WIRELESS AUTHENTICATION USING REMOTE PASSWORDS

by

Andrew Harding

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Andrew Harding

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Kent E. Seamons, Chair
Date	Daniel Zappala
Date	Eric K. Ringger

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Andrew Harding in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

Parris Egbert
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

WIRELESS AUTHENTICATION USING REMOTE PASSWORDS

Andrew Harding

Department of Computer Science

Master of Science

Current authentication methods for wireless networks are difficult to maintain. They often rely on globally shared secrets or heavyweight public-key infrastructure. Wireless Authentication using Remote Passwords (WARP) mitigates authentication woes by providing usable mechanisms for both administrators and end-users. Administrators grant access by simply adding users' personal messaging identifiers (e.g., email addresses, IM handles, cell phone numbers) to an access control list. There is no need to store passwords or other account information. Users simply prove ownership of their authorized identifier to obtain wireless access.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Kent Seamons, for his many hours of direction and feedback. I would also like to thank Tim van der Horst for his many contributions and editing help. I would also like to thank the other members of my committee for their feedback.

This research was supported by funding from the National Science Foundation under grant no. CCR-0325951, prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

Table of Contents

List of Tables

List of Figures

Chapter 1 — Introduction

Wireless networks provide an attractive means for network access. They allow for convenient roaming and device deployability without the burden of network cables and port accessibility; access to wireless networks is bounded only by the limitations of wireless radios.

Although convenient, this allows anyone with a radio antenna to passively eavesdrop network communication, and anyone with a transmitter to inject or modify packets. For these reasons, it is paramount that wireless networks provide secure authentication to prevent unauthorized access to network resources and provide confidentiality and integrity to transmitted information.

The original security mechanism standardized for 802.11 networks is Wired Equivalent Privacy (WEP) [6]. WEP has proven to be insecure and several attacks targeting its vulnerabilities have rendered it relatively worthless. Since WEP has been broken there has been significant effort to create alternative authentication mechanisms.

These new mechanisms are hard to configure, use, and maintain. They are usually based on PKI, global passphrases, or username/password pairs (see Section 7). These methods are either too heavy or inflexible and lack general support for environments with a dynamic user-base, such as corporations or universities where delegation and guest access are frequent. Many authentication systems face these same problems: complexity, rigidity and poor maintainability.

Wireless Authentication using Remote Passwords (WARP) mitigates wireless authentication problems by building on Simple Authentication for the Web (SAW) [10]. WARP provides manageable and usable wireless authentication. By proving

ownership of an authorized personal messaging identifier, a user authenticates without pre-established secrets or the heavy cost and inconvenience of PKI. The burden of wireless access control is dramatically reduced by WARP, where the only needed information is a personal messaging identifier (e.g., email address, IM handle, cell phone number).

The following are some motivating scenarios to promote the deployment of WARP:

- **Home User** Johnny, a security minded user, is anxious to set up a wireless network at home. Johnny is wary of current "home" solutions that use global passphrases. He is a member of a linux users group and often holds meetings in his residence. Johnny dislikes having to share his passphrase with his guests. He also does not have the resources or time to setup and maintain the infrastructure required for EAP-TLS [2] or MSCHAPv2 [13]. He wants to provide secure wireless connectivity with the least amount of inconvenience for himself and his guests.

- **Conference Committee** A week long conference is being held at a university. It is desired that all participants have wireless connectivity during the conference. Obtaining user accounts for each participant through the IT department is time consuming, and the task of distributing usernames and passwords is unwieldy. Certificate-based methods are also undesirable; the burden of obtaining a certificate is considered a waste of the attendees' time. Additionally, the conference staff have enough preparations without adding the hassle of configuring the wireless network. Forgotten username/password pairs or improperly configured certificates could spell disaster for the staff on the first day of the conference, and they want to avoid wasting time that could

otherwise be used for gainful participation in the conference. In short, they want to spend minimal time managing the wireless connectivity and reduce the number of steps participants need to take (and potentially do wrong) in order to authenticate.

- **Corporation Environment** Various employees from another company are visiting daily. Until this point, the company has tied the wireless authentication server to the user accounts used for regular computer and network access. Because of the tight integration, IT staff have had no extra burden managing wireless access; creation/deletion of the user accounts for a new/terminated employee automatically grants/revokes wireless access privileges. But now, without access to the collaborating company's user accounts, they have been forced to add temporary users every time someone visits.

