

Secure on-line configuration for SIP UAs

Master thesis literature study

By: Erik Ehrlund
Email: eehrlund@kth.se

Examiner:
Professor Björn Pehrson

Supervisors:
Doctor Jon-Olov Vatn
Phd student Erik Eliasson

Abstract

This literature study report is part of the master thesis project “Secure on-line configuration for SIP UAs”¹. The report will present and try to evaluate some different techniques for secure authentication and downloading of contact lists primarily based on a user's password. It will start by giving a broad overview of the problem that is to be solved and then moves on to different protocols that solves parts of the problem. The report will then end with case studies of techniques already in use and a conclusion.

The result of this literature study will be the base for further work and implementation in the SIP user agent (SIP UA) *minisip*² driven by Erik Eliasson and Jon-Olov Vatn at KTH.

1 Secure online configuration for SIP UAs, <http://www.tslab.ssvl.kth.se/thesisprojects/2006/eehrlund/>

2 Minisip, www.minisip.org

Table of Contents

Abstract.....	2
1. Introduction.....	4
1.1 Background.....	4
1.2 Overview of master thesis	4
2. Authentication and key agreement protocols	5
2.1 Current Protocols used on the Internet.....	5
2.2 Strong Password Protocols.....	6
2.2.1 DH - Diffie-Hellman	7
2.2.2 EKE - Encrypted Key Exchange.....	8
2.2.3 SPEKE - Simple Password-authenticated Exponential Key Exchange	9
2.2.4 SRP - Secure Remote Password.....	10
2.2.5 PDM - Password Derived Moduli	11
2.2.6 Other strong password protocols	11
2.2.6.1 AMP - Authentication and key agreement via Memorable Passwords.....	11
2.2.6.2 Snapi - Secure Network Authentication with Password Identification.....	11
2.2.6.3 AuthA.....	11
2.2.7 Conclusions on Strong Password Protocols.....	12
2.3 SASL - Simple Authentication and Security Layer.....	12
2.4 CHAP protocols.....	13
2.5 Conclusion, Authentication and key agreement protocols.....	13
3. Framework	14
3.1 SACRED - Securely available credentials.....	14
4. Related work.....	16
4.1 Jabber.....	16
4.2 WebDAV.....	17
5. Conclusions.....	18
5.1 Step 1. Authentication.....	18
5.1.1 Implementation.....	19
5.2 Step 2. Credential management.....	19
5.2.1 Implementation	20
5.3 Goals of Master Thesis.....	20
5.3.1 Primary Goals.....	20
5.3.2 Secondary Goals.....	20
5.2 Design and Implementation.....	21
5.2.1 Scenario	21
5.2.1.1 Actors	22
5.2.1.2 Sacred	23
5.2.1.3 Certificate management with SRP.....	24
5.2.2 Limitations	26
5.2.3 Implementation	26
6. References.....	27
7. Abbreviations	31

1. Introduction

1.1 Background

The Voice over IP market is growing and the success of Skype is a clear indication of that. Skype claims to have over 100 million users[1] and is still growing heavily. One can argue much around what have made Skype so successful. One contributing factor is its close relationship and appearance of a regular IM (Instant Messaging) system, which users are very familiar with. This makes the “transition” from regular IMs to Skype easier and more people are willing to change. Another contributing factor is the fact that they were one of the first to offer free device to device calls (as long as both devices were connected to the Internet and ran Skype).

Skype uses a proprietary method for the signalling and media transfer which many, especially companies, dislikes. There is a standard that specifies the control signals and the rest of the protocols needed for voice over IP (RFC 3261, 2327 and 3550). These standards have been constructed and agreed upon to ensure interoperability between different software phones and companies. A problem with these standards is that they, in their basic form, do not emphasise the security, which is quite important when dealing with computer networks. The reason why the security is more important in a computer networks is that its much easier to “capture” the traffic and listen in to what people are talking about in a computer network than in PSTN (Public Switched Telephone Network)[3].

1.2 Overview of master thesis

There are many SIP UAs out on the market and *minisip* is just one out of many. What differentiate *minisip* from the rest of the SIP UAs is that it concentrates heavily on security. It tries to solve the issues by authenticating the parties involved in the communication and encrypting the traffic between two UAs and the signalling between the UA and the registrar. This is primarily done with asymmetric keys[2] (public/private keypair) that is stored locally on the device (*minisip* can also use shared secrets). These keys are used in different ways to ensure the privacy and confidentiality of the users. One way they are used is by signing Diffie-Hellman parameters to protect them from the man-in-the-middle attack (more on Diffie-Hellman in section 2.2.1).

Having the keys stored locally on the device that uses them works good if every user only have one device or knows how to move the keys about. This limits the users, as he either *physically* needs to carry the keys around, in for example a *smart card*, or *remember* the keys (which is not likely as most of them are completely random and over 1024 bits long) in order to use a new device.

Another problem is introduced when there is more than one user of a particular device. Different users needs different keypairs (different certificates so that they can be identified correctly by others) and different settings (which registrar to contact, phone book etc.) .

This master thesis will investigate and try to implement a system where a user instead of carrying the information with him can sign into a service, that is available over the Internet, and download his credentials and configuration files by only remembering a password and a user name (alice@kth.se) .

2. Authentication and key agreement protocols

The first and the biggest problem that this master thesis is trying to solve is how to securely download a user's credentials. The need for the credentials was shortly explained in section 1.2 "overview of master thesis" in this report. In this section different protocols to download a user's credentials will be discussed and evaluated.

There has been a lot of research in this area by the standardisation organisations. All the major standardisation organisations have or have had a group working with this problem or problems related to it (IEEE P1363[4], IETF SACRED wg[5] and ISO/IEC 11770-4[6]). The IETF working group defined requirements on a protocol that downloads a user's credentials and it also defined a framework and a protocol called SACRED (Securely Available Credentials) which is published in RFCs 3157, 3760 and 3767. IEEE P1363 are working more generally and are trying to standardise a couple of protocols for authentication and key agreement for use on the Internet.

The problem of downloading the credentials and doing it without giving away any useful information to an attacker can be divided into three smaller problems: how to authenticate the server, how to authenticate the user and how to set up a shared strong secret so that the channel can be encrypted. All these problems are not new and are in fact the very basics of secure communication over an insecure medium (prove who you are talking to and setting up a virtual secure channel via for example encryption). Once these problems are solved the server can send the credentials unencrypted (the channel is encrypted so its only the server and the clients that can read and write on it) or preferable encrypted with the users passwords so that the server do not have full access to the users credentials (also adds some security if the server is somehow compromised).

2.1 Current Protocols used on the Internet

As mentioned in the previous section the problem of mutual authentication and key agreement is not a new one and in fact there are a lot of applications using different protocols to achieve this on the Internet today, for example banking applications. Most of the current applications relies on SSL (Secure Socket Layer) or the new version of it TLS (Transport Layer Security)[7]. Both SSL and the new TLS relies heavily on a the PKI (Public Key Infrastructure) that is implemented today to achieve authentication of the parties (especially the server as the client can authenticate itself via for example passwords if it trusts the server).

There are many difficulties with PKIs and to understand them a deeper knowledge and understanding of how public/private key encryption and certificates works is needed [8].

The PKI that is implemented today is the hierarchical, meaning that the certificates should be signed by a trust anchor. A very popular one is VeriSign Trust Network which most of the Swedish banks use to sign their certificates.

One of the problems with PKIs comes from its design. How can one ensure trust in a company that signs yours and others certificates? There has been problems even with the most popular trust anchors. [9] describes how VeriSign accidentally issued a certificates to a false user claiming to be an employee of Microsoft. One way of solving this problem is to create your own Certificate Authority, CA. This works well after you have configured your device with the public key of the CA, but there is no trivial way to do this "on the fly" (download the public key it automatically). The problem is not that the public key should be kept secret, the problem is that one can not be 100% sure that someone has not changed the public key to their public key before it is downloaded to the device.

