain and implement the registration attack, and propose a solution on how to counter the attack in Section IV. In Section V we show how a modified PAKE can be used to add mutual authentication in SIP. Support for the security abstraction layers GSS-API and SASL is added, discussed and evaluated in Section VI. We present the conclusion and future work in Section VIII.

II. State of knowledge

The DAA is currently the most common authentication mechanism for SIP. DAA is simple, but rather insecure. It is the only authentication mechanism which support in SIP is mandatory [8, Section 22]. DAA uses the MD5 hash function and a challenge-response pattern, and relies on a shared secret between client and server within a SIP domain [24]. DAA is performed during the SIP `REGISTER` handshake between the UA and the SIP server, as depicted in messages 1-3 and 6 in Fig. 10. The UA receives a nonce value from the SIP server, computes a digest hash value over the nonce, the shared secret and some other SIP header values, and send it to the SIP server. The SIP server computes the same digest hash. If both digests are identical, the UA is authenticated. The DAA is weak and vulnerable to a serious real-world attack, as described in Section IV-A.

Based on the DAA, Undery [29] proposed a more flexible use of variables protected by the digest. His paper addresses the shortcomings of DAA and suggests to allow the server to decide which headers it requires to be included and protected by the digest computation. Unfortunately, his approach does not require specific headers fields to be included. His approach is therefore vulnerable to the same vulnerability presented and implemented in this paper.

Yang et al. [30] also conclude that DAA is weak. They argue that, since DAA is vulnerable to an off-line password guessing attacks, a more secure authentication method is required. They propose an authentication method based on Diffie-Hellman. Unfortunate, they do not discuss nor add any additional SIP headers in their new authentication scheme. Therefore, their solution is also vulnerable to the registration attack implemented in this paper.

Secure MIME (S/MIME) [31] is an authentication mechanism presented in the SIP core specification document RFC3261 [8]. S/MIME intends to achieve end-to-end authentication between UAs. The entire SIP message is encapsulated in a specific SIP message using MIME, which is signed and optionally encrypted. The receiving UA checks whether the sending UA's certificate is signed by a trusted authority. Since S/MIME depend on end-user certificates, the UAs must support multiple root certificates since no consolidated certificate authority exists. Additionally, certificate handling issues, such as revocation and renewal, complicate the use of certificates. There has been rather limited industry support for S/MIME.

Transport Layer Security (TLS) [32] support for SIP, called "Secure SIP" and denoted "SIPS", has gained some industry momentum. TLS is designed to make use of TCP to provide a protected end-to-end communication between two endpoints. The application data, here SIP, are encrypted and integrity-protected. The communicating endpoints authenticate using digital certificate, usually X.509 certificates, and thus require a public key infrastructure (PKI). TLS does not offer end-to-end confidentiality and integrity protection of SIP messages, since the TLS connection must be terminated and initiated for each hop between intermediate SIP servers. The use of TLS also restricts SIP to use TCP as transport protocol. By using TLS, SIP relies on a lower communication layer protocol to enforce security mechanisms.

Two other authentication methods have emerged within the Internet Engineering Task Force (IETF):

1) The *P-Asserted Identity* [33] is intended to work within a trusted environment. An unprotected SIP header is appended by the UAs SIP server that informs the receiving SIP server that the identity of the UA has been checked and thus can be trusted. However, since the SIP header is sent in clear rather than protected by cryptography methods, it can easily be removed by an attacker without any of the communicating peers noticing this.
2) The *SIP Strong Identity* [34] introduces a new SIP service, the "authentication service", which signs a hash over selected SIP header values, and includes the signature as a SIP header along with a URI that points to the sender's certificate. The receiver computes the same hash and compares the results. However, using this method, only the client is authenticated and an attacker can remove these headers without implications.

Note that both "P-Asserted Identity" and "SIP Strong Identity" rely on a successful DAA authentication to be applicable. These are also applied by the SIP servers rather than the clients themselves, and are thus only providing indirect authentication of the client since the server is authenticating on behalf of the client. None of these authentication methods have seen any widespread deployment yet [14].

Palmieri et al. [35], [36], dismiss DAA as a usable authentication method, and instead craft a new authentication schema with digital signatures based on public-key encryption. But since they rely on certificates, their solution suffers under similar certification handling issues as S/MIME and TLS. They also admit that relying on PKI is both difficult and costly to implement. Liao et at. [37], propose an improved authentication in SIP with self-signed public keys on elliptic curves. However, Liao's proposal uses smart-cards to store authentication data and rely on a trusted third party [38].

The H.323 recommendation for the VoIP protocol from the International Telecommunication Union (ITU) has failed to see widespread adoption by industry players, and is considered abandoned in favor of SIP/RTP [10]. The authentication methods in H.323, specified in H.235 [39], [40] uses well established security mechanism, like certificates, and Diffie-Hellman key exchange, to enforce authentication. Further analysis is needed to see whether the H.235 standard protects the signaling better than SIP.

The Inter-Asterisk eXchange (IAX) [41], also published by the IETF, establishes a competing protocol to SIP/RTP. IAX has several security properties that are better than SIP. By multiplexing channels over the same link and transporting both signaling and media over the same port, enforcing security mechanisms is easier. IAX supports two authentication methods: *1*) MD5 Message Digest authentication [42] computed

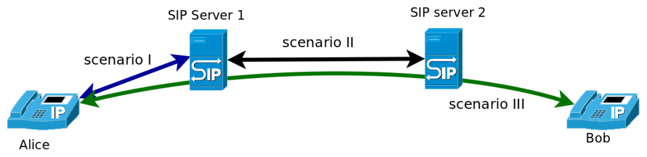


Fig. 1: Three different usage scenarios where authentication in SIP is desired.

over a pre-shared secret and a challenge (nonce), or *2*) using RSA public-key encryption on the challenge. In both methods, the nonce value is the only protocol parameter that is integrity protected by the authentication. Future work needs to investigate whether the IAX authentication method is adequately secure.

The SIP protocol needs an authentication mechanism that avoids the security vulnerabilities the currently used DAA has. A replacement authentication mechanism should preferably not rely on PKI, have support for strong mutual authentication, and support all three scenarios listed in the upcoming Section III.

III. Method and case study

In Norway, both private companies and public authorities are migrating from PSTN to VoIP [23]. To create suitable scenarios we study the VoIP installation of three companies in Norway; one medium sized company with 150 employees, and two larger companies with 3000 and 4700 employees. We have gathered several of these VoIP configurations and setups, and replicated the installations in our test lab [43]. In these companies, most of the employees have their own VoIP phone, called a User Agent (UA). All VoIP servers run the Linux operating system with the open source telephony platform *Asterisk* [44]. We found in these configurations that the digest authentication is the only authentication method for the UAs.

In the following paragraphs, the numbers in parentheses refer to the numbers in Fig. 2, where the workflow in our method is shown.

In order to gain knowledge of the SIP protocol we use the specification documents (1), here the SIP standard. Then, we analyze VoIP network traffic going through the test lab (5). We have implemented two VoIP setups based on configurations from our industry partners ((2) and (3)). The network traffic is intercepted and saved to file using the network tool *tcpdump* (4). The network traffic is then analyzed off-line using the packet analyzer, *Wireshark* (5). An example of such an analysis is shown in Fig. 3.

As an additional input we consider threats given by [15] and given in earlier work, such as a SIP attack analyzed by Hagalisletto and Strand [25], using the protocol analyzer PROSA (6). We explain this attack in more detail in Section IV-A, and implement and execute the attack using the network tool *NetSED* (7) as shown in Fig. 7. Based on the security requirements (9) obtained from the SIP specification, we then checked if the authentication method (10) was compromised

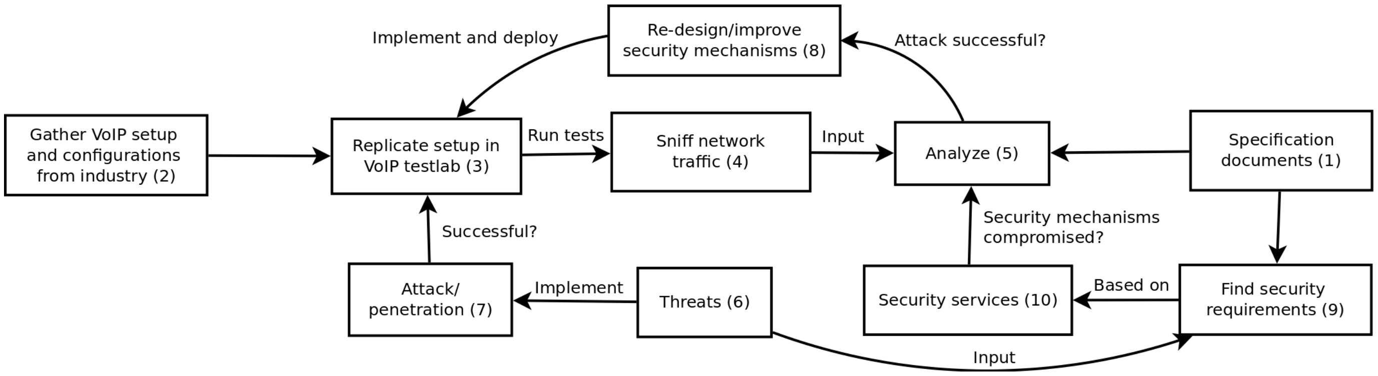
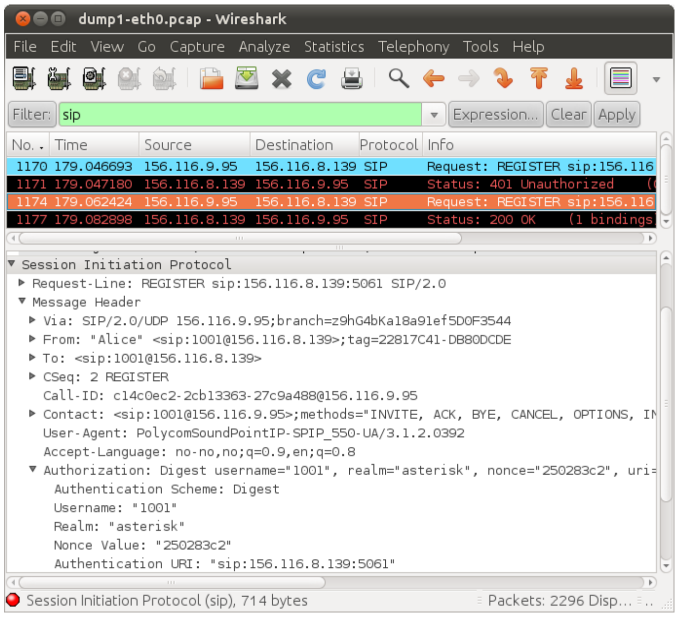