The rest of the paper is organized as follows. Section 2.1 discusses SAW, which represents a major building block for the design of WARP. Section 3 introduces WARP, its design goals, and how it solves the wireless authentication problem. Section 4 describes the development of the prototype WARP implementation. A detailed threat analysis of the system is given in Section 5 and Section 6 discusses deployability. Related work follows in Section 7 and Section 8 concludes with research contributions and future work.

Chapter 2 — Background

WARP provides wireless authentication by building on a website authentication protocol called SAW. This section presents an overview of SAW. It also introduces a chicken-and-egg problem created when trying to authenticate to a wireless network using a protocol that depends on authenticating with a party that is only reachable through network connectivity. SRP, another building block of WARP, is presented in Section 3.1.

2.1 SAW

Simple Authentication for the Web (SAW) [10] leverages personal messaging (e.g., email, text and instant messages) to eliminate user-specific passwords at web sites. SAW significantly improves the basic technique employed by the "Forgot your password?" link common to many web sites by off-loading user authentication to unmodified email providers.

Figure 2.1 contains a diagram of SAW. Authentication is performed by users successfully retrieving two short-lived, single-use *Authentication Tokens*. These tokens ($AuthToken_{user}$ and $AuthToken_{email}$) are created by using a conventional secret splitting scheme to divide a single secret ($AuthToken_{complete}$). $AuthToken_{user}$ is returned directly to the user over the secure link used to initiate the authentication (e.g., HTTPS), while $AuthToken_{email}$ is emailed. If the user returns both tokens then the authentication is successful. Since $AuthToken_{user}$ is returned over a secure link, passively observing $AuthToken_{email}$ is worthless.

SAW provides many advantages over typical website authentication. For example, users log in to websites using credentials they are already familiar with. Because the number of usernames and passwords required to remember is reduced, users and

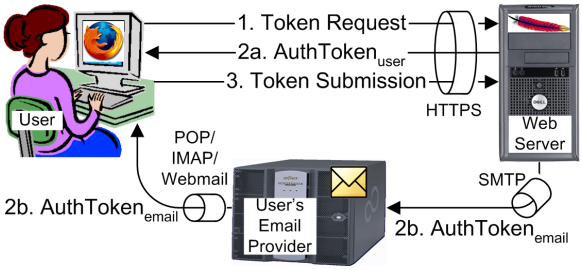

Figure 2.1: The SAW protocol. Based on the user's email address, submitted in (1), a server distributes two authentication tokens. AuthToken$_{user}$ (2a) is returned directly to the user while AuthToken$_{pm}$ (2b) is emailed. Both tokens must be returned (3) to successfully authenticate. Each login attempt involves its own unique, short-lived, single-use tokens.

---

administrators will less frequently be burdened with resettting passwords. Access control is also simplified to an access control list of authorized personal messaging identifiers. Account management is also completely off-loaded to personal messaging providers for sites that only use accounts for access control.

Token submission in WARP will be accomplished by submitting proof of token possession as opposed to the tokens themselves.

SAW is subject to an active impersonation attack. By submitting a victim's email address to a site, an attacker obtains an $AuthToken_{user}$. Consequently, by observing the victim's unencrypted email traffic, the attacker acquires the associated $AuthToken_{email}$ and authenticates as the victim.

SAW's threat analysis argues that SAW provides an acceptable level of risk for sites that employ email-based password resets (EBPR) because they are also susceptible to a similar attack in which an attacker requests a password reset for the victim and then observes the resulting email message sent by the site. The prolific adoption of EBPR indicates that these risks are manageable. Many people

use secure connections for their email, removing the risk.

For more information about SAW's assurances and how it generalizes to other personal messaging mediums (e.g., text and instant messaging) please refer to [10].

2.2 The Chicken and the Egg

SAW cannot be applied to wireless authentication by itself. SAW requires users to retrieve $AuthToken_{pm}$ from their personal messaging provider in order to prove ownership of their personal messaging identifier. Many personal messaging providers (e.g., email, instant messaging) rely on Internet connectivity for message retrieval. SAW authentication with these providers is therefore dependant on Internet connectivity.