Some other problems with PKIs are described in [10] and in [11]. One of the problems that is explained is how software today identifies a secured connection. A browser does this by displaying a lock and the identity of the owner of the certificate somewhere on the screen. This is to symbolise that the connection is secured. It is easy to fool a user with this by giving false names in the certificate.

As the goal of this master thesis is to enable a client to be downloaded to any device and then retrieve its credentials from a server, that the software do not know anything about. The regular way of doing it, with PKI, is not enough. This is because of the problem with getting the credential servers public key.

2.2 Strong Password Protocols

"Strong password protocols are designed so that someone who eavesdrops on authentication exchange, or someone impersonating either end will not obtain enough information to do off-line verification of password guesses." *Network Security, Private communication in a public world* [12].

The above description describes exactly what we are looking for in a authentication protocol. *Strong password protocols* are designed to not give enough information for verification of passwords in an off-line brute force attack (for examples dictionary attacks as the protocols are dealing with weak secrets). If the protocol is designed correctly this should leave the attacker with only one way to verify passwords guesses and that is to query the server (which is an audible on-line attack that the server can deal with in appropriate manner). The biggest drawback of Strong password protocols is the IPR (Intellectual Property Rights) status situation which will be explained more in the conclusion of this subsection.

Almost all strong password protocols depend, one way or another, on the old Diffie-Hellman protocol. To get a good understanding of the protocols, this subsection will first give an overview of Diffie-Hellman and then briefly explain the most popular Strong password protocols. The subsection will end with some conclusions.

2.2.1 DH - Diffie-Hellman

Diffie-Hellman is an old protocol dating all the way back to 1978 when Martin E. Hellman and Whitfield published their publication[13] (later Ralph C. Merkle was credited for his contributions by Hellman and Diffie). Their protocol builds on the assumption that it is computationally infeasible to solve a discrete logarithm problem. If we use old Alice and Bob to represent client and server, the algorithm works as follows:

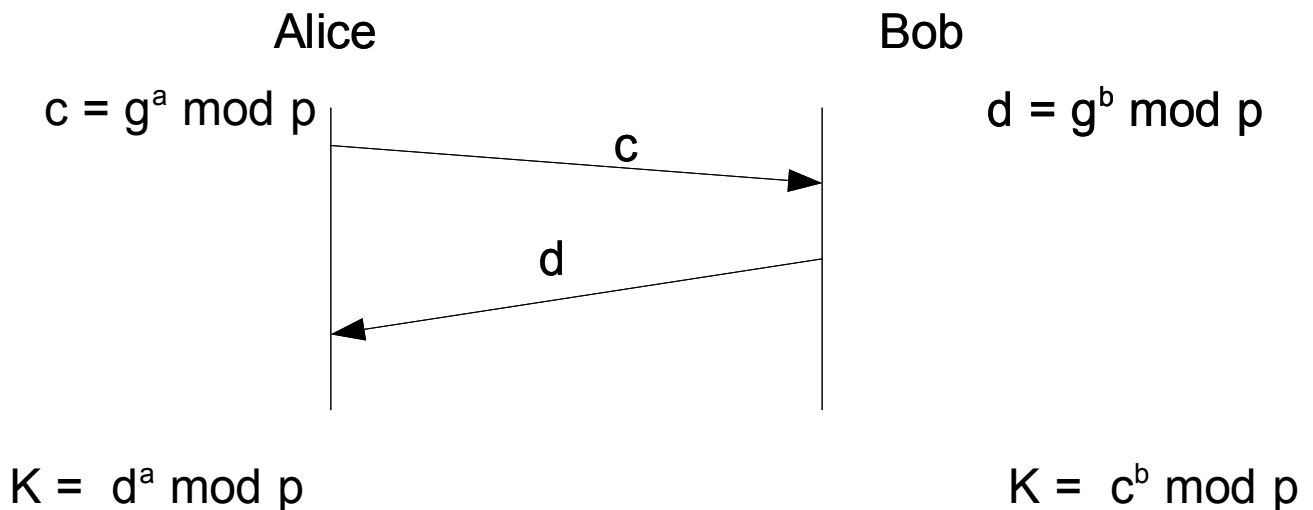


Figure 1. Shows the Diffie-Hellman exchange and establishment of session key

Before the above exchange happens Alice and Bob has somehow agreed upon g and p . p and g does not need to be kept secret and can be public. For a more in detail description see [13][15][16]. The Diffie-Hellman protocol is used in many different protocols for example the Strong password protocols and [14]. The DH protocol protects the messages from a passive attacker [17] where an attacker only listens to the messages but do not alter them in any way, but it offers little protection against active attackers (where the attacker not only listens but inserts messages to) and especially the attack called “man-in-the-middle”[18]. The patent on this protocol, that was filed in 1978, has expired making DH free of charge and has enabled researchers to use the primitives to construct more advanced protocols to protect against the “man in the middle attack”.

2.2.2 EKE - Encrypted Key Exchange

The EKE protocol was the first strong password protocol. It was designed by Bellare and Merritt and presented during the Proceedings of IEEE “Symposium on Research in Security and Privacy” May 1992 in Oakland. The protocol suggested to make use of passwords to achieve mutual authentication and the establishment of a session key. There are many variants of EKE but the most basic and popular is the DH-EKE, which modified the Diffie-Hellman protocol presented in 2.2.1 to protect it against the “man in the middle attack”. The protocol works as follows:

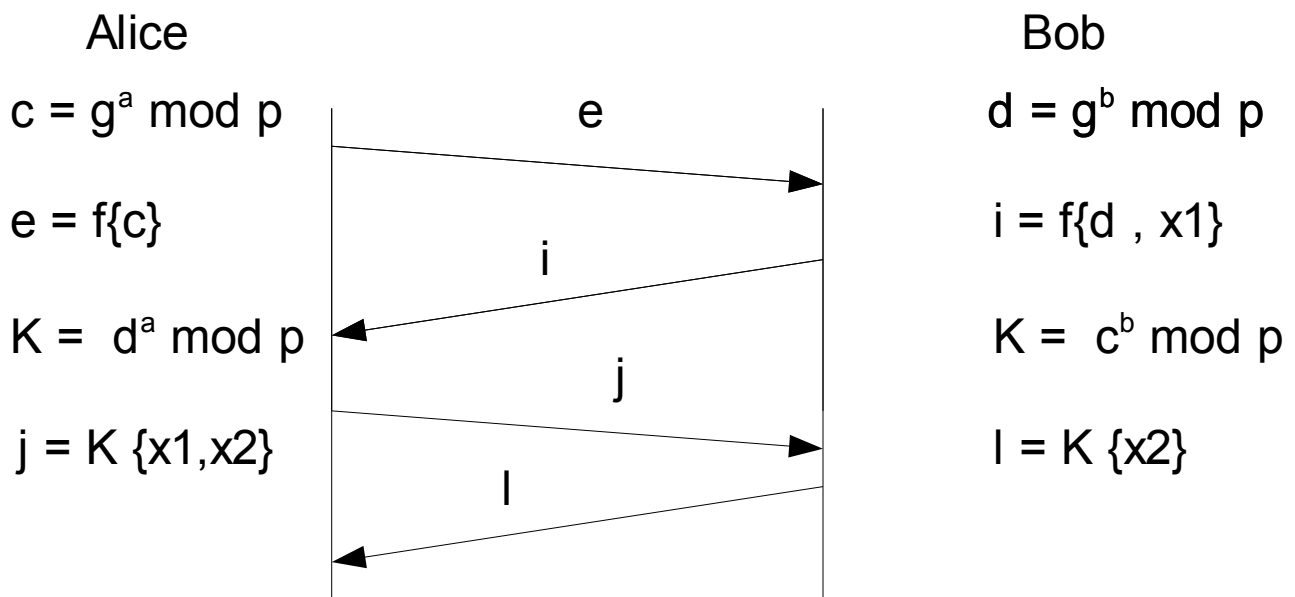


Figure 2. Shows the EKE session key exchange and authentication of both parties.