Gather VoIP setup and configurations from industry (2)
Replicate setup in VoIP testlab (3)
Run tests
Sniff network traffic (4)
Input
Analyze (5)
Specification documents (1)
Find security requirements (9)
Based on
Security services (10)
Security mechanisms compromised?
Attack successful?
Re-design/improve security mechanisms (8)
Implement and deploy
Successful?
Attack/ penetration (7)
Implement
Threats (6)
Input

Fig. 2: Workflow for analysis of the SIP authentication method.


dump1-eth0.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: sip
Expression... Clear Apply

```
No. . Time       Source        Destination   Protocol Info
1170 179.046693 156.116.9.95  156.116.8.139 SIP      Request: REGISTER sip:156.116
1171 179.047180 156.116.8.139 156.116.9.95  SIP      Status: 401 Unauthorized    (
1174 179.062424 156.116.9.95  156.116.8.139 SIP      Request: REGISTER sip:156.116
1177 179.082898 156.116.8.139 156.116.9.95  SIP      Status: 200 OK      (1 bindings

▼ Session Initiation Protocol
  ▶ Request-Line: REGISTER sip:156.116.8.139:5061 SIP/2.0
  ▼ Message Header
    ▶ Via: SIP/2.0/UDP 156.116.9.95;branch=z9hG4bKa18a91ef5D0F3544
    ▶ From: "Alice" <sip:1001@156.116.8.139>;tag=22817C41-DB80DCDE
    ▶ To: <sip:1001@156.116.8.139>
    ▶ CSeq: 2 REGISTER
      Call-ID: c14c0ec2-2cb13363-27c9a488@156.116.9.95
    ▶ Contact: <sip:1001@156.116.9.95>;methods="INVITE, ACK, BYE, CANCEL, OPTIONS, IN
      User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
      Accept-Language: no-no,no;q=0.9,en;q=0.8
    ▼ Authorization: Digest username="1001", realm="asterisk", nonce="250283c2", uri=
        Authentication Scheme: Digest
        Username: "1001"
        Realm: "asterisk"
        Nonce Value: "250283c2"
        Authentication URI: "sip:156.116.8.139:5061"

Session Initiation Protocol (sip), 714 bytes          Packets: 2296 Disp...
```

Fig. 3: Network analysis using the network tool Wireshark.

by the real-world attack. After careful analysis of the SIP headers we found that the SIP registration attack could be countered by a modification of the SIP authentication method (8).

We identify three scenarios where identity in SIP needs to be handled, as depicted in Fig. 1: Scenario *I* between the UA and the local SIP server; Scenario *II* between SIP servers; and Scenario *III* end-to-end.

Scenario I between the UA and the local SIP server is relevant when the UA comes online and before any outgoing calls can be placed. Then, the UA must register itself to a local SIP server. During the SIP register handshake, the server usually challenges the UA to authenticate. Before placing a call (sending a SIP `INVITE`), the UA might be challenged again by the server to authenticate. The most common authentication method used between UA and server today is the DAA.

Scenario II handles the authentication between SIP servers to achieve trust between SIP servers. It is not desirable to have SIP traffic handled by an unknown or untrusted SIP server that might have malicious intent. However, since most SIP servers today use some kind of SIP peering [6], the relationships between servers are often static and pre-defined. Therefore the identities between SIP servers are often predetermined by other security mechanisms than what are offered by SIP (like IPSec, TLS etc.).

Scenario III is about end-to-end authentication, which determines the identity of both the caller and the callee across different SIP domains. This is of particular importance and not easily attained in SIP. There is an increased threat and fear for both VoIP phishing and SPIT (Spam over Internet Telephony), that might seriously affect SIP-based VoIP services. By enforcing end-to-end authentication in SIP, these threats might be mitigated or prevented.

We list authentication mechanisms in SIP and their support in these three SIP scenarios in Table I on the facing page.

IV. Digest Access Authentication

The SIP Digest Access Authentication (DAA) [24] is currently the most common authentication scheme for SIP. Other authentication schemes have emerged, but DAA is the only mandatory authentication scheme [8, Section 22]. DAA uses a challenge-response pattern, and relies on a shared secret between client and server. Since the DAA relies on a shared secret and is only meaningful for a specific realm, its usage is limited to Scenario I.

SIP is heavily influenced by the HTTP request-response model, where each transaction consists of a request that requires a particular response. The SIP messages are also similar in syntax and semantics to both HTTP and SMTP [10]. A SIP message consists of several headers and a body. The SIP header fields are textual, always in the format `<header_name>: <header_value>`. The header value can contain one or more parameters. We show an example SIP header message in Fig. 5.

Any SIP request can be challenged for authentication. We show an example SIP DAA handshake in Fig. 4, and refer to the protocol clauses with a number in parentheses. The initial SIP `REGISTER` message (1) from Alice is not authorized and must be authenticated. The SIP server responds with a

TABLE I: List of SIP authentication mechanisms and their support.

	Supported authentication scenarios			Supported SIP methods	
Authentication mechanisms	scenario I	scenario II	scenario III	`REGISTER`	`INVITE`
Digest Access Authentication (DAA)	yes	no	no	yes	yes
Secure MIME (S/MIME)	no	no	yes	yes[a]	yes
Secure SIP (SIPS) using TLS	yes	yes	no[b]	yes	yes
P-Asserted Identity	no	yes	no	no	yes[c]
SIP Strong Identity	no	yes	no	no	yes[d]
Password Authenticated Key Exchange (PAKE)	yes	no	no	yes	yes
Generic Security Service API (GSS-API)	yes	yes	yes	yes	yes
Simple Authentication and Security Layer (SASL)	yes	yes	yes	yes	yes

[a] Not intended to be used with SIP `REGISTER`, however there are no constrains in the SIP specification for using S/MIME in addition to DAA.
[b] SIPS only offers hop-by-hop confidentiality and authentication protection and thus no end-to-end protection.
[c] Does not provide an authentication method *per se*, but provide identity authentication in a trusted environment.
[d] The authentication service is handled by intermediate SIP servers to verify UAs across SIP domains.

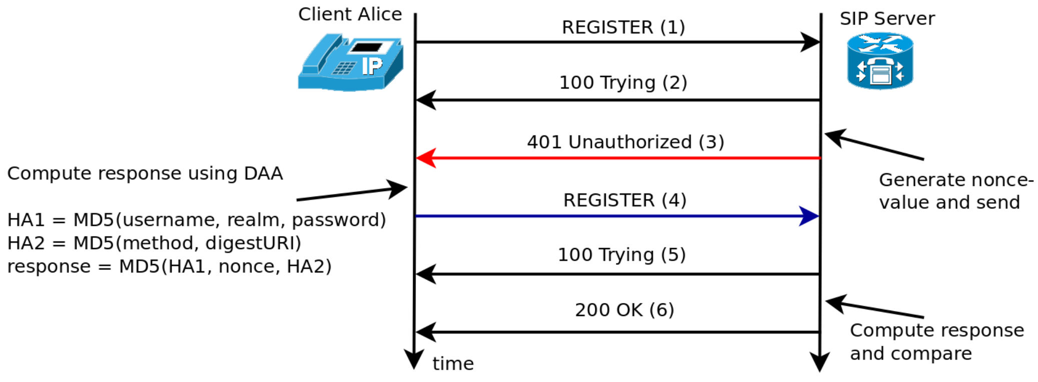

Client Alice
SIP Server
REGISTER (1)
100 Trying (2)
401 Unauthorized (3)
REGISTER (4)
100 Trying (5)
200 OK (6)
Compute response using DAA
HA1 = MD5(username, realm, password)
HA2 = MD5(method, digestURI)
response = MD5(HA1, nonce, HA2)
Generate nonce-value and send
Compute response and compare
time

Fig. 4: The SIP Digest Access Authentication method during a SIP `REGISTER` transaction.

`401 Unauthorized` status message (3) which contains a `WWW-Authenticate` header with details of the challenge, including a *nonce* value. The client computes the required SIP digest that is embedded in (4) as an `Authorization` header. The SIP server, upon receiving the `Authorization` header, must perform the same digest operation, and compare the result. If the results are identical, the client is authenticated, and a `200 OK` message (6) is sent.