The reliance on personal messaging delivery over the Internet introduces an interesting chicken and egg problem. How do users prove ownership of their personal messaging identifiers using the Internet when the reason they are authenticating in the first place is to allow network and Internet connectivity? Four potential solutions have been identified:

- **Temporary Connectivity** First, a temporary connection could be allowed; a user would have limited time to authenticate before his connectivity was terminated. This would give the user ample time to access whatever personal messaging resources he needed in order to prove ownership of his personal messaging identifier.

  This solution carries increased liability and is undesirable as it allows anyone to have temporary connectivity and access to network resources. This access could be used to launch attacks or otherwise circumvent security measures.

- **Filtered Connectivity** A second approach builds on the first. It attempts to allow clients to exchange traffic with only their personal messaging

providers. This would work by allowing the client to submit their personal messaging identifier to the access point. The access point would authorize the identifier and allow the client to obtain IP-level connectivity to a restricted network with limited Internet access. The client would use this restricted access to retrieve $AuthToken_{pm}$ from his provider. After proving knowledge of both authentication tokens, the client would be switched to the real network.

Although only "restricted" network access is provided, this approach could lead to abuse in *at least* three different ways: 1) Provides potential launching points for attacks into the larger network; 2) Allows services to be accessed for non-authorized identifiers on the same provider; and 3) Allows exchange of data not pertinent to retrieving the authentication token.

Protection against abuse is complex but not impossible. Sophisticated filters could restrict traffic to authorized personal messaging protocols. Preventing the use of providers for unauthorized identifiers would require parsing the protocol in transit, a realistically complicated task, to compare the identifier that was first submitted with the one actually being used within the protocol. Filtering would be impossible if encrypted protocols where used.

Enforcing what data was downloaded using the personal messaging protocol would also be difficult. Data-limiting caps could be used but would be difficult to fine-tune because of protocol diversity; users need to exchange just enough information to retrieve $AuthToken_{pm}$ and no more.

The complexity of the safeguards needed to decrease the liability of this approach make it largely unacceptable.

- **Out-of-band Message Delivery** The chicken and egg problem does not ex-

ist for providers that employ out-of-band channels to deliver messages to users. For example, SMS-enabled[1] cell phones exchange messages through the cellular providers' network. SMS could be used to send and retrieve $AuthToken_{pm}$. $Authtoken_{user}$ would be sent to the wireless client. $AuthToken_{pm}$ could then be entered into the wireless client (either manually or using automation software). Authentication would be completed by proving knowledge of both tokens to the wireless access point.

Unfortunately this solution does not work for users without cell phones and could be complicated by latency in the cell phone network. Those who do have cell phones are required to have them available whenever wireless access is desired. Depending on the cell phone plan, a small fee generally charged to send/receive SMS messages may also discourage use.

- **Surrogate Authentication** Ideally, a solution is desired where ownership of an authorized personal messaging identifier is proven to the wireless authentication server without regular network access. A surrogate approach uses the authentication server as a middleman between the wireless supplicant and the personal messaging provider. This allows the authentication server to authenticate the client without granting full network access.

  Several distinguishing factors exist between the surrogate and limited connectivity approaches. An IP-level of connectivity is not required; the Extensible Authentication Protocol (EAP) [1] can be used to tunnel the authentication traffic. The access point has control over what protocols are used to retrieve the authentication token from the messaging provider. The access point directly uses these protocols, limiting unauthorized data transfer.

[1]SMS provides text messaging for mobile devices.

> The surrogate approach is not without its problems. Users are not likely to trust the access point to log in to their personal messaging provider for them to retrieve the authentication token. They need some guarantee that the access point cannot steal the users' credentials or otherwise obtain account access.

In this thesis we develop a surrogate solution that allows the access point to provide successful authentication between the user and their provider without requiring disclosure of the user's credentials.

Chapter 3 — WARP

Wireless Authentication using Remote Passwords (WARP) solves the chicken and egg problem. WARP employs an authentication server (AS) as a surrogate in proving ownership of an authorized identifier without granting full network-access to the wireless supplicant. The AS relays three rounds of messages between the wireless supplicant and the user's personal messaging provider, and then obtains proof of successful authentication between the two parties. Since all traffic flows through the AS, it is important that sensitive information is not leaked that would allow the AS to compromise the user's credentials or impersonate the user at the personal messaging provider.