$f(x)$ = some function with x as input

$K\{x\}$ = x encrypted with K

The only real difference between this protocol and DH is that instead of sending c and d in plaintext, the values are encrypted using a function that depends on the shared password. What this small change achieves is that only Alice and Bob can see and create the values that are sent making it impossible for a “man in the middle” to change the values in an undetectable way. For a more exact description of the values and steps in this protocol see *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks* [19]. The protocol described in [19] uses clear text passwords which was later corrected in a new version called A-EKE (Augmented EKE) [20]. One should be careful with implementing EKE as some of its variants have been proven insecure by Sarval Patel [33].

EKE is patented under US patent 5,241,599 [21] and requires explicit permission or a license from AT&T Bell Laboratories in order to implement it.

2.2.3 SPEKE - Simple Password-authenticated Exponential Key Exchange

The SPEKE protocol was developed after EKE and was designed by David Jablon in 1996[22]. The two protocols are very similar as both uses passwords to alter the messages sent and received in DH. Instead of encrypting the first message sent from Alice and the first from Bob with a function that depends on the password, SPEKE uses a function to alter the base g in a way dependent on the password as shown in figure 3 below.

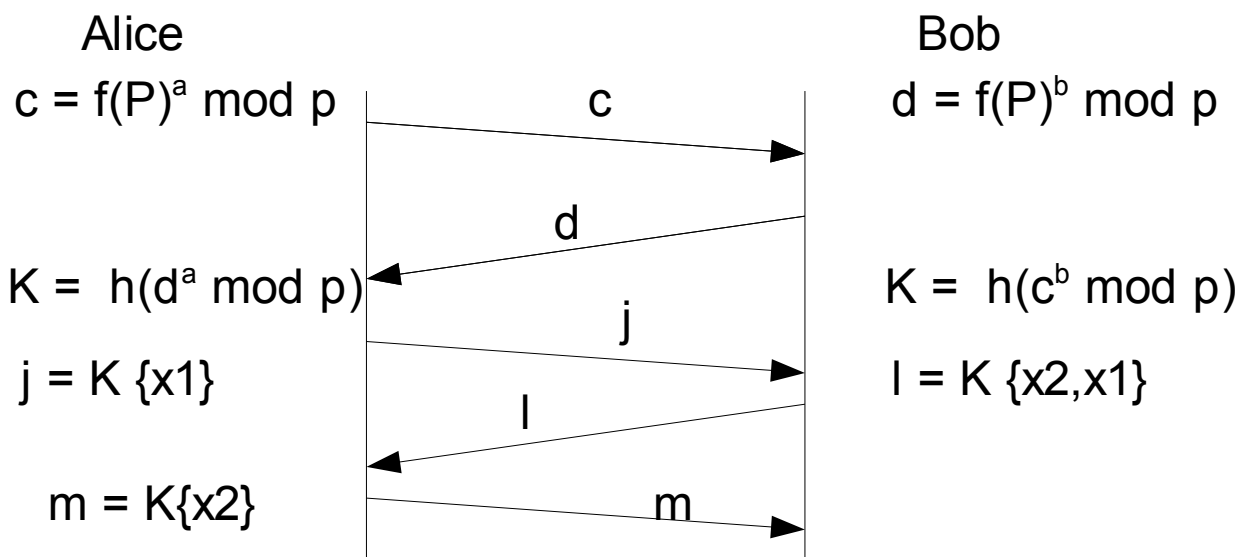


Figure 3. Shows how SPEKE authenticates and establishes a strong shared secret.

$h(x)$ = hash of x

$f(x)$ = some function with x as input

$K\{x\}$ = x encrypted with K

As one quickly sees the difference between EKE and SPEKE is very small but they have completely different patents and IPR status. SPEKE shares the strength of EKE but achieves them in a different way.

SPEKE is covered by US patent 6,226,383 [23] filed by Integrity Sciences, Inc (license now belongs to Phoenix Technologies).

2.2.4 SRP - Secure Remote Password

Around the same time as Jablon was working on SPEKE, Thomas Wu (in some places called Tom Wu) worked on another protocol called SRP, Secure Remote Password. SRP is the strong password protocol that have received most attention and has in fact been standardised by IETF in RFC 2945[24]. SRP has been revised many times and are now up in revision 6.

Although SRP differs the most from DH of the protocols mentioned earlier in this document, the basic is still the same with a modified regular DH exchange. To get an understanding on how SRP works see figure 4.

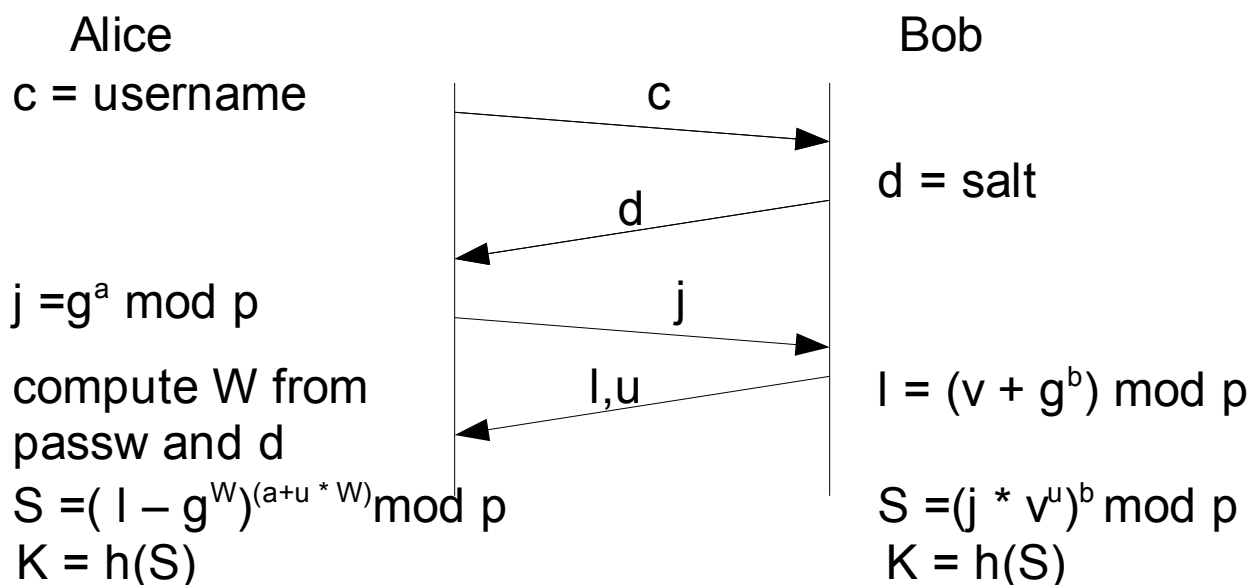


Figure 4. Shows roughly how SRP-3 works.

$v = \text{verifier stored at server. it is dependent on password}$

$h(S) = \text{hash of } S$

$u = \text{some random number publicly revealed}$

After the above messages both sides proves that they have derived the same encryption key by some simple challenge response messages (this also authenticates both sides as they are the only ones being able to deduce this key). The SRP exchange is a bit more complicated to follow than EKE and SPEKE and it (as mentioned) has been revised 6 times. To get a deeper knowledge of SRP-3 RFC 2945 [24] or [25] are good (SRP-6 is explained in [26]).