The SIP DAA is almost identical to the HTTP digest access authentication [24]. As we will show later, too few attributes (SIP header values) are included in the digest computation, thus leaving some values unprotected. Formally, the DAA is expressed as follows:

$$\begin{aligned} \mathit{HA1} &= \mathrm{MD5}(A1) \\ &= \mathrm{MD5}(\mathit{username} : \mathit{realm} : \mathit{password}) \\ \mathit{HA2} &= \mathrm{MD5}(A2) = \mathrm{MD5}(\mathit{method} : \mathit{digestURI}) \\ \mathit{response} &= \mathrm{MD5}(\mathit{HA1} : \mathit{nonce} : \mathit{HA2}) \end{aligned}$$

In this context, *A1* is the concatenated string of Alice's *username*, the *realm* (usually a hostname or domain name) and the shared secret *password* between Alice and the server. For *A2*, the *method* is the SIP method used in the current transaction, in the above example that would be `REGISTER`. In a `REGISTER` transaction the *digestURI* is set to the URI

```
REGISTER sip:CompanyA SIP/2.0
Via: SIP/2.0/UDP
156.116.9.95;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
From: Alice <sip:alice@CompanyA>;tag=1234648905
To: Alice <sip:alice@CompanyA>
Contact: "Alice" <sip:alice@156.116.9.95:5060>
Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
CSeq: 19481 REGISTER
User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
Authorization: Digest
username="alice",realm="asterisk",nonce="3b7a1395",response="
ccbde1c3c129b3dcaa14a4d5e35519d7",uri="sip:CompanyA",algorith
m=MD5
Max-Forwards: 70
Expires: 3600
Content-Length: 0
```

Fig. 5: The only attributes included in the digest response (blue) are depicted in green.

in the *To:*-field. The digest authentication *response* is the hash of the concatenated values of *HA1*, the *nonce* received from the server, and *HA2*. A SIP `REGISTER` message with a computed digest embedded in the `Authorization` header is shown in Fig. 5. DAA provides only reply protection due to the nonce value and one-way message authentication. There is no encryption of the content, nor confidentiality support, except the shared secret *password* between client and server. All messages are sent in clear. DAA only works within a local domain so cross-domain authentication is not supported, which

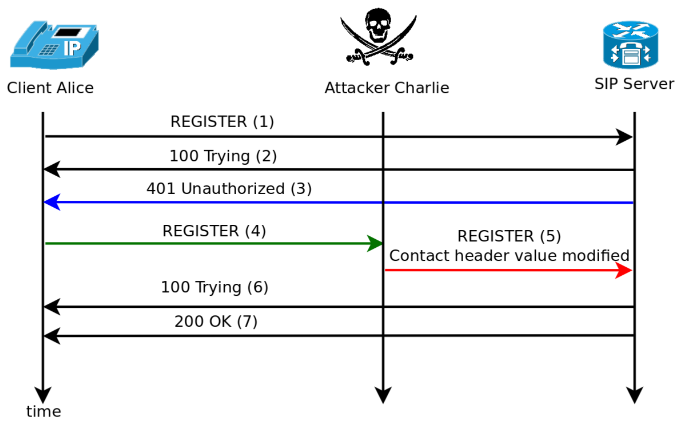


Fig. 6: The attacker Charlie can modify the `Contact` header value, and thereby have all Alice's calls redirected to him.

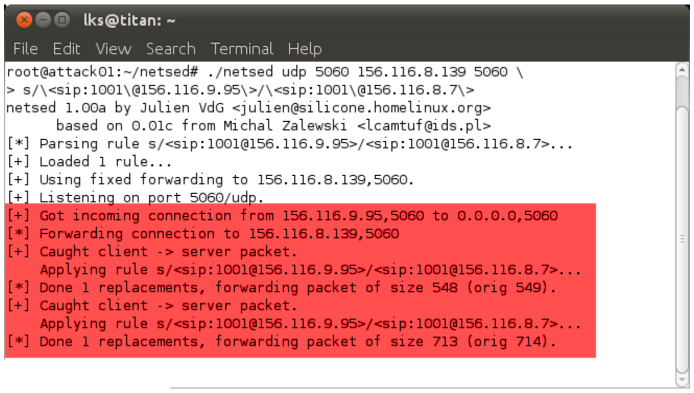
implies that end-to-end authentication is not supported. There is no provision in the DAA for the initial secure arrangement between a client and server to establish the shared secret. However, DAA has low computation overhead compared to other methods [18].

A. Attack on Digest Access Authentication

When a UA comes online it registers its contact point(s) to a *location service*. Contact points are the preferred methods a user can be contacted by, for example using SIP, mail, or IM. Usually, only a SIP URI contact method is present. The location service is responsible to redirect SIP requests (for VoIP calls) to the correct SIP end-point. For example, an incoming SIP call destined to `alice@CompanyA.org` does not contain information about which hostname or IP-address Alice's phone can be reached. Therefore, a SIP proxy will query the location service to receive Alice's phone's hostname or IP-address, and then forward the call to this address.

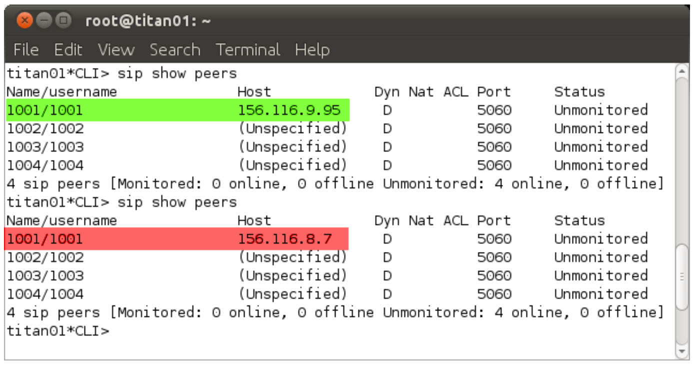
The binding of Alice's phone to a hostname or IP-address is done during the `REGISTER` transaction, as depicted in Fig. 4. Before the binding, or registration, the SIP server should ask the client to authenticate itself, as explained in the previous section. After a successful authentication, the client's hostname or IP-address is registered. A re-registration is normally done at regular intervals. This registration is repeated usually every 3-15 minutes, depending on the configuration. The client's preferred contact methods, including hostname or IP-address, is carried in the SIP header `Contact`, as depicted in Line 5 in Fig. 5. However, this SIP header value is sent in clear, and is not protected by DAA. Thus, the registration is vulnerable to a man-in-the-middle attack [25].

If an attacker modifies the hostname or IP-address in the *contactURI* header value during a `REGISTER` phrase, as depicted in Fig. 6, all requests, and hence calls, to the client will be diverted to a hostname or IP-address controlled by an attacker. Here, Alice cannot perceive that she is unreachable. An attacker can modify Alice's `REGISTER` session in real-time using NetSED [45] as depicted in Fig. 7. The SIP



```
root@attack01:~/netsed# ./netsed udp 5060 156.116.8.139 5060 \
> s/\<sip:1001\@156.116.9.95\>/\<sip:1001\@156.116.8.7\>
netsed 1.00a by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
[*] Parsing rule s/<sip:1001@156.116.9.95>/<sip:1001@156.116.8.7>...
[+] Loaded 1 rule...
[+] Using fixed forwarding to 156.116.8.139,5060.
[+] Listening on port 5060/udp.
[+] Got incoming connection from 156.116.9.95,5060 to 0.0.0.0,5060
[*] Forwarding connection to 156.116.8.139,5060
[+] Caught client -> server packet.
    Applying rule s/<sip:1001@156.116.9.95>/<sip:1001@156.116.8.7>...
[*] Done 1 replacements, forwarding packet of size 548 (orig 549).
[+] Caught client -> server packet.
    Applying rule s/<sip:1001@156.116.9.95>/<sip:1001@156.116.8.7>...
[*] Done 1 replacements, forwarding packet of size 713 (orig 714).
```

Fig. 7: The network packet stream editor NetSED modifies network packets in real time based on a regular expression (in red).



```
titan01*CLI> sip show peers
Name/username              Host            Dyn Nat ACL Port     Status
1001/1001                  156.116.9.95     D          5060     Unmonitored
1002/1002                  (Unspecified)    D          5060     Unmonitored
1003/1003                  (Unspecified)    D          5060     Unmonitored
1004/1004                  (Unspecified)    D          5060     Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 4 online, 0 offline]
titan01*CLI> sip show peers
Name/username              Host            Dyn Nat ACL Port     Status
1001/1001                  156.116.8.7      D          5060     Unmonitored
1002/1002                  (Unspecified)    D          5060     Unmonitored
1003/1003                  (Unspecified)    D          5060     Unmonitored
1004/1004                  (Unspecified)    D          5060     Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 4 online, 0 offline]
titan01*CLI>
```

Fig. 8: Host name before (green) and after a successful attack (red), which makes Asterisk believe that Alice's phone (with number 1001) is reachable at an IP-address of the attacker's choice.

server (Asterisk), will not detect nor suspect that anything is wrong, and register Alice's phone number with the attackers IP address, as seen on Asterisk's terminal in Fig. 8. When Asterisk receives a call to Alice, the call will be forwarded to the attackers registered IP address. If this vulnerability is left incorrect, it constitutes a fatal flaw.

B. Improving the Digest Access Authentication

The SIP digest authentication is weak, which is stated in both the SIP specification [8], and the digest specification [24]. Specifically, DAA only offers protection of the value in the `To` header called the `Request-URI` and the *method*, but no other SIP header values are protected.

A minor modification of DAA can counter the registration hijack attack [25], which is caused by having too few SIP header parameters protected by the digest. Since an attacker can modify and redirect all requests, we protect the header by including the `Contact` header value in the digest. By including the `Contact` value, which we name *contactURIs* in the digest, we effectively counter the registration hijack attack.