To prevent such a disclosure a strong password protocol is employed to provide authentication between the supplicant and personal messaging provider. We base WARP on the Secure Remote Password (SRP) [11] protocol, a strong password protocol designed to provide password-based mutual authentication between a user and a host.

WARP requires proof of the authentication to be demonstrated to a third party (the authentication server) without disclosing any sensitive information. SRP by itself does not meet these requirements. WARP augments SRP to create a generalized solution to allow a third-party to assert successful authentication between two parties. The augmented protocol is called Surrogate Secure Remote Password (sSRP). WARP uses sSRP to protect user credentials during authentication.

To provide WARP wireless authentication, administrators first obtain a list of personal messaging identifiers for authorized users. The administrator uses the administrative software on the wireless access point (or wireless authentication server

used by the access point) to input the list of identifiers. His job is now finished. Users authenticate by giving their personal messaging identifier and password to their supplicant software. WARP then uses sSRP to prove successful authentication between the user and personal messaging provider to the access point.

Design Goals In designing WARP, we identify several design goals drawn from the motivating scenarios. These goals fall into two categories: convenience and security.

From the user viewpoint, they should not be required to memorize yet another password. Instead, they should be able to authenticate using the same credentials used to access their personal messaging accounts. This authentication should be done in a secure manner, using sSRP, such that sensitive credentials are never disclosed to the AS.

From the view of administrators, they should not be required to generateor distribute certificates for PKI because this is complex and difficult. They should likewise not be required to distribute or manage user-specific usernames and passwords as this is tedious and burdensome. Instead, the system should only require that a personal messaging identifier be associated with each guest. Although server-side certificates are required on the AS to establish an EAP-TTLS tunnel between the supplicant and AS (to prevent impersonation attacks; see Section 5), WARP should not inhibit deployability or increase sign-up overhead by requiring certificates on the client.

In summary, the design goals are:

Convenient

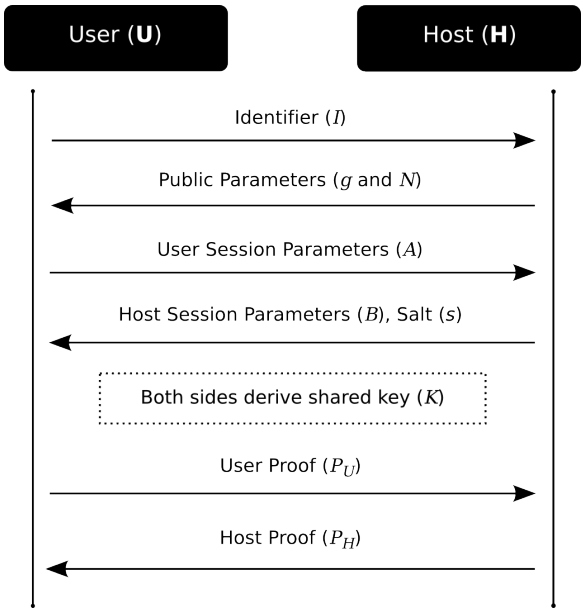

Figure 3.1: SRP protocol outline.

- Users authenticate using familiar credentials
- No additional account provisioning is required
- Access controlled by a list of authorized identifiers

Secure

- User credentials are never disclosed
- Protected against passive eavesdropping
- Resilient against active attacks

3.1 Secure Remote Password

Secure Remote Password (SRP) [11] is a strong authentication protocol that performs authentication using a password. A key-exchange takes place during the authentication, leaving both user and host with a shared key. SRP is a zero-knowledge

proof protocol; it reveals no information to eavesdroppers during authentication that can be used to mount an offline attack against the password. It is also resilient against well-known passive and active attacks. The host does not store passwords for each identifier in plaintext, but instead a unique salt and verifier. The salt is used with the plaintext password to generate the verifier. If the host is compromised, users are protected, as the verifier is not password-equivalent and therefore cannot be used to impersonate the user at the host.

Figure 3.1 shows the SRP protocol. Here is a description of the protocol that takes place between the user (**U**) and the host (**H**):

1. **U**: sends its identifier (I) to **H**.
2. **H**: looks up the public parameters (g and N) for I and returns them to **U**.
3. **U**: generates its session parameters (A) and sends them to **H**.
4. **H**: generates its session parameters (B) and returns them to **U** along with the salt (s) for I.
5. **U** and **H**: derive the shared key (K).
6. **U**: sends proof (P_U) of K to **H**.
7. **H**: verifies P_U (one-way authentication) and sends its own proof (P_H) to **U**.
8. **U**: verifies P_H (mutual authentication).