SRP is patented by Stanford university [27] but instead of selling the license, they offer it under a royalty-free license [28] and are offering source code and much other information on their website³. There are some concerns with the patents that will be discussed later in section 2.2.7 "Conclusions on Strong password protocols"

³ Stanford University, The Stanford SRP Authentication Project. <http://srp.stanford.edu/>

2.2.5 PDM - Password Derived Moduli

PDM is another protocol for key agreement and authentication that was developed by Radia Perlman and Charlie Kaufman [29]. PDM is designed to circumvent the patents in place but no one has tested it in court so the Internet society remain sceptic against it. If it was shown that it in fact does not infringe on patents in place, PDM would be unpatented and free to implement and use. PDM is also designed to not be so computational intensive on the server as the rest of the protocols. Instead it is the client that must do the heavy work. This is not a nice property for *minisip* as *minisip* is used on weaker machines like PDAs. It is said in [45] that it takes approximately 10 seconds to generate a 500-bits safe prime (which is needed for the exchange) on a 400 Mhz processor. 400 Mhz is a common speed on PDAs today. Perlman and Kaufman also suggest a method to speed up generation of even larger primes with good performance. One could argue that the credential download only takes place one time and only in the start-up phase of the phone which makes it less time critical (if a user wants to update his or hers password, and similar actions, after the credentials have been downloaded the public/private key pair could be used) but users would get annoyed with long start-up times. PDM is described in [29].

2.2.6 Other strong password protocols

There are many variants of strong password protocols but the ones presented above are the most popular ones. To not limit this report to only contain strong password protocols this subsection will present in shorter form the rest of the protocols considered for standardisation by the IEEE P1363 workgroup and give references to more information for the interested reader. That these protocols are explained shorter than the ones above does not exclude them from the conclusions, it is just that the protocols are very similar (modify one or more parameters of the DH exchange).

2.2.6.1 AMP - Authentication and key agreement via Memorable Passwords

The AMP protocol was invented by Taekyoung Kwon. AMP is similar to the other protocols in the sense that it uses DH to establish "perfect forward secrecy" and agree upon a session key [30]. The AMP protocol is said to have no patent restrictions by Kwon [31] but some feels that it may infringe on the same patents as SRP (this will be discussed in more detail in section 2.2.7).

2.2.6.2 Snapi - Secure Network Authentication with Password Identification

In 1999 Philip Mackenzie and Ram Swaminathan invented Snapi that was later extended to Snapi-X which secures Snapi from server compromises [32](it hashes the password so a dictionary attack on the stolen information is still possible). Snapi is owned by Lucent Technologies and is the first protocol that is "provable" secure but it is somewhat slower than SRP.

2.2.6.3 AuthA

Is very similar to the DH-EKE exchange with a minor changes. It was designed by Mihir Bellare and Philip Rogaway in March 14 2000[34]. As it is so closely related to EKE it probably falls under the EKE patent and those requires permission from Lucent to implement it.

2.2.7 Conclusions on Strong Password Protocols

All strong password protocols have similar security strengths as they all build upon DH (there are concerns about some of the EKE implementations [33]). The biggest problem that all “strong password protocols” suffers from are insecure IPRs and patents (not an uncommon thing in the security industry[37]). The situation has gone so far that IETF have chosen weaker algorithms in some of their standards (for example SACRED which will be presented in section 3.1) just to avoid the problem. The foundation of the problem are these two IPR claims [35] which makes the situation unclear, do they have claim over the SRP patent ? (in case of yes, they probably have claim over the other protocols as well). Stanford university do have a patent on SRP (as mentioned in section 2.2.4) but the other patents where issued before SRP, so if its somehow shown that SRP infringe on patents [23] or [21], the SRP patent would be invalidated. So far (to my knowledge) noone has been sued on this matter but that is not an evidence that noone ever will be. One could go to court to settle this but such trials are expensive and the outcome can go either way. IETF have asked both Lucent and Phoenix to make their claims clearer but both companies are unwilling to do so. Many believe it is because both companies have their own framework for strong password protocols which they wishes to sell (including Thomas Wu the creator of SRP who expresses his opinions in a mailing list for openssh unix[36]). Lucent and Phoenix probably reasons that if they can create uncertainty about the other protocols more companies will choose their frameworks instead.

If this matter is settled I think we will see Strong password protocols take over many parts of Internet authentication as it is simple yet very secure (securer than certificates which have many problems mentioned in section 2.1).

2.3 SASL - Simple Authentication and Security Layer

“The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms” *Simple Authentication and Security Layer (SASL), RFC 4422*

SASL is another IETF standard which is designed to take care of the authentication between communicating parties. SASL is designed, to the extent possible, to abstract away the authentication from different programs and methods so that programmers do not need to redo this in every program. This is done by “inserting” an authentication layer which runs below the application layer. What this layer does is that it works like an API for the application layer to provide authentication capabilities. This makes it possible to change the actual authentication method used independently of the program in question (a truth with modifications as different authentication mechanisms require different information to authenticate the other party). SASL also gives programs the abilities to negotiate which authentication mechanism to be used “on the fly”.

The work on SASL is ongoing [38] and the latest RFC is 4422 which describes the SASL functionality and specifies one out of many methods to achieve authentication (external authentication by for example TLS).

2.4 CHAP protocols

Some of the SASLs mechanisms to authenticate the parties are simple challenge-response protocols. Challenge-response protocols are the most basic way to authenticate. What this means is that the server sends a challenge to the client which the client must correctly answer in order to be authenticated. The most basic challenge-response protocol is where the server just asks for first the username and then the password which is sent in clear text to the server. This is not a secure protocol if the channel is not secured. There are many variants of CHAP protocols and they achieve the same result, client or mutual authentication. Most CHAP protocols are vulnerable to both active and passive attacks (passive by listening in on the traffic and then do a dictionary attack against it to crack the password[17], active by actively changing some of the packets that are transferred or just act as a man-in-the-middle[18]). The way most implementers choose to make this secure is by doing a TLS handshake before the CHAP begins. What this does is that it reassures the client that it is talking to the correct server (by looking at the server certificate) and also provide means for the client to send the challenges encrypted. This requires the server to have a valid certificate and the client to have a way to validate the certificate. Another way to make CHAP protocols more secure is to do anonymous authentication with for example DH to establish a “secure” channel. Protecting the channel with a DH established key protects the information sent back and forth from a passive attack but not against active attackers that intercepts and modifies the DH exchange. This document will not go into details on CHAP protocols as there are so many and they have almost the same performance. Some examples are MS-CHAP v1/v2(RFC 2433 / 2759), digest authentication (RFC 2617) .

2.5 Conclusion, Authentication and key agreement protocols

The securest way to do authentication that I have found is to use one of the "strong password protocols" presented in section 2.2 (not the ones broken by [33]). The reason why these protocols are more secure is that they don't give away any information for an attacker to work with. They modify random numbers in a predictable way by using the password as a secret. If an attacker tries to decrypt these messages (with the help of for example a dictionary) he or she will see an output of random numbers and have no way to verify that it is the “right” random numbers. The protocols also establishes a shared strong secret after the initial authentication. This secret is protected even if the password is somehow later cracked as DH enables perfect forward secrecy. The downside of “strong password protocols” are, as mentioned earlier, their unclear IPR status (which is a pretty big downside as an implementer may be sued).

CHAP and certificates are not completely insecure but they suffer from some drawbacks, certificates have the problem of how to retrieve the initial certificates and CHAP protocols tend to leak information that an attacker can use (some hash of the password which can be used by an attacker to verify password guesses against). CHAP protocols can be strengthened with an anonymous DH before the actual exchange starts. This protects it against passive attackers but not active.