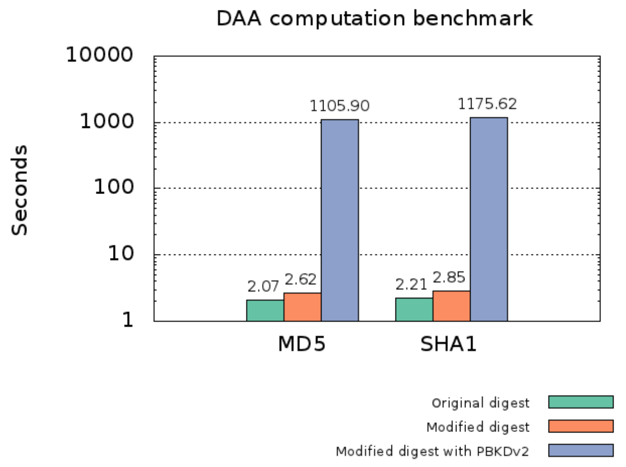


Fig. 9: The computation overhead for 100.000 iterations for original DAA, our modified DAA, and modified DAA with PBKDFv2 for both MD5 and SHA1.

We define $HA0$ with $contactURIs$. The new digest computation algorithm is as follows:

$$\begin{aligned} HA0 &= \text{MD5}(A0) = \text{MD5}(contactURIs) \\ HA1 &= \text{MD5}(A1) \\ &= \text{MD5}(username : realm : password) \\ HA2 &= \text{MD5}(A2) = \text{MD5}(method : digestURI) \\ response &= \text{MD5}(HA0 : HA1 : nonce : HA2) \end{aligned}$$

Weaknesses in the MD5 hash have been found. In particular we mention collision attacks where two different input values produce the same MD5 hash [46]. This weakness is not known to be exploitable to reveal a user's password [47]. Nonetheless, a stronger hash function, like SHA1 [48], is recommend.

We implemented and tested our modified DAA by using the Python Twisted [49] networking engine, using both MD5 and SHA1. According to our test, the computation overhead by including $HA0$ with the $ContactURIs$ is minimal, as shown in Fig. 9. The difference between the original DAA and our modified DAA with MD5 for 100.000 authentication requests on a 2.2Ghz Intel CPU, is only 0.55 seconds, a negligible amount.

A modified DAA means a modification of the SIP standard. Since the SIP standard has seen widespread industry adoption, it can be difficult to re-deploy a non-standardized SIP DAA. To prevent a modification of the SIP standard, we can use the DAA parameter `auth-param` to store our modified digest response. The parameter `auth-param` is reserved "for future use" [24, page 12], and can be a part of the `Authorization` header.

SIP devices that do not support the updated and more secure digest, can and will ignore this value, and use the original DAA for authentication. However, we cannot recommend this approach, since an attacker could remove this value and force the usage of the original standardized DAA. We would prefer to modify the DAA digest computation to force an upgrade to the new improved DAA method, instead of compromising on security.

C. Using a Password-Based Key Derivation Function

The improved DAA, described above, is still vulnerable to dictionary-based off-line (brute-force) attacks. The attacker can intercept the message exchange, and do an exhaustive (brute-force) search for the password. To increase the cost of such search, we add support for a key derivation technique with the purpose of increasing the cost of producing the digest from the shared secret, thereby also increasing the difficulty of the brute-force attack.

We introduce support for "Password Based Key Derivation Function version 2" (PBKDFv2) as specified by Kaliski [50] from RSA Laboratories. PBKDFv2 works by using a key derivation function (KDF) on the password (P) and salt (S) to derive the key (DK) as:

$$DK = \text{KDF}(P, S)$$

When applied to the DAA, P is the shared secret and S is the nonce issued from the SIP server. The DK is derived by these required steps:

1) The maximum length $dkLen$ of the derived key DK is given as:
$$dkLen > (2^{32} - 1) * hLen$$
where $hLen$ denotes the length in octets of the pseudo-random function output, which is 16 for MD5 and 20 for SHA-1. We implement and benchmark both MD5 and SHA-1. However, the use of MD5 is not recommended due to weaknesses and attacks found [51].
2) We let l be the number of $hLen$-octet blocks in the derived key, rounding up, and r the number of octets in the last block:
$$\begin{aligned} l &= \left\lceil \frac{dklen}{hLen} \right\rceil \\ r &= dkLen - (l - 1) * hLen \end{aligned}$$
3) For each block of the derived key, the function F is applied. The function F take password P, salt S, the iteration count c and the block index to compute the block:
$$T_1 = F(P, S, c, 1)$$
$$T_2 = F(P, S, c, 2)$$
$$...$$
$$T_l = F(P, S, c, l)$$
Here, function F is defined as the exclusive-or sum of the first c iterates of the underlying pseudo-random function PRF (using HMAC-SHA1 [52]) applied to the password P and the concatenation of the salt S and the block index i:
$$F(P, S, c, i) = U_1 \oplus U_2 \oplus ... \oplus U_c$$

where:

$$\begin{aligned} U_1 = &\quad PRF(P, S \parallel INT(i)) \\ U_2 = &\quad PRF(P, U_1) \\ &\quad ... \\ U_c = &\quad PRF(P, U_{c-1}) \end{aligned}$$

Here, $INT(i)$ is a four-octet encoding of the integer i, most significant octet first.

4) Then the blocks are concatenated and the first $dkLen$ octets is extracted to produce a derived key DK:

$$DK = T_1 \parallel T_2 \parallel ... \parallel T_l < 0..r-1 >$$

5) The derived key DK is returned base64-encoded [53].

We implemented and tested DAA with PBKDFv2 with c iterations set to the recommended value of 1000. Input to the PBKDFv2 is the password (shared secret) and the nonce from the SIP server. The new modified DAA replaces the *password* with the derived key DK from PBKDFv2, thus a modified DAA algorithm is as follows:

$$\begin{aligned} \mathit{HA0} = &\quad \mathrm{MD5}(A0) = \mathrm{MD5}(\mathit{contactURIs}) \\ \mathit{HA1} = &\quad \mathrm{MD5}(A1) \\ = &\quad \mathrm{MD5}(\mathit{username} : \mathit{realm} : DK) \\ \mathit{HA2} = &\quad \mathrm{MD5}(A2) = \mathrm{MD5}(\mathit{method} : \mathit{digestURI}) \\ \mathit{response} = &\quad \mathrm{MD5}(\mathit{HA0} : \mathit{HA1} : \mathit{nonce} : \mathit{HA2}) \end{aligned}$$

As shown in Fig. 9, the computation overhead using PBKDFv2 is significant compared to the original DAA. The result is as expected, since each DAA computation using PBKDFv2 calls a HMAC function 1000 times. This increase the cost of an exhaustive brute-force search for the shared secret used by DAA, without a significant impact of deriving individual DK used by a UA to authenticate with DAA.

While DAA with PBKDFv2 reduces much of the risk of a brute-force dictionary attack, it does not provide us with means to authenticate the SIP server.

V. Password Authenticated Key Exchange

In the following, we discuss how to add support for a variant of PAKE denoted as "Key Agreement Method 3" (KAM3) as a cryptographic protocol [26, page 17]. PAKE has the following attractive features: *1*) PAKE provides mutual authentication between UA and the SIP server, and thus a rogue SIP server can not claim that the authentication succeed without knowing the shared password. PAKE assures the UA that the SIP server knows the UA's encrypted password. *2*) Reuse of the shared password used by DAA as the UA's credential, which enables our approach to easily replace DAA used within a local SIP domain (scenario I). *3*) PAKE offers strong protection of the shared secret if the communication is eavesdropped, that prevents brute-force attacks, including dictionary-based off-line attacks, to which the DAA is vulnerable to.

Our approach follows the work of Oiwa et al. [54]. They use KAM3 to introduce a stronger authentication in HTTP and their initial design and specification is submitted to the IETF as an Internet Draft [55]. We have adapted their approach to SIP, since SIP closely resembles HTTP in both message structure and flow, and we need to prevent the `REGISTER` hijack attack presented earlier [1].

In KAM3, the UA and the SIP server compute cryptographic keys based on the shared password. These keys are exchanged, and a shared session secret is computed based on these keys. Each peer sends then a hash value computed of the session secret and some other values, to the requesting peer. The receiving peer computes the same hash value, and compares it with the received hash value. If these are identical, the sending peer is authenticated.

PAKE supports several authentication algorithms, which differ in their underlying mathematical groups and security parameters [55]. The only mandatory supported authentication algorithm, the *iso-kam3-dl-2048-sha256*, uses the 2048-bit discrete-logarithm defined in RFC3526 [56] and the SHA-256 hash function.

A. Initial requirements

In the following section, we let q an odd prime integer defining the number of elements in $F(q)$ which is a representation of a finite group. We let g the generator of a subgroup of r elements in $F(q)$. The one-way hash function is denoted as H.

Before the authentication starts, username and password must be set and configured. We compute a weak secret π used by the client as a one-way hash of the values realm, $\mathit{username}$ and $\mathit{password}$:

$$\pi = H(\mathit{realm}, \mathit{username}, \mathit{password})$$

Here, realm is the protection domain where SIP authentication is meaningful for a set of $\mathit{username}$ and $\mathit{password}$. The server does not need to store the shared password directly, only a specially encrypted version $J(\pi)$, where J is the password verification element derivation function defined as:

$$J(\pi) = g^{\pi} \bmod q$$

B. PAKE message exchange

We need to extend the current SIP `REGISTER` handshake by one extra round-trip of SIP messages between the UA and the SIP server. These two extra messages are depicted in blue and numbered (4) and (5) in Fig. 10. A more detailed specification is given in the following paragraphs, where the numbers refer to the protocol clauses depicted in Fig. 10.