Further communication between **U** and **H** can be encrypted using keying material derived from the shared key K. It is important that the host not use K before it has verified the user's proof. Failure to do so would provide a malicious user with information that can be used to brute force the password. There are several

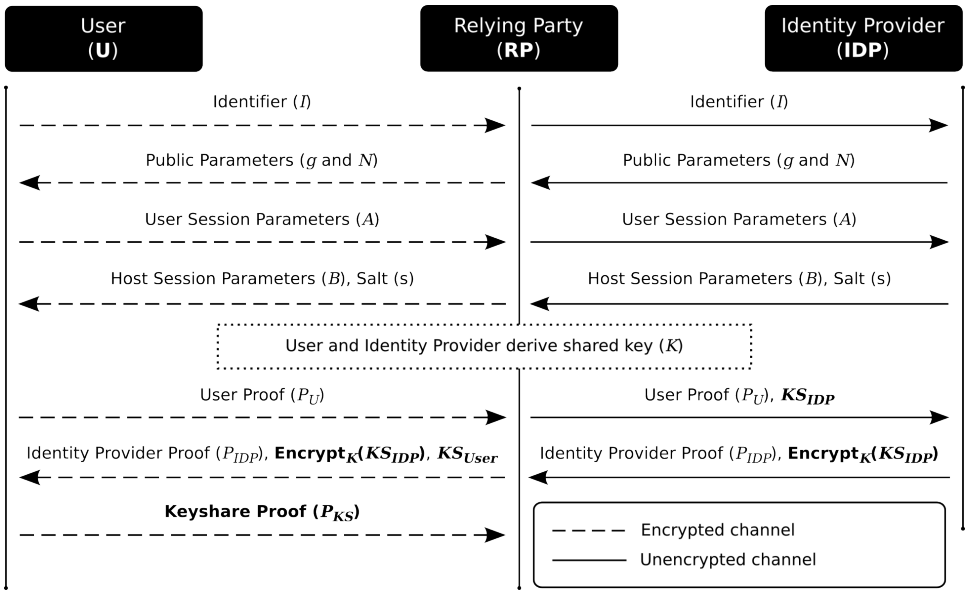

Figure 3.2: sSRP protocol outline (augmentations to SRP in bold).

constraints on the values used within SRP, which are detailed extensively in [11] and [12].

The first round of SRP is used to obtain the public parameters, g and N. It is possible to reduce SRP to a two round protocol if g and N are already known by **U** and **H**. This is done by combining the first two rounds to send I and A to **H** instead of just I.

3.2 Surrogate SRP (sSRP)

We augment SRP for use in WARP, creating the Surrogate Secure Remote Password (sSRP) protocol. SRP is designed to maintain security in the presence of a man in the middle. It is this property that makes SRP an attractive protocol to incorporate into WARP. sSRP intentionally inserts an additional party which acts as a surrogate authenticator for the user by relaying messages between the user and host. Adding additional messages based on the token authentication scheme employed by SAW also allows sSRP to demonstrate proof of successful authentication

between the user and host to the surrogate party (also known as the relying party). sSRP does so without disclosing the user's credentials.

Figure 3.2 shows the sSRP protocol. In sSRP, an identity provider (**IDP**) assumes the role of the SRP host. This protocol also introduces a relying party (**RP**), a party that relies on successful authentication between the user and identity provider, through which all messages are relayed. Like SRP, the protocol is initiated by the user (**U**). Here is a description:

1. **U**: submits its identifier (I) to **RP**.
2. **RP**: performs the following:
   (a) authorizes I and opens up a channel to **IDP**.
   (b) forwards I to **IDP**.
3. **IDP**: looks up I, retrieves I's public parameters (g and N), and sends them to **U** via **RP**.
4. **U**: generates its SRP session parameters (A) and sends them to **IDP** via **RP**.
5. **IDP**: generates its SRP session parameters (B) and sends them, with the salt (s), to **U** via **RP**.
6. **U** and **IDP**: derive the SRP session key (K).
7. **U**: sends proof (P_U) of K to **RP**.
8. **RP**: generates a random value (KS) and splits it into two keyshares[1], KS_{IDP} and KS_{user}. Sends both KS_{IDP} and P_U to **IDP**.