The SASL framework specifies many different mechanisms to achieve authentication, some are GSSAPI, EXTERNAL, ANONYMOUS and DIGEST-MD5. They are variants of different CHAP protocols.

3. Framework

This section will present a framework that was built by the IETF Sacred wg to take care of download and management of credentials.

3.1 SACRED - Securely available credentials

The IETF Sacred wg [39] started its work in the early 2000 to create a protocol for secure download of credentials (private keys, passwords and so on). The Sacred workgroup has finished all their milestones and the workgroup was closed in February 2006. It was closed although many of its participants was worried about that it would be just another standard that never gets implemented because of the required certificates in the authentication procedure.

The Sacred framework is specified in three different RFCs, RFC 3157, 3760 and 3767. RFC 3157 defines a set of requirements on the protocol while 3760 gives a framework. RFC 3157 is an implementation of the Sacred protocol. The implementation defines a set of mandatory and optional operations for both account management and credential management.

The optional operations are:

- Information request - Request information about values for account creation
- Create Account - Creates an account
- Remove Account - Deletes an account
- Modify Account - Modifies an account

The must operations are:

- Credential Upload
- Credential Download
- Credential Delete

The way IETF has encoded these operations are with XML encoding.

The protocol further specifies a series of implementation decisions. BEEP (RFC 3080) has been selected as the protocol to carry the information because it has built in support for TLS and SASL. BEEP is like HTTP an application transport protocol. The difference is that BEEP is more general and a little more advanced than HTTP. It for example supports multiple “channels” inside a single TCP connections so that for example FTPs instruction channel and the data channel could be realised over the same TCP connection (instead of two different connections). To be Sacred compliant IETF have chosen that servers and clients must support DIGEST-MD5 over TLS with server authentication in order to be compliant with RFC 3157.

There has been a lot of discussion whether or not to change the standard to use http and that the authentication mechanisms should be changed to one of the strong password protocols. The workgroup decided, against many participants wills, to not include http as the change is not that big and to not implement strong password protocols because of their unclear IPR statuses.

4. Related work

There are a lot of different software out on the market that uses different ways to do authentication. This section looks closer into two protocols designed by IETF and how they do the authentication and retrieval of phone book and other user specific information.

4.1 Jabber

Jabber is an instant messaging software that is built and maintained by “Jabber software foundation”⁴. It is open source and the underlying protocols that it uses was approved for standardization by the IETF XMPP wg in October 2004 (RFCs 3920 and 3921). What differs Jabber from other instant messaging protocols is that it is completely decentralised and the servers are not controlled by any organisation. Anyone that wants can download the software and host their own Jabber server. This gives Jabber large flexibility and companies can use the Jabber software for their internal communication (they control the whole network from client to server).

The most interesting parts of Jabber, for this document and this master thesis, is it's security features. That is, how does it authenticate it's users and how does the client retrieve his or hers phone book (in XMPP called roster).

In short, Jabber relies heavily on IETF standards, so for the authentication and security parts it has implemented SASL and TLS. SASL has briefly been explained in section 2.3 and TLS is mainly used to set up a secure channel between the two communicating parties but it can also be used in the SASL EXTERNAL method to authenticate both parties (for example by using certificates). There is still some compatibility issues between old and newer clients and servers which has lead to that some Jabber servers still allow plain text or message digest login.

The users phone book (in jabber called roster) is stored and kept on the server. When a user is authenticated it downloads a copy of the phone book but the original is kept on the server. This is due to how Jabber exchanges “presence” information about other users i.e. displaying his on-line or off-line status. The servers keeps a record for each contact and in that record it saves subscription information (for example whether or not a user want to know the status of another user). All changes made to the local copy of the phone book must propagate to the server in order to have affect.

The exact implementation details are hard to give as there are many variants of the Jabber client and server software.

4 Jabber Software foundation, <http://www.jabber.org/>

4.2 WebDAV

WebDAV is a standardisation for remote document handling using HTTP. It is specially designed to support multiple users that collaborate to finish a document or web resource[40]. The WebDAV wg has produced many RFCs but this master thesis is only interested in how WebDAV achieves authentication (the user information should only be accessible by the user).

The WebDAV wg is not very clear on how the actual authentication should take place but it states that compliant software must support the digest authentication scheme RFC 2617 which offers some protection for the password.

5. Conclusions

This report has given an overview of some ways to authenticate a client to a server and it has also given an introduction to a frameworks that manages credentials. It has focused much on Strong password protocols and the problems with the IPRs has been presented. What has been left out is the *minisips* needs and what's actually going to be implemented in it. This section will make some conclusions based on the information presented above and give an outline for development and implementation in *minisip*.

To securely download something 2 steps needs to be taken. The first step is authentication and the second step is the actual download/upload of the requested information. The implementation will have to execute in this order to provide good security. Each step will be presented in section 5.1.

The basic needs for *minisip* and the reason for this master thesis was briefly presented in section 1.2. The credentials that is downloaded will be used in for example MIKEY[41] and or SRTP to provide extended security. This will further be discussed in more detail in section 5.2.

5.1 Step 1. Authentication

The first step must be the authentication phase, where the client and server authenticate each other. Mutual authentication is required for most operations as the client needs to know that he gets the right credentials (or upload it to the correct server) and the server needs to know that it sends the credentials to the right person (and not someone trying to attack the system). During this authentication phase a secret key should be negotiated for extra security (so that the credentials is not sent in clear text as they are vulnerable to dictionary attacks more on this in step 2).

There is essentially 2 completely different ways to provide the authentication based on the information given in this report. The two methods are simple CHAP and the little more advanced (and secure) strong password protocols. If one only looks on security it is very clear that one of the strong password protocols are the best choice, as they give no information at all to an attacker and they set up an encrypted channel, but if the complete picture is looked at the choice is not that clear any more, because of the IPRs.

I should mention that I am no lawyer and I have no training in legal and patent issues but I think (as Tom Wu in[36]) that the IPRs are used to obstruct the development of other Strong password protocols in order to gain market shares. That said, if one of the strong password protocols are chosen the best one probably is SRP, as it has gone through a lot of scrutiny and it is in fact a standard in both IETF and IEEE. There is also a patent covering SRP which is owned by Stanford University (they offer an royalty free license [28]). This reassures, to some extent, that the techniques used are different from the ones patented by Lucent and Phoenix (although cross patents are known to have happened so this gives no absolute guarantees).

If the risks are deemed to high one can always fall back on the CHAP protocols. The CHAP protocols are less secure than strong password techniques. CHAP protocols are susceptible for both passive and active attackers. The CHAP protocols can be enhanced by doing an anonymous DH. This protects the exchange from passive attackers.

5.1.1 Implementation

There is a lot of ways to implement authentication based on the suggestions above. For example one can just implement the specific algorithms and then be done with it. Another way to do it is to use the IETF standard SASL. The SASL framework is very flexible and can be used with different variants of CHAP and one of the Strong password protocols namely SRP (thanks to the method EXTERNAL and TLS-SRP[42]).

There are a lot of free libraries that offers the abilities to perform authentication. For SASL there are for example GNU-SASL and Cyrus-SASL which both are open source and RFC compliant. To enhance these with the capabilities of anonymous DH and TLS-SRP, GNU-TLS can be used (which is already a component of *minisip*). There are also a large collection of other libraries that can perform different CHAP authentications.