The UA registers to a SIP *location service* (SIP server). The initial SIP `REGISTER` message (1) from the UA is not authorized, and must be authenticated. The SIP server responds with a `401 Unauthorized` status message (2), which contains a `WWW-Authenticate` header with details of the challenge, including *realm* and *algorithm*. The UA constructs a cryptographic value w_a generated from a random integer s_a:

$$w_a = g^{s_a} \bmod q$$

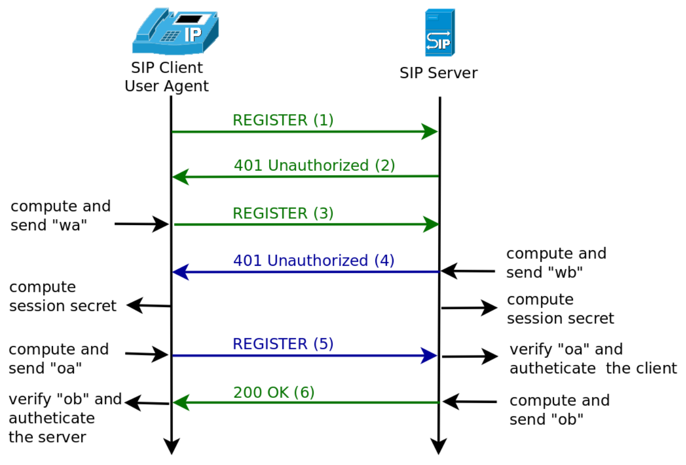


Fig. 10: SIP `REGISTER` message flow with mutual authentication security using PAKE.

This value is sent in a new SIP `REGISTER` message (3) to the SIP server. The SIP server proceeds to generate and send another cryptographic value w_b, which is generated from $J(\pi)$, the received value w_a and a random integer s_b:

$$w_b = (J(\pi) \times w_a^{H(1,w_a)})^{s_b} \bmod q$$

At the next step, each peer computes a session secret z. The UA derives z based on π, s_a, w_a and w_b:

$$z = w_b^{(s_a+H(2,w_a,w_b))/(s_a*H(1,w_a)+\pi) \bmod r} \bmod q$$

Likewise, the SIP server derives z based on s_b, w_a and w_b using the following function:

$$z = (w_a \times g^{H(2,w_a,w_b)})^{s_b} \bmod q$$

The session secret z matches only if both peers have used the secret credentials generated from the same shared secret. The above equations are directly derived from the PAKE HTTP authentication specifications [55]. The next step is to validate the value of z at the communicating peer.

The UA sends a third SIP `REGISTER` message (5) and includes the value o_a which is a hash value computed as:

$$o_a = H(4, w_a, w_b, z, contactURIs)$$

Here, $contactURIs$ is the value of the UA's `Contact` SIP header value. This value is integrity-protected to prevent register hijacking attacks as presented in [1]. The SIP server, upon receipt of o_a, performs the same hash operation, and compares the results. If these results are identical, the UA is authenticated. The SIP server then sends a final message (6), with the value o_b computed as:

$$o_b = H(3, w_a, w_b, z, contactURIs)$$

When the UA receives o_b, it verifies this value by computing its hash value. If the results are identical, the SIP server is authenticated to the UA. After a complete message exchange, the UA is authenticated to the SIP server, and the SIP server has been authenticated to the UA.

C. SIP message support for PAKE

We embed the cryptographic values derived in the previous section as base64-encoded [53] SIP header values. We re-use the SIP DAA headers to carry PAKE authentication data, so that PAKE can be used as a drop-in replacement for DAA. A SIP `REGISTER` message with a DAA `Authorization` header is depicted in Fig. 15. Again, we refer to the protocol clauses with a number in parentheses as depicted in Fig. 10.

The UA first sends a SIP `REGISTER` without any authentication credentials (1). The SIP server responds with a `401 Unauthorized` status message (2), which contains a `WWW-Authenticate` header with header values *realm* and *algorithm*:

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: Mutual realm="asterisk",
  algorithm="iso-kam3-dl-2048-sha256"
```

The UA then computes w_a and sends it to the SIP server using a new SIP `REGISTER` message (3), with the required values embedded in the `Authorization` header:

```
SIP/2.0 REGISTER
Authorization: Mutual user="alice",
  algorithm="iso-kam3-dl-2048-sha256",
  wa="Q29tcHV0ZWQgd2E...ljaCBcyBsb25nCg=="
```

The next required values in the authentication mechanism w_b, o_a and o_b are embedded and sent using these two SIP headers.

VI. Security Programming Interfaces

A modified PAKE authentication can more easily replace the current digest (DAA) authentication used in SIP, since they both rely on a shared secret and use the same SIP headers. PAKE also introduces a stronger authentication than DAA. However, a more flexible authentication mechanism is desired. Different VoIP scenarios require different security requirements, and the communicating peers should be able to negotiate the best possible authentication mechanism supported.

Instead of adding numerous different authentication mechanisms to SIP based on different security requirements, it is desirable to keep the changes to the SIP standard to a minimum. The industry might also be reluctant to adopt immature and non-standardized security services, like different (new) authentication mechanisms. Adding support to a security programming interface will require only small changes to the SIP standard.

A security programming interface provides a generic interface for application layer protocols like SIP, with a layer of abstraction for different security services like authentication, integrity or confidentiality. Using a security programming interface, an application does not need to support or implement every authentication method, but use the provided security API [57]. Support for two security programming interfaces, the “Generic Security Services API” (GSS-API) and “Simple Authentication and Security Layer” (SASL), are added to SIP.

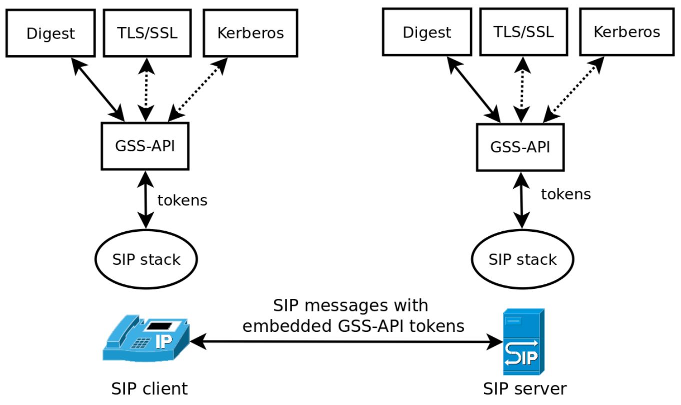


Fig. 11: The GSS-API interface in SIP.

Both are developed by the IETF, have been extensively tested, and are now classified as mature standards by the IETF.

A. Generic Security Services API

The GSS-API is not a communication protocol in itself, but relies on the application to encapsulate, send, and extract data messages called "tokens" between the client and server. The tokens' content are opaque from the viewpoint of the calling application, and contain authentication data, or, once the authentication is complete, portion of data that the client and server want to sign or encrypt. The tokens are passed through the GSS-API to a range of underlying security mechanisms, ranging from secret-key cryptography, like Kerberos [58], to public-key cryptography, like the Simple Public-Key GSS-API Mechanism (SPKM) [59]. The GSS-API interface to SIP is depicted in Fig. 11. For an application, the use of the GSS-API becomes a standard interface to request authentication, integrity, and confidentiality services in a uniform way. However, GSS-API does not provide credentials needed by the underlying security mechanisms. Both server and client must acquire their respective credentials before GSS-API functions are called.

To establish peer entity authentication, a security context is initialized and established. After the security context has been established, additional messages can be exchanged, that are integrity and, optionally, confidentially protected. To initiate and manage a security context, the peers use the *context-level* GSS-API calls. The client calls `GSS_Init_sec_context()` that produces a "output_token" that is passed to the server. The server then calls `GSS_Accept_sec_context()` with the received token as input. Depending on the underlying security mechanism, additional token exchanges may be required in the course of context establishment. If so, `GSS_S_CONTINUE_NEEDED` status is set and additional tokens are passed between the client and server until a security context is established, as depicted in Fig. 14.

After a security context has been established, *per-message* GSS-API calls can be used to protect a message by adding a Message Integrity Code (MIC) with `GSS_GetMIC()` and verifying the message with `GSS_VerifyMIC()`. To encrypt and decrypt messages, the peers can use `GSS_Wrap()` and `GSS_Unwrap()`. Thus, two different token types exist:

1) *Context-level tokens* are used when a context is established.
2) *Per-message tokens* are used after a context has been established, and are used to integrity or confidentiality protect data.

In addition to send and receive tokens, the application is responsible to distinguish between token types. This is necessary because different tokens types are sent by the application to different GSS-API functions. But since the tokens are opaque to the application, the application must use a method to distinguish between the token types. In our solution, we use explicit tagging of the token type that accompanies the token message.

1) SIP message support for GSS-API: When a SIP client is authenticated to a server using DAA, the authentication handshake data is encapsulated in the `WWW-Authenticate` header from server to client, and the `Authorization` header from client to server. We reuse these headers for GSS-API support, and instead of encapsulate DAA data, we send the GSS-API tokens. An example of both DAA `Authorization` header and the new `Authorization` header with GSS-API data is depicted in Fig. 12.

During the initialization of a security context it is necessary to identify the underlying security mechanism to be used. The caller initiating the context indicates at the start of the token the security (authentication) mechanism to be used. The security mechanism is denoted by a unique Object Identifier (OID). For example, the OID for the Kerberos V5 mechanism is `1.2.840.113554.1.2.2`. However, the initiating peer cannot know which security mechanism the receiving peer supports. If an unsupported "mech_type" is requested, the authentication fails. The GSS-API standard resolves this by recommending to manually standardizing on a fixed "mech_type" within a domain. Since SIP addresses are designed to be global [6], and not confined to a local domain, a GSS-API *negotiation* mechanism is required. The SPNEGO is such a GSS-API negotiation mechanism.

The "Simple and Protected GSSAPI Negotiation Mechanism" (SPNEGO [60] is a pseudo security mechanism that enables peers to negotiate a common set of one or more GSS-API security mechanisms. The GSS-API stack with SPNEGO is shown in Fig. 13. The client sends a prioritized list of supported authentication mechanisms to the server. The server then chooses the preferred authentication method based on the received list from the client. The client initiates `GSS_Init_sec_context()` as with an ordinary GSS-API security mechanism, but requests that SPNEGO is used as the underlying GSS-API mechanism ("mech_type"). The SPNEGO handshake between client and server is communicated by sending and receiving tokens. After the handshake, the client and server initiate and set up a security context (authentication) using the agreed GSS-API security mechanism.