[1]In SAW, KS, KS_{IDP} and KS_{user} are called $AuthToken_{complete}$, $AuthToken_{pm}$, and $AuthToken_{user}$.

9. **IDP**: verifies P_U (one-way authentication).

10. **IDP**: encrypts[2] KS_{IDP} with K and sends that back to **RP**, along with its own proof (P_{IDP}).

11. **RP**: sends the message, along with KS_{user}, to **U**.

12. **U**: verifies P_{IDP} (mutual authentication).

13. **U**: proves knowledge of both keyshares by:

    (a) decrypting KS_{IDP} using K.

    (b) creating the keyshare proof (P_{KS}) by hashing each message along with KS_{IDP} and KS_{user}. Specifically: $P_{KS} = H(I||g||N||A||s||B||P_U||P_{IDP}||KS_{IDP}||KS_{user})$.

    (c) sending P_{KS} to **RP**.

14. **RP**: asserts successful authentication between **U** and **IDP** by verifying P_{KS} (since only successful authentication would have brought KS_{IDP} and KS_{user} to **U** for inclusion in P_{KS}).

sSRP leaves both **U** and **RP** with a shared key (KS_{IDP} and KS_{user}, or in other words KS) to be used as keying material to encrypt future transmissions. This is different than the SRP shared key K, which is of no use after sSRP has completed.

Just like SRP, the first two rounds of sSRP could be combined if the public parameters g and N are known a priori.

In order to protect against eavesdropping and impersonation attacks, the link between the user and relying party must provide confidentiality, integrity, and authentication of the relying party. This protects the transmission of KS_{user} as it is

[2]See Section 5.1.

sent to the user. Section 5.6 presents a motivating discussion about avoiding exploitation of sSRP as a covert channel by encrypting the link between the relying party and identity provider. Barring concern for preventing such an attack, encrypting this link is optional since sensitive information is not transmitted across this channel.

Upon first inspection, it may seem like KS_{user} provides no additional security assurances. KS_{user} serves two purposes: 1) Prevents IDP from having the full keying material (IDP never sees KS_{user}); and 2) Makes KS_{IDP} by itself worthless, as an attacker would need both keyshares to either impersonate the user or decrypt post-authentication transmissions.

sSRP is very helpful in proving ownership of a personal messaging identifier without having to employ the use of the personal messaging medium itself. Ubiquitously deployed, this service would provide a mechanism useful not only to WARP, but also to SAW and many other mechanisms that rely on proof of identifier ownership through password-based authentication.

sSRP has been described as a protocol to authenticate a user using a personal messaging identifier such as an email address or instant messaging handle. However, sSRP can be used with any password-based identity provider where an sSRP service can be deployed. This means that users could authenticate to a wireless network using identifiers such as OpenID or Unix logins.

3.3 Employing sSRP in WARP

WARP is an incarnation of sSRP for wireless authentication. In WARP, the wireless supplicant **S** takes on the role of the user and the authentication server (**AS**) that of the relying party.

EAP-WARP, a new EAP method, has been created to support WARP. EAP-

WARP encapsulates the sSRP protocol as it travels between the supplicant and the authentication server. EAP-WARP works as follows:

1. **S** and **AS**: use EAP-TTLS to authenticate **AS** and provide confidentiality and integrity for the link.

2. **S**, **AS**, and **IDP**: perform three-round sSRP.

3. **AS**: sends an EAP-Success message back to **S**.

4. **S** and **AS**: export KS_{IDP} and KS_{user} as EAP keying material to encrypt wireless communication for the session.

Upon submission of the sSRP keyshare proof P_{KS}, the supplicant has proven to the authentication server its ownership of an authorized identifier residing on the identity provider. In doing so, the supplicant has not revealed any information to the authentication server that would compromise his password on the identity provider.

Chapter 4 — Implementation

Software to support wireless authentication using WARP has been developed and will soon be publicly available. This software includes `libssrp`, is general purpose library that provides sSRP functionality. Supplicant and AS software have been developed using `libssrp` to support EAP-WARP for use in WPA-Enterprise authentication. We describe an sSRP service implementation meant to be deployed on identity providers and perform a simple performance evaluation of WARP.