5.2 Step 2. Credential management

The next step is the step where the client operates on its credential(s) that is stored on the server. To provide more security the server should not have the credentials in plain text rather it should be encrypted in some way that only the client (acting on behalf of the user) can decrypt (with the help of for example a password). This protects the credentials from a server compromise and do in fact enable the information to be sent over an unencrypted channel. If the information is sent over an unencrypted channel the security will be lower as passwords are weak secrets and are most often susceptible for dictionary attacks (but if a CHAP protocol has been used for authentication an attacker will already have information to perform a dictionary attack on).

The only standards for credential management that I have been able to find is IETF Sacred. The Sacred framework was presented in section 3. The Sacred framework is also mentioned in some other IETF drafts related to SIP [43]. The implementation of Sacred will therefore enable *minisip* to support these standards.

5.2.1 Implementation

The Sacred protocol do not have any cryptographic in it self and relies heavily on the underlying protocols to take care of authentication and the set up of a secure channel. In order to be compliant to RFC 3157 an implementation is forced clearly answer the questions in section 3.1 of that RFC.

I have only been able to track down one implementation of the Sacred protocol which is the SourceForge Sacred project⁵ and have only heard rumours that there is an academic implementation. The Sacred project has released an open source version of Sacred implemented in JAVA which is the wrong language for *minisip* (*minisip* is implemented in C and C++).

5.3 Goals of Master Thesis

This master thesis will have a couple of primary and secondary goals. This subsection will be updated after the meeting with my supervisors.

5.3.1 Primary Goals

- Implement a credential downloading system (for example Sacred with SASL)
- Secure download of some user specific configuration files (phone book , etc)[44]

5.3.2 Secondary Goals

- Update the configuration files so they are compliant to a standard

5 SourceForge.net Sacred Project. <http://sourceforge.net/projects/sacred/>

5.2 Design and Implementation

The problems presented in section 1.1 *Overview of master thesis* are quite straight forward if looked briefly upon but gets more and more complex as the full problem is understood. In a few words, this thesis is trying to accomplish secure download of some information without any prior knowledge of the server (except the address of it and your own user name and password). The solution should follow some standardisation so that *minisip* can be as interoperable as possible. The main motivation for this master thesis comes from the problems of mobility. Today if I want to use a SIP phone, for example *minisip*, on a computer or PDA for the first time I need to manually transfer everything that I need from my previous UA or start to set up everything from scratch. This section will try to give more details on the problem by giving a scenario and propose two ways it can be solved.

5.2.1 Scenario

A user Alice travels around the world a lot and do not like to pay the extra fee that is required if she calls from abroad with regular telephone services. Because of this she has started to use IP telephony to have the possibility to move around without changing her telephone number and lowering her costs. She has two different “phone numbers”, one that she uses for calls to other SIP client and another that she can use for calls to regular phones. Alice is interested in security and knows that calls over computer networks can easily be recorded and eavesdropped upon and uses because of this clients that can handle SRTP, MIKEY and other security protocols. As Alice travels a lot she uses different computers and PDAs quite often and do not want to reconfigure everything every time, especially she do not want to bring her long term credentials, that is required for her secure calls, or phone book with her (with all her contacts). Instead she wants this to be downloaded, in a secure way, into the new UA when she signs on (like jabber rosters or msn contacts).

In the scenario above we can see that the problem is that Alice wants to be able to use any computer or PDA without having to add all her contacts and manually enter the long term credential. This situation could also occur inside a company where a user resides in multiple rooms and likes to have the same phone book in all of them. The scenario also introduces the possibility that a user have multiple accounts that he or she wishes to use simultaneous. This raises the question if more than one account should be managed by one server or if every account needs its own credential and configuration server. There are also some design decisions to be made if one server should provide both the configuration and the credentials or if separate servers should be used (so that other ways to retrieve the credentials could be used).

This document has presented several different authentication schemes and a framework for credential management. In order to securely download something mutual authentication is required so a authentication scheme has to be implemented, but the functionality after authentication could look very different depending on implementation.

5.2.1.1 Actors

It is not entirely clear how an implementation should handle that a user has multiple accounts and therefore also have the possibility to have multiple credentials. Should the user have one credential server that contain all his credentials, even those that do not relate to the account(s) managed by that service provider?, for example Alice have two accounts (alice@example.com and alice@example.org) which has different certificates. Should Alice store both her certificates on alice@example.com or should she store the certificates on each server respectively. Another option would be that she stored the information on a third server not related to any of her accounts (maybe operated by her company or herself). Maybe the most intuitive solution would be to have her retrieve her credentials from each account server but this make the set up phase a bit longer (need to contact each server in turn to start everything). Another disadvantage to bind the credentials to be retrieved from the operator of an account would be that maybe Alice do not trust this operator enough to handle her credentials (the operator can always do some kind of attack on the information stored as it have to depend on weak secrets) but she trust it to route her traffic (otherwise she should chose some other operator).

Another question that was presented in previous section was if one server should manage both credential and user configuration download. The benefit of having only one server is that it simplifies the start up phase as the UA only needs to contact two servers (the credential and user configuration server and the registrar). It would even be possible in the future to merge these two servers as well. This would be an even bigger advantage if one had to retrieve each accounts information separately (with multiple accounts and different servers for credentials and configuration files $3x$ servers would need to be contacted, where x is number of accounts). The biggest disadvantage to have the one server store all information is that it limits its use, one would have to handle special cases if credentials was managed some other way (smart cards or stored locally).

To summaries this section, 4 different ways to structure the implementation has been shown, one server per account or all accounts in one place and one server takes cares of both credentials and user configuration or separate servers.

5.2.1.2 Sacred

As mentioned before Sacred is a framework for credential management. The Sacred protocol is not SIP specific and can be used by any service that needs to retrieve and store credentials remotely. Sacred accomplishes this by including a “credential selector” in the download request message (and upload message). This enables a client to select the information it wants if it knows the selector. To explain how Sacred works the scenario in section 5.2.1 will be solved by using Sacred.

1. Alice downloads and installs *minisip* on the computer or PDA that is currently available for her.
2. Alice starts *minisip* and gets asked for a Sacred server address*
3. *minisip* connects to the server and sends its greeting.
4. The Sacred server also sends a greeting containing what profiles it supports (SASL, sacred and TLS)
5. *minisip* chooses a way to authenticate itself and sets up a TLS secured channel to the server**
6. Alice is authenticated and can start using the Sacred protocol to download her credentials by sending download requests with the appropriate selectors***

* If the Sacred server is administrated by one of Alice's operators the address could be taken from Alice user name (alice@example.com)

** Not necessary in that order or even separate (both is achieved simultaneous with TLS-SRP)

*** Here is a small problem. How does *minisip* know the appropriate selector ? One way this could be solved is to use “Alice” as the selector but Alice might have more credentials than the ones used for SIP on the specified server. Another way to solve this could be by using “SIP” as selector which could ensure that only SIP phone credentials are downloaded. This would work but would not work well. Lets say that Alice have more accounts as in the scenario. Would the selector “SIP” give her both credentials ? What if she only wants to go on-line on one of them ? There are two intuitive solutions to this problem. The first one is that Alice uses “SIP” as selector and downloads every credential under it but she gets asked which accounts she wants to go on-line on. The second one would have Alice type in the selectors manually but one problem here is that she might not remember all accounts so the Sacred protocol would have to be extended with a “list credentials” command.

Alice's configuration files could be stored with the credentials but that is unintuitive and would make changes to the information stored frequent (not an unsolvable problem but would require more logic in the Sacred server or that *minisip* sends the updates bundled which would send an unnecessary large amount of traffic). A better choice for configuration management with Sacred is to have a separate configuration server (which also would enable the UA to follow the draft [44]).

This would add a couple of steps.