```
REGISTER sip:CompanyA SIP/2.0
Via: SIP/2.0/UDP 192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
From: Alice <sip:alice@CompanyA>;tag=1234648905
To: Alice <sip:alice@CompanyA>
Contact: "Alice" <sip:alice@192.168.1.102:5060>
Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
CSeq: 19481 REGISTER
User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
Authorization: Digest
username="alice",realm="asterisk",nonce="3b7a1395",response="ccbde1c3c129b3dcaa14a4d5e35
519d7",uri="sip:CompanyA",algorithm=MD5
Max-Forwards: 70
Expires: 3600
Content-Length: 0
```

```
REGISTER sip:CompanyA SIP/2.0
Via: SIP/2.0/UDP 192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
From: Alice <sip:alice@CompanyA>;tag=1234648905
To: Alice <sip:alice@CompanyA>
Contact: "Alice" <sip:alice@192.168.1.102:5060>
Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
CSeq: 19481 REGISTER
User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
Authorization: GSSAPI ttype="context"
token="0401000B06092A864886F712010202DACD139402AAF44350CDE32"
Max-Forwards: 70
Expires: 3600
Content-Length: 0
```

Fig. 12: A SIP `REGISTER` message with the original DAA `Authorization` header to the left, and the same header carrying GSS-API data to the right.

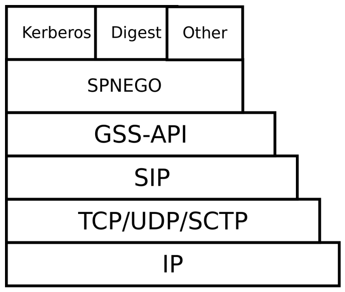


Fig. 13: The GSS-API protocol stack with the SPNEGO negotiation mechanism and underlying security mechanisms.

2) SIP authentication using GSS-API and SPNEGO: When discussing PAKE authentication earlier, we added one round-trip of SIP messages between the UA and the SIP server. When using GSS-API with the SPNEGO, the number of SIP messages going back and forth depends on the underlying authentication mechanism. We therefore extend the SIP `REGISTER` handshake with an arbitrary number of round-trips, until the underlying authentication mechanism has completed communication.

In the following paragraphs, the numbers in parentheses refer to the numbers in Fig. 14. When a client comes online and registers itself to a "location service" (SIP server), it does so by sending a SIP `REGISTER` message (1). We define the token type in the variable `ttype`. In the following messages, the `ttype` is set to "context" indicating that these tokens are *context-level tokens.* The first message (1) does not contain any `Authorization` header. The server responds with an empty `WWW-Authenticate` header (3):

```
REGISTER SIP/2.0
WWW-Authenticate: GSSAPI ttype="context"
  token=""
```

The client then calls `GSS_Init_sec_context()` with SPNEGO as underlying GSS-API mechanism to negotiate a common authentication mechanism (4). The GSS-API "mech_type" is set to SPNEGOs OID `1.3.6.1.5.5.2`. The token data might be in binary format, depending on the security mechanism used. Since the SIP headers are in ASCII string format, the token data is base64 encoded:

```
SIP/2.0 401 Unauthorized
Authorization: GSSAPI ttype="context"
  token="0401000B06092A864886F7120..."
```

The server retrieves the GSS-API data, the token, and passes this to the SPNEGO GSS-API mechanism. In this first initial token, the client embeds authentication data for its first preferred authentication mechanism. This way, should the server accept the clients preferred mechanism, we avoid an extra SIP message round trip. If the client's preferred method was accepted by the server, the server passes the relevant authentication data to the selected authentication mechanism in a `401` SIP message (5). The selected authentication method continues to pass tokens between client and server as many times as necessary to complete the authentication (6-7-N) and establish a security context. Once the security context is established, it sends a `200 OK` SIP message (N+2). Should the server have some last GSS-API data to be communicated to the client to complete the security context, it can be carried in a `WWW-Authenticate` header embedded in the `200 OK` message:

```
SIP/2.0 200 OK
WWW-Authenticate: GSSAPI ttype="context"
  token="dd02c7c2232759874e1c20558701..."
```

If the client's preferred mechanism is not the server's most preferred mechanism, the server outputs a negotiation token and sends it to the client embedded in a new `401` SIP message (5). The client processes the received SIP message and passes the authentication data to the correct authentication mechanism. The GSS-API then continues as described in the previous paragraph.

B. Simple Authentication and Security Layer

The Simple Authentication and Security Layer (SASL), defined in RFC4422 [28], provides an interface for authentication and an authentication negotiation mechanism. It provide the same security services as GSS-API and is implemented and used in several popular communications protocols applications like IMAP, SMTP and LDAP[1].

As with GSS-API, the SASL framework does not provide authentication mechanisms in itself, but supports different underlying authentication mechanisms through a standardized

[1]The Carnegie Mellon University's implementation: http://asg.web.cmu.edu/sasl/ and the GNU SASL library: http://www.gnu.org/software/gsasl/ are two popular and freely available SASL libraries.

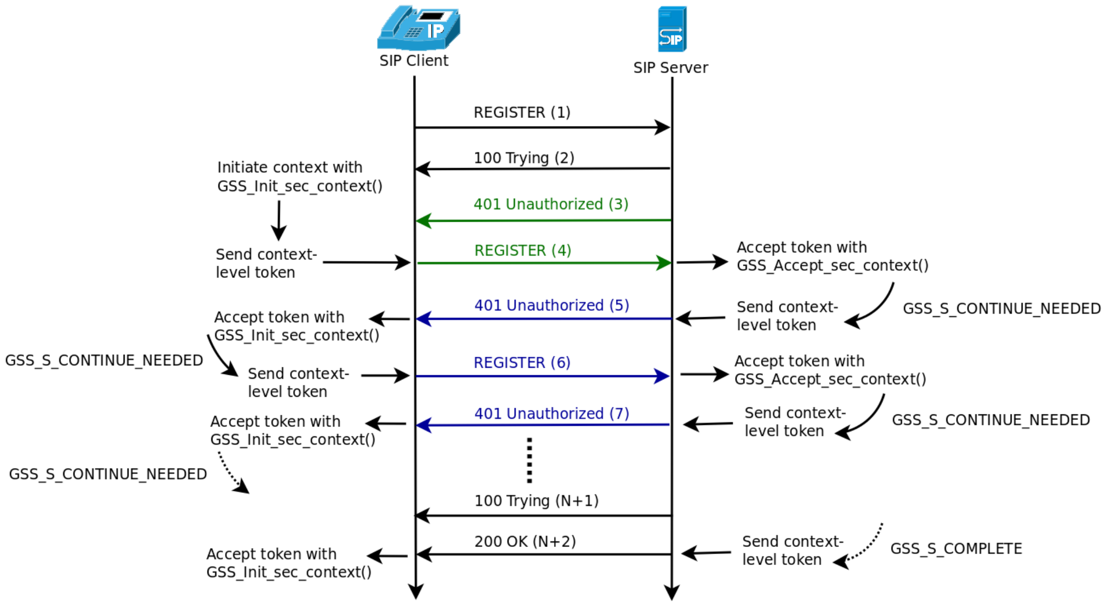


Fig. 14: SIP `REGISTER` message flow with GSS-API security context establishment (authentication).

interface[2]. SASL does not provide a transport layer and thus relies on the application, to encapsulate, send and extract SASL messages between client and server, which in our case is the SIP protocol. The SASL messages sent between client and server contain authentication data, and are opaque from the viewpoint of the calling application (SIP). The application only needs to add support to a SASL software library implementation, and thus have support to a range of underlying authentication mechanisms the library supports.

While the GSS-API is primarily intended for use with applications, SASL is used in, and intended for, communication protocols. The functionalities offered by the GSS-API and SASL are alike, but the SASL specification is more high-level, and allows more freedom in implementing the SASL requirements. SASL also supports more underlying security mechanisms than the GSS-API. By using the “GS2” mechanism family, the GSS-API can be used as an underlying security mechanism in SASL. However, the GSS-API negotiation mechanism SPNEGO cannot be used due to security concerns [61, Section 14].

1) SIP message support for SASL: In SASL terminology, the description on how to encapsulate SASL negotiation and SASL messages for a given protocol, is called a “SASL profile”. The SIP protocol stack with SASL is shown in Fig. 16. We create a SASL profile for SIP by reusing the `WWW-Authenticate` and `Authorization` SIP headers used by the digest authentication, shown earlier. Instead of encapsulating DAA data, we embed SASL messages, as depicted in Fig. 15.

As with the GSS-API, we need to increase the number of messages going back and forth between the SIP client and server. The number of messages depends on the required message exchange needed by the used underlying authentication mechanism.

In the following paragraphs, the numbers in parentheses refer to the SIP message numbers in Fig. 14. The SASL specification only outlines a very high-level method of how the server should advertise its supported mechanisms to the client. We implement the mechanism negotiation in the first three messages in the SIP `REGISTER` handshake (1-4). The UA starts by requesting authentication from the SIP server, with no `Authorization` header (1). The SIP server responds with a `401 Unauthorized` SIP message (3), with the supported and available mechanisms embedded in the `WWW-Authenticate` header:

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: SASL
  negotiate="DIGEST-MD5 NTLM GS2-KRB5"
```