4.1 libssrp

`libssrp` is a C library that provides the functionality needed to conduct sSRP authentication. The library is general purpose and is meant to be used by applications that supply their own transport functionality. This allows `libssrp` to be used across many different transport mediums.

The current version of `libssrp` relies on `OpenSSL` [9] for its cryptographic primitives and arbitrary precision integers. `libssrp`, by default, relies on an SRP-compatible password file populated with salts and verifiers generated from plaintext passwords. An API can alternatively be used to allow flexible retrieval of salts and verifiers. An argument to stay with the default configuration is given in Section 6.

4.2 Supplicant

In order to provide wireless supplicant functionality, the `wpa_supplicant` [7] open-source package has been extended to support EAP-WARP. The extension consist of two files: 1) One C source file to provide EAP-WARP support, and 2) A patch file that modifies `wpa_supplicant` to include the extension. The extension is less than 400 lines of code.

This simple extension provides a layer that extracts sSRP packet data and pro-

vides that to `libssrp`. sSRP packet data returned from `libssrp` is inserted into an EAP packet that is returned to `wpa_supplicant`. The extension also exports keying material to `wpa_supplicant` using KS_{IDP} and KS_{user}.

4.3 Authentication Server

`FreeRADIUS` [8], an open-source RADIUS server, has also been extended with EAP-WARP support. The extension is around 800 lines of C code and comments, and like the extension for `wpa_supplicant`, provides extraction and insertion of sSRP messages to and from EAP packets, as well as exporting keying material. The `libssrp` library is again used to provide the bulk of the functionality.

4.4 sSRP Service

An incarnation of the sSRP service has been written to use the `libssrp` library to provide sSRP over TCP/IP. The service is written in C, can daemonize, and supports logging to syslog. Although just a prototype, little effort would be required to turn it into a fully deployable service. The functionality the server needs to provide is limited, and identity providers could implement their own in a relatively short amount of time, especially when using the `libssrp` library.

4.5 Performance

Secure authentication systems that are overly expensive in time, computation, or maintainance, are unlikely to be adopted. This section analyzes the performance of WARP and compares it to existing authentication methods.

Two areas are analyzed: 1) supplicant authentication time; and 2) stress tests of the sSRP service. Table 4.1 contains the specifications for machines used in the performance analysis. The RP and IDP reside on the same machine for the first experiment but are thereafter separate. Machines on the network are connected through a 100Mb/s ethernet switch. A Linksys WRT54G Wireless Router acts

User:	IBM Thinkpad T60 Intel Core2 Duo processor at 2.0Ghz 2GB Physical Memory Gentoo Linux running Linux kernel 2.6.21 `wpa_supplicant` wireless supplicant software with WARP patchset
Relying Party:	Dell Optiplex 745 Desktop Intel Core2 Duo processor at 2.6Ghz 4GB Physical Memory Gentoo Linux running Linux kernel 2.6.22 `FreeRADIUS` authentication server software with WARP patchset
Identity Provider:	Dell Optiplex 745 Desktop Intel Core2 Duo processor at 2.6Ghz 4GB Physical Memory Gentoo Linux running Linux kernel 2.6.22 Prototype sSRP service daemon

Table 4.1: Machine specifications for performance analysis.

as an Access Point to connect wireless supplicants to the LAN and provides both WPA-Personal and WPA-Enterprise authentication methods.

Performance results are gathered by testing with an IDP located on the same local area network as the RP (or in our case, the same machine) which is a typical setup for an enterprise environment. Additional performance hits due to latency between the RP and IDP are expected when the IDP resides off-site. Even a few second time delay due to an off-site IDP is likely to be acceptable for authentication in a non-roaming environment (or environments where complete re-authentication

is not necessary when moving from one AP to another).

Authentication Method	Time (Milliseconds)			
	Avg.	Std. Dev.	Min	Max
WARP	197	17	170	239
EAP-TLS	76	37	45	225
WPA-Personal	14	11	4	54

Table 4.2: Performance results comparing three authentication methods over 30 iterations.

Table 4.2 contains performance results from three different authentication methods: 1) WARP; 2) EAP-TLS; and 3) WPA-Personal. EAP-TLS and WPA-Personal are popular authentication mechanisms in