7. *minisip* asks Alice for address of profile server and sends a SUBSCRIBE to the user profile server.****
8. *minisip* creates a TLS session with the user profile server to authenticate the server and sends a SUBSCRIBE *****
9. the profile server answers with a content redirect message inside a NOTIFY message (or sends the profile inside the NOTIFY message if its small enough) The redirect message can point to any service for downloading information, for example https.
10. *minisip* connects and authenticates itself to the content server (using for example https) and sends a GET message on the profile.
11. the content server delivers the content to *minisip*

**** This information could be present somewhere in the credentials.

***** *minisip* can here use any authentication means even TLS certificates as her root certificates where downloaded earlier.

The biggest advantage with implementing Sacred is that the software would follow the Sacred standard and those be able to use any Sacred server (not only the operators). This would enable Alice to have all her credentials stored in one place even the ones not related to Internet telephony.

One disadvantage the Sacred protocol has is that is so general and have because of this some overhead.

5.2.1.3 Certificate management with SRP

Another solution to the problem described in section 5.2.1 *Scenario* is to implement a new protocol for downloading of the credentials. What this protocol should do it to first authenticate both side and set up a TLS tunnel. After this the credentials should be downloaded somehow. It could follow the Sacred framework but not the way the actual protocol works. This is what the IETF SIP wg has done in a draft for certificate management[43]. They propose to use SIP messages over a secured tunnel to upload (PUBLISH) and download (SUBSCRIBE) certificates. An example is show in the figure below which taken from the draft [43]. The figure shows how Bob first uploads is credentials and then at a later time retrieves it.

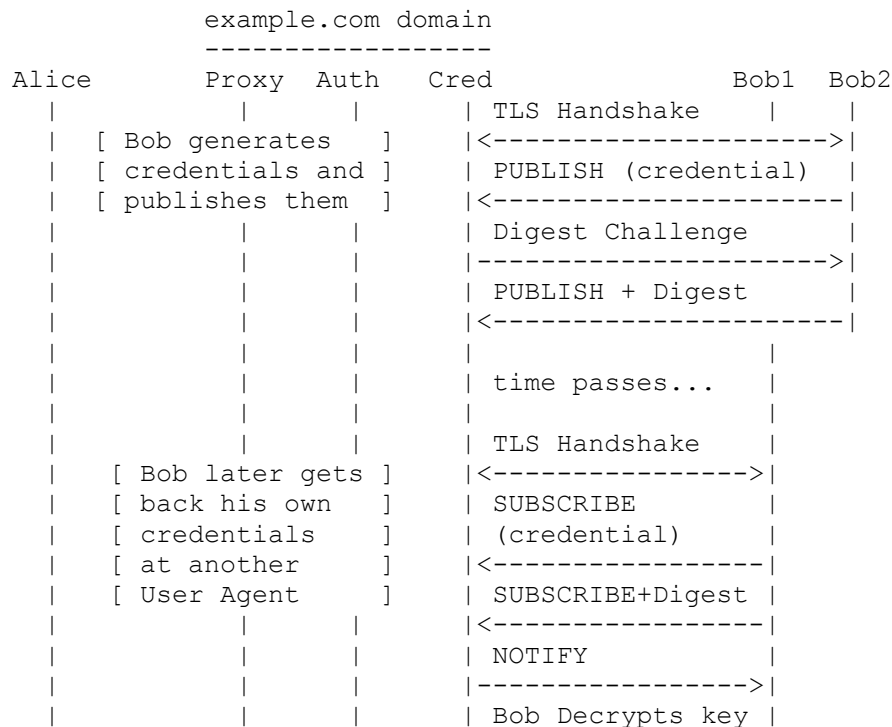


Figure 5 shows a proposed way for managing credentials by SIP wg.

More details and complete figure can be found in [43].

The way shown in figure 5 requires certificates to be used in the first TLS handshake (Bob authenticates Cred via certificates). Certificates had some problems that was discussed in section 2.1 and it was especially difficult in the start up phase.

Instead of doing a TLS handshake based on certificates one could use the TLS draft [42] to remove the need for certificates and also remove the digest challenge completely (the user gets authenticated in the TLS handshake).

The scenario 5.2.1 is solved with this protocol below.

1. Alice downloads and installs *minisip* on the computer or PDA that is currently available for her.
2. Alice starts *minisip* and gets asked for credential server address *
3. *minisip* connects to the credential server and does a TLS-SRP negotiation to authenticate itself and the server. **
4. *minisip* sends a SUBSCRIBE message to the credential server inside the TLS tunnel
5. the credential server responds with a NOTIFY message containing the credentials

* Could by default be extracted from the login name `alice@example.com`

** SASL could be used to provide more than one way to authenticate.

From the way the SIP wg designed the protocol its quite clear that they intend it to be used on a “per account” basis where Alice needs to do download each credential separately. So to have multiple accounts the procedure above would have to be repeated for every account.

To get the user configuration one could either include it in the credential (was discussed in previous section) or use the draft [44] which would add messages 7-11 from the previous section.

5.2.2 Limitations

All on-line configurations suffers from one very big disadvantage. A user needs to trust the software and the system that the UA is running on. The whole security falls if the computer or PDA is somehow compromised with for example a key logger. If it is compromised an attacker could easily get the password and those all the credentials and configuration files. The only way around this problem is to have the credentials in for example a *smart card* so that the system can not directly read the content, just interact with it.

5.2.3 Implementation

Both Sacred and certificate management with SRP are very much alike which was shown in the previous sections. Both solves the same problem in a similar manner. The biggest difference between them is that Sacred is a more general protocol than certificate management with SRP which also makes Sacred a little more heavy. The generalization is Sacred's biggest advantage over certificate management with SRP as it would enable a user to use any Sacred server to store and download his or hers credentials (this advantage comes only if Sacred servers are deployed outside the scope of SIP).

Because of the limitations presented in 5.2.2 it would be nice to have the user configuration and credentials management separate as this would enable *minisip* to in the future implement *smart cards* and still be able to download the user configuration (it would also allow locally stored credentials to be used). To support both *smart card* and *credential downloading* implementations the authentication mechanism for the user configuration server should be one that both can easily use (certificates or other methods). If possible the [44] draft should be followed to make the profile delivery as interoperable as possible.

6. References

- [1] Skype (2006). [www] http://about.skype.com/corporate_factsheet.pdf. Last access on 2006-06-17
- [2] Wikipedia. [www] http://en.wikipedia.org/wiki/Public_key. Last access on 2006-07-03
- [3] Per Lövgren, PC för alla (5/2006) *Varning! Din ip-telefon kan vara avlyssnad* . [www] http://www.idg.se/ArticlePages/200606/21/20060621104752_IDG.se466/20060621104752_IDG.se466.dbp.asp . Last access on 2006-07-03.
- [4] IEEE P1363. [www] <http://grouper.ieee.org/groups/1363/> . Last access on 2006-07-04
- [5] IETF Sacred wg. [www]. <http://www3.ietf.org/proceedings/05aug/sacred.html> . Last access on 2006-07-04
- [6] ISO/IEC 11770-4 (2006-05-04). [www] <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39723&ICS1=35&ICS2=40&ICS3=> . Last access 2006-07-04
- [7] Wikipedia. [www] http://en.wikipedia.org/wiki/Secure_Sockets_Layer . Last access on 2006-07-04
- [8] Sun Microsystems, Inc *Public Key Infrastructure Overview* (August 2001). [www] <http://www.sun.com/blueprints/0801/publickey.pdf> . Last access on 2006-07-04
- [9] Sans Institute, [www] http://www.sans.org/reading_room/papers/download.php?id=679&c=80c9dec08d4b581d738caceb89082798&portal=cf77f3b3df1b54180fe3e589b302e283. Last access on 2006-07-07
- [10] KPMG, [www] http://www.us.kpmg.com/RutUS_prod/Documents/12/DC80502.pdf. Last access on 2006-07-07
- [11] Netcraft LTD. [www] http://news.netcraft.com/archives/2004/03/08/ssls_credibility_as_phishing_defense_is_tested.html . Last access on 2006-07-07
- [12] Charlie Kaufman, Radia Perlman, Mike Speciner. *Network Security, Private communication in a public world*. Second edition (2002). Chapter 12. ISBN: 0-13-046019-2
- [13] *New Directions in Cryptography*, W. Diffie and M. Hellman, [www] <http://www-ee.stanford.edu/~hellman/publications/24.pdf> . Last access on 2006-07-10
- [14] IETF, *Diffie-Hellman Key Agreement Method* (1999). [www] <http://www.ietf.org/rfc/rfc2631.txt>. Last access on 2006-07-10
- [15] Wikipedia. [www] <http://en.wikipedia.org/wiki/Diffie-Hellman>. Last access on 2006-07-10