The client selects the best mechanism from the received list that it supports and sends a new SIP `REGISTER` message (4). This message includes an `Authorization` header requesting authentication with “GS2-KRB5” as the preferred mechanism. The initial authentication data is embedded base64 encoded to the *data* parameter:

```
SIP/2.0 REGISTER
Authorization: SASL mechanism="GS2-KRB5",
  data="SUZZT1VDQU5SR...JUPVVQU5FUkQK="
```

The server retrieves the SASL data, and passes the message to the SASL library which handles the authentication. The selected authentication method continues to pass SASL messages between client and server as many times as necessary to

[2]A list of registered SASL mechanisms is maintained by IANA: http://www.iana.org/assignments/sasl-mechanisms/sasl-mechanisms.xml

```
1.    REGISTER sip:CompanyA SIP/2.0
2.    Via: SIP/2.0/UDP
      192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
3.    From: Alice <sip:alice@CompanyA>;tag=1234648905
4.    To: Alice <sip:alice@CompanyA>
5.    Contact: "Alice" <sip:alice@192.168.1.102:5060>
6.    Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
7.    CSeq: 19481 REGISTER
8.    User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
9.    Authorization: Digest
      username="alice",realm="asterisk",nonce="3b7a1395",response=
      "ccbde1c3c129b3dcaa14a4d5e35519d7",uri="sip:CompanyA",
      algorithm=MD5
10.   Max-Forwards: 70
11.   Expires: 3600
12.   Content-Length: 0
```

```
1.    REGISTER sip:CompanyA SIP/2.0
2.    Via: SIP/2.0/UDP
      192.168.1.102;branch=z9hG4bK32F3EC44EB23347BFB0D488459C69E4E
3.    From: Alice <sip:alice@CompanyA>;tag=1234648905
4.    To: Alice <sip:alice@CompanyA>
5.    Contact: "Alice" <sip:alice@192.168.1.102:5060>
6.    Call-ID: 2B6449C74C10D4F95006A6C034E79E8E@CompanyA
7.    CSeq: 19481 REGISTER
8.    User-Agent: PolycomSoundPointIP-SPIP_550-UA/3.1.2.0392
9.    Authorization: SASL mechanism="DIGEST-MD5"
      data="YAzgusSGGeRFGw9nfUvOAxcEDzZCBmKY1H2ElnegaccBcx3DUSkGNW
      Y4qfiSwcXwjLtoqW0eBNog7ixHN"
10.   Max-Forwards: 70
11.   Expires: 3600
12.   Content-Length: 0
```

Fig. 15: A SIP `REGISTER` message with the original DAA `Authorization` header to the left, and the same header carrying SASL data to the right.

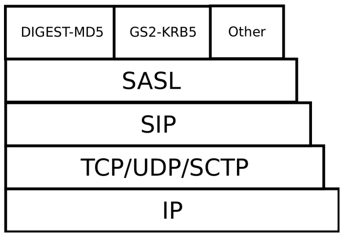


Fig. 16: The SIP SASL stack is similar to the SIP GSS-API stack with underlying security mechanisms.

complete the authentication (messages 5-6 are repeated). Once the authentication is complete, the SIP server sends a `200 OK` SIP message. Should the server have some last SASL data to be communicated to the client to complete the authentication, it can be carried in a `WWW-Authenticate` header embedded in the `200 OK` message (N+2):

```
SIP/2.0 200 OK
WWW-Authenticate: SASL mechanism="GS2-KRB5",
  data="TFoG9rP56zrVH...YaAOndwPew6NdxKr"
```

As soon as the `200 OK` message is received and processed, the client is authenticated to the SIP server. Since the mechanism negotiation is not integrity-protected, the UA is vulnerable to a "down-grade" attack. An attacker can intercept and modify the negotiation messages so that the least favorable authentication method is used.

VII. Migration towards a secure authentication

We propose a two step migration towards a secure authentication in SIP. While our attack on the DAA could be countered by including the SIP header value `ContactURI` in the digest, it did not provide any protection against offline dictionary attacks. We implemented and showed that the use of "Password-Based Key Derivation Function version 2" (PBKDFv2) on the shared secret to make dictionary- and brute-force attacks significant harder to execute on the DAA. However, this method does not authenticate the SIP server, only the client.

Our *first* migration step suggests to replace the DAA with a modified "Password Authenticated Key Exchange" (PAKE) that is more secure than the DAA, introduce mutual authentication and re-use the shared secret used by the DAA. These properties make PAKE a preferred mechanism over the DAA with PBKDFv2. However, using PAKE does not leave any room for future extensions nor modification of authentication in SIP once implemented.

The *second* migration step takes the limitations of the previous mechanisms into consideration, and is seen as the most viable way of solution. The last authentication method introduces support for a GSS-API/SASL security layer which enables SIP to transparently support and use more secure authentication methods in a unified and generic way without the need for later changes to the SIP protocol specification.

Support for the GSS-API/SASL security layer in SIP, have the following attractive properties that address real-world security concerns:

1) Mature, stable and industry adopted standards: The industry might be reluctant to adopt immature and non-standardized security services, like different (new) authentication mechanisms. Both the GSS-API and SASL are stable, mature standards that have been adopted by the industry. Thus, implementing GSS-API or SASL should not be considered a drastic nor radical change by the relevant standardizing bodies (the IETF) nor the VoIP industry.
2) Minimal changes to the SIP standard required: The authentication data re-use the existing SIP DAA headers, so minimal changes to the SIP *message contents* are required. Also, minimal changes are required to the SIP *message flow*, since the authentication handshake is just extended with a number of required SIP message roundtrips to complete the new authentication exchange.
3) Flexible and adaptive to new requirements and future changes: Instead of adding numerous different authentication mechanisms to SIP based on different security requirements, it is desirable to keep the changes to the SIP standard to a minimum. By adding support to a security layer in SIP, adding new or modifying existing underlying authentication mechanisms does not need any redesign of the SIP specification standard. In this case, only the GSS-API/SASL software library needs to be updated.

Thus, authentication in SIP becomes adaptive to future extensions.

VIII. Conclusion and future work

We have seen that the widely deployed authentication method DAA in SIP is weak and vulnerable to attacks. Moreover, we have confirmed and verified that the attack analyzed earlier [25] can be performed on the SIP protocol in real-time. We have examined this authentication method, and proposed a solution to counter the serious registration attack. By including more SIP header parameters in the authentication digest this attack can be countered.

The original SIP designers focused on functionality and compliance at the cost of security. A more thorough investigation of the SIP DAA in the design phase would have revealed the vulnerability presented here, and the vulnerability could have been prevented early on. Our remedy presented here solves a serious problem with the DAA.

Therefore, we wanted to replace DAA with support for an better, more robust authentication scheme. We have added support for a improved authentication mechanism that can easily replace DAA based on a modified PAKE algorithm. This new authentication mechanism adds support for mutual authentication and is more secure than DAA. We have also shown that the modified PAKE authentication can easily function as a drop-in replacement for DAA. However, a more flexible authentication mechanism is desired in the long-term. Different VoIP installations have different security requirements that may require different security services.

We introduced a security programming interface, which provides a security abstraction layer. This abstraction layer adds support to a range of underlying authentication mechanism in a unified way. As long as SIP supports the security layer, new authentication mechanisms can be added later, without requiring any change to the SIP protocol. Support for two security layers were added, the GSS-API and SASL. We recommend the use of SASL, as SASL has more industry deployment, has support for more underlying authentication mechanisms, and is specifically designed for communications protocols.

We envisage a two-step migration towards a stronger authentication scheme in SIP. First, the modified PAKE authentication is implemented and deployed. Second, the long-term solution is to deploy SASL with support for a range of underlying authentication mechanisms.

Future work will look into implementing a proof of concept for PAKE-enabled UA and SIP server, including overhead evaluation benchmarks for the new authentication algorithm. We also plan to evaluate different SASL security mechanisms and their implications for SIP, and decide which authentication mechanisms should be mandatorily supported through SASL.

We plan to co-operate with the IETF and the “kitten” WG to further elaborate GSS-API and SASL support for SIP. We hope our work will gain acceptance and industrial deployment, so that the previously mentioned security attacks can be countered.

Acknowledgment

This research is funded by the EUX2010SEC project in the VERDIKT framework of the Norwegian Research Council (Norges Forskningsråd, project 180054).

References

[1] L. Strand and W. Leister, “Improving SIP authentication,” in *Proceedings of the Tenth International Conference on Networking (ICN2011)*. Xpert Publishing Services, Jan 2011, pp. 164 – 169.

[2] L. Strand, J. Noll, and W. Leister, “Generic security services API authentication support for the session initiation protocol,” in *Proceedings of The Seventh Advanced International Conference on Telecommunications (AICT2011)*. Xpert Publishing Services, Mar 2011, pp. 117 – 122.

[3] L. Strand, W. Leister, and A. Duric, “Migration towards a more secure authentication in the session initiation protocol,” in *The Fifth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE2011)*. Xpert Publishing Services, Aug 2011.

[4] “Det norske markedet for elektroniske kommunikasjonstjenester 2009 (The Norwegian market for electronic communication services 2009),” Post- og teletilsynet (The Norwegian Post and Telecommunications Authority), 2010. [Online]. Available: http://www.npt.no/ikbViewer/Content/119027/Ekomrapport_2009_.pdf [Accessed: 1. Jul 2011]

[5] Telecommunication Development Sector (ITU-D), “The world in 2010,” ITU-T ICT facts and figures, 2010.