- [16] Kieth Palmgren, *Diffie-Hellman Key Exchange - A Non-Mathematician's Explanation* [www] <http://www.securitydocs.com/library/2978> . Last access on 2006-07-10
- [17] Wikipedia. [www] http://en.wikipedia.org/wiki/Passive_attack . Last access on 2006-07-10
- [18] Wikipedia. [www] http://en.wikipedia.org/wiki/Man_in_the_middle_attack . Last access on 2006-07-10
- [19] S. Bellovin M. Merritt, *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks* (may 1992). [www] <http://citeseer.ifi.unizh.ch/bellovin92encrypted.html> .Last access on 2006-07-11
- [20] S.Bellovin M. Merritt, *Augmented Encrypted Key Exchange: a Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise* (1993) [www] <http://citeseer.ifi.unizh.ch/bellovin93augmented.html> . Last access on 2006-07-11
- [21] AT&T Bell Laboratories, *Cryptographic protocol for secure communications*, US patent 5,241,599 (August 31, 1993). [www] <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=5,241,599>
- [22] D. Jablon, *Strong Password-Only Authenticated Key Exchange* (september 25 1996) [www] <http://www.jablon.org/jab96.pdf> . Last access on 2006-07-12
- [23] Integrity Sciences, Inc. *Cryptographic methods for remote authentication*, US patent 6,226,383 (March 25 1997) . [www] <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6,226,383> . Last access on 2006-07-12
- [24] IETF RFC 2945, *The SRP Authentication and Key Exchange System* (September 2002). [www] <http://www.ietf.org/rfc/rfc2945.txt> . Last access on 2006-07-13
- [25] Thomas Wu, *The Secure Remote Password Protocol* (November 11, 1997) [www] <http://srp.stanford.edu/ndss.html> .Last access on 2006-07-13
- [26] Thomas Wu, *SRP-6: Improvements and Refinements to the Secure Remote Password Protocol* (October 29, 2002). [www] <http://srp.stanford.edu/doc.html> . Last access on 2006-07-13
- [27] The Board of Trustees of the Leland Stanford Junior University, *System and method for securely logging onto a remotely located computer*, US patent 6,539,479 (March 25, 2003) [www] <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6,539,479> . Last access on 2006-07-13
- [28] Standford University, SRP License. [www] <http://srp.stanford.edu/license.txt> .Last access on 2006-07-13
- [29] Radia Perlman Charlie Kaufman, *Secure Password-Based Protocol for Downloading a Private Key*. [www] <http://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/perlman.pdf> . Last access on 2006-07-13

- [30] Taekyoung Kwon *Ultimate Solution to Authentication via Memorable Password* (May 2000). [www] <http://grouper.ieee.org/groups/1363/passwdPK/contributions.html#amp> . Last access on 2006-07-18
- [31] Taekyoung Kwon, *IEEE P1363.2 – AMP* (March 2004) [www] http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/p1363_200403.ppt
Last access on 2006-07-18
- [32] Philip MacKenzie and Ram Swaminathan, *Secure Network Authentication with Password Identification* (August 1999). [www] <http://grouper.ieee.org/groups/1363/passwdPK/contributions.html#MS>
- [33] Sarval Patel, *Number Theoretic Attacks On Secure Password Schemes* (1997) [www] <http://citeseer.ist.psu.edu/patel97number.html> . Last access on 2006-07-18
- [34] Mihir Bellare and Phillip Rogaway, *The AuthA Protocol for Password-Based Authenticated Key Exchange* (March 2000). [www] <http://grouper.ieee.org/groups/1363/passwdPK/contributions.html#autha>
Last access on 2006-07-18
- [35] IETF IPR status pages, SRP. [www] <http://www.ietf.org/ietf/IPR/LUCENT-SRP> and <http://www.ietf.org/ietf/IPR/PHOENIX-SRP-RFC2945.txt> . Last access on 2006-07-19
- [36] mailing list openssl-unix-dev. [www] <http://marc.theaimsgroup.com/?l=openssl-unix-dev&w=2&r=1&s=srp&q=b>
Last access on 2006-07-19
- [37] Robert Moskowitz, *Patents problems* (March 4, 2002). [www] <http://www.networkcomputing.com/1305/1305colmoskowitz.html> .Last access on 2006-07-19
- [38] IETF SASL wg. [www] <http://www.ietf.org/html.charters/sasl-charter.html> .Last access on 2006-07-24
- [39] IETF Sacred wg. [www] <http://www1.ietf.org/html.charters/OLD/sacred-charter.html>
Last access on 2006-07-28
- [40] IETF WebDAV wg. [www] <http://www.ietf.org/html.charters/webdav-charter.html> Last access on 2006-07-29
- [41] J. Bilien, *Key Agreement for secure Voice over IP*, Master Thesis, KTH, December 2003.[www] <ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/031215-Johan-Bilien-report-final-with-cover.pdf> Last access on 2006-07-30.
- [42] IETF TLS wg (June 26, 2006). [www] <http://www.ietf.org/internet-drafts/draft-ietf-tls-srp-12.txt>. Last access on 2006-07-30
- [43] IETF SIP wg (June 25, 2006). [www] <http://www.ietf.org/internet-drafts/draft-ietf-sip-certs-01.txt> Last access on 2006-07-30

- [44] IETF SIPPING wg (Mar 6, 2006). [www]
<http://tools.ietf.org/wg/sipping/draft-ietf-sipping-config-framework/draft-ietf-sipping-config-framework-08.txt> . Last access on 2006-07-30
- [45] Charlie Kaufman Radia Perlman, *PDM: A New Strong Password-Based Protocol*, *In proceedings of 10th USENIX Security Symposium, USENIX Association* (August 17, 2001)[www] http://www.usenix.org/events/sec01/full_papers/kaufman/kaufman.pdf
Last access on 2006-08-09

7. Abbreviations

UA - User Agent

SIP - Session Initiation Protocol

IP - Internet Protocol

IM - Instant Messaging

RFC - Request for comments

PSTN - Public Switched Telephone Network

IEEE - Institute of Electrical and Electronics Engineers

IETF - Internet Engineering Task Force

ISO/IEC - International Standards Organization/International Electrotechnical Commission

SSL - Secure Socket Layer

TLS - Transport Layer Security

PKI - Public Key Infrastructure

CA - Certificate Authority

IPR - Intellectual Property Rights

DH - Diffie-Hellman

EKE - Encrypted Key Exchange

SPEKE - Simple Password-authenticated Exponential Key Exchange

SRP - Secure Remote Password

PDM - Password Derived Moduli

AMP - Authentication and key agreement via Memorable Passwords

Snapi - Secure Network Authentication with Password Identification

SASL - Simple Authentication and Security Layer

CHAP - Challenge-Handshake Authentication Protocol

SACRED - Securely Available CREDENTIALS

wg - Work group