[6] L. Strand and W. Leister, “A Survey of SIP Peering,” in *NATO ASI - Architects of secure Networks (ASIGE10)*, May 2010.

[7] International Telecommunication Union, “7 kHz Audio-Coding within 64 kbits/s,” ITU-T Recommendation G.722, 1993.

[8] 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, updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt [Accessed: 1. Jul 2011]

[9] 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, updated by RFCs 5506, 5761, 6051, 6222. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt [Accessed: 1. Jul 2011]

[10] H. Sinnreich and A. B. Johnston, *Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., August 2006.

[11] H. Dwivedi, *Hacking VoIP: Protocols, Attacks, and Countermeasures*, 1st ed. No Starch Press, Mar. 2009.

[12] D. Endler and M. Collier, *Hacking Exposed VoIP: Voice over IP Security Secrets and Solutions*. McGraw-Hill Osborne Media, November 2006.

[13] A. M. Hagalisletto and L. Strand, “Designing attacks on SIP call set-up,” *International Journal of Applied Cryptography*, vol. 2, no. 1, pp. 13–22(10), July 2010.

[14] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne, *SIP Security*. WileyBlackwell, Mar. 2009.

[15] VoIPSA, “VoIP security and privacy threat taxonomy,” Public Realease 1.0, Oct. 2005. [Online]. Available: http://voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf [Accessed: 1. Nov 2011]

[16] D. York, *Seven Deadliest Unified Communications Attacks*. Syngress, Apr. 2010.

[17] P. Park, *Voice over IP Security*. Cisco Press, Sep. 2008.

[18] S. Salsano, L. Veltri, and D. Papalilo, “SIP security issues: The SIP authentication procedure and its processing load,” *Network, IEEE*, vol. 16, pp. 38–44, 2002.

[19] D. Kuhn, “Sources of failure in the public switched telephone network,” *Computer*, vol. 30, pp. 31–36, 1997.

[20] A. D. Keromytis, *Voice over IP Security - A Comprehensive Survey of Vulnerabilities and Academic Research*, 1st ed. New York, NY: Springer New York, 2011, vol. 1.

[21] International Telecommunication Union (ITU), “Security Architecture For Open Systems Interconnection (OSI),” The International Telegraph and Telephone Consultative Comittee (CCITT), X.800 Standard X.800, 1991.

[22] "Research project: EUX2010SEC – Enterprise Unified Exchange Security"." [Online]. Available: http://www.nr.no/pages/dart/project_flyer_eux2010sec [Accessed: 1. Nov 2011]

[23] L. Fritsch, A.-K. Groven, L. Strand, W. Leister, and A. M. Hagalisletto, "A Holistic Approach to Open Source VoIP Security: Results from the EUX2010SEC Project," *International Journal on Advances in Security*, no. 2&3, pp. 129–141, 2009.

[24] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617 (Draft Standard), Internet Engineering Task Force, Jun. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2617.txt [Accessed: 1. Jul 2011]

[25] A. M. Hagalisletto and L. Strand, "Formal modeling of authentication in SIP registration," in *Second International Conference on Emerging Security Information, Systems and Technologies SECURWARE '08*. IEEE Computer Society, August 2008, pp. 16–21.

[26] International Organization for Standardization and ISO, "ISO/IEC 11770-4:2006: Information technology – Security techniques – Key management – Part 4: Mechanisms based on weak secrets," 2006.

[27] J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1," RFC 2743 (Proposed Standard), Internet Engineering Task Force, Jan. 2000, updated by RFC 5554. [Online]. Available: http://www.ietf.org/rfc/rfc2743.txt [Accessed: 1. Jul 2011]

[28] A. Melnikov and K. Zeilenga, "Simple Authentication and Security Layer (SASL)," RFC 4422 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4422.txt [Accessed: 1. Jul 2011]

[29] J. Undery, "Ieft draft: SIP authentication: SIP digest access authentication," IETF, Tech. Rep., Jul. 2001.

[30] C. Yang, R. Wang, and W. Liu, "Secure authentication scheme for session initiation protocol," *Computers & Security*, vol. 24, no. 5, pp. 381–386, Aug. 2005.

[31] J. Peterson, "S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)," RFC 3853 (Proposed Standard), Internet Engineering Task Force, Jul. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3853.txt [Accessed: 1. Jul 2011]

[32] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt [Accessed: 1. Jul 2011]

[33] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," RFC 3325 (Informational), Internet Engineering Task Force, Nov. 2002, updated by RFC 5876. [Online]. Available: http://www.ietf.org/rfc/rfc3325.txt [Accessed: 1. Jul 2011]

[34] J. Peterson and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," RFC 4474 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4474.txt [Accessed: 1. Jul 2011]

[35] F. Palmieri, "Improving authentication in voice over IP infrastructures," in *Advances in Computer, Information, and Systems Sciences, and Engineering*, K. Elleithy, T. Sobh, A. Mahmood, M. Iskander, and M. Karim, Eds. Springer Netherlands, 2006, pp. 289 – 296.

[36] F. Palmieri and U. Fiore, "Providing true end-to-end security in converged voice over IP infrastructures," *Computers & Security*, vol. 28, no. 6, pp. 433–449, Sep. 2009.

[37] Y. Liao and S. Wang, "A new secure password authenticated key agreement scheme for SIP using self-certified public keys on elliptic curves," *Computer Communications*, vol. 33, no. 3, pp. 372–380, Feb. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366409002631 [Accessed: 1. Jul 2011]

[38] Y. Liao, "Secure password authenticated key exchange protocols for various enviroments," Ph.D. dissertation, Tatung University, Dec. 2009.

[39] International Telecommunication Union, "H.323 security: Framework for security in H-series (H.323 and other H.245-based) multimedia systems," ITU-T Recommendation H.235.0, 2005.

[40] ——, "H.323 security: Framework for secure authentication in RAS using weak shared secrets," ITU-T Recommendation H.235.5, 2005.

[41] M. Spencer, B. Capouch, E. Guy, F. Miller, and K. Shumard, "IAX: Inter-Asterisk eXchange Version 2," RFC 5456 (Informational), Internet Engineering Task Force, Feb. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5456.txt [Accessed: 1. Jul 2011]

[42] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321 (Informational), Internet Engineering Task Force, Apr. 1992, updated by RFC 6151. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt [Accessed: 1. Jul 2011]

[43] L. Strand, "VoIP lab as a research tool in the EUX2010SEC project," Norwegian Computing Center, Department of Applied Research in Information Technology, Tech. Rep. DART/08/10, April 2010.

[44] "Asterisk: The Open Source PBX & Telephony Platform." [Online]. Available: http://www.asterisk.org/ [Accessed: 1. Nov 2011]

[45] "NetSED: The network packet stream editor." [Online]. Available: http://silicone.homelinux.org/projects/netsed/ [Accessed: 1. Nov 2011]

[46] X. Wang and H. Yu, "How to break MD5 and other hash functions," *IN EUROCRYPT*, vol. 3494, 2005.

[47] P. Hawkes, M. Paddon, and G. G. Rose, "Musings on the wang et al. md5 collision," Cryptology ePrint Archive, Report 2004/64, 2004.

[48] D. Eastlake 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174 (Informational), Internet Engineering Task Force, Sep. 2001, updated by RFCs 4634, 6234. [Online]. Available: http://www.ietf.org/rfc/rfc3174.txt [Accessed: 1. Jul 2011]

[49] "Twisted Matrix Labs." [Online]. Available: http://twistedmatrix.com [Accessed: 1. Nov 2011]

[50] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0," RFC 2898 (Informational), Internet Engineering Task Force, Sep. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2898.txt [Accessed: 1. Jul 2011]

[51] S. Turner and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," RFC 6151 (Informational), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6151.txt [Accessed: 1. Jul 2011]

[52] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104 (Informational), Internet Engineering Task Force, Feb. 1997, updated by RFC 6151. [Online]. Available: http://www.ietf.org/rfc/rfc2104.txt [Accessed: 1. Jul 2011]

[53] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings," RFC 4648 (Proposed Standard), Internet Engineering Task Force, Oct. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4648.txt [Accessed: 1. Jul 2011]

[54] Y. Oiwa, H. Watanabe, and H. Takagi, "Pake-based mutual http authentication for preventing phishing attacks," *CoRR*, vol. abs/0911.5230, 2009. [Online]. Available: http://arxiv.org/abs/0911.5230 [Accessed: 1. Jul 2011]

[55] Y. Oiwa, H. Watanabe, H. Takagi, Y. Ioku, and T. Hayashi, "Mutual Authentication Protocol for HTTP," Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://tools.ietf.org/html/draft-oiwa-http-mutualauth-08 [Accessed: 1. Jul 2011]

[56] T. Kivinen and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)," RFC 3526 (Proposed Standard), Internet Engineering Task Force, May 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3526.txt [Accessed: 1. Jul 2011]

[57] D. Todorov, *Mechanics of User Identification and Authentication: Fundamentals of Identity Management*, 1st ed. Auerbach Publication, Jun. 2007.

[58] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," RFC 4121 (Proposed Standard), Internet Engineering Task Force, Jul. 2005, updated by RFC 6112. [Online]. Available: http://www.ietf.org/rfc/rfc4121.txt [Accessed: 1. Jul 2011]

[59] C. Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)," RFC 2025 (Proposed Standard), Internet Engineering Task Force, Oct. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc2025.txt [Accessed: 1. Jul 2011]

[60] L. Zhu, P. Leach, K. Jaganathan, and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism," RFC 4178 (Proposed Standard), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4178.txt [Accessed: 1. Jul 2011]

[61] S. Josefsson and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family," RFC 5801 (Proposed Standard), Internet Engineering Task Force, Jul. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5801.txt [Accessed: 1. Jul 2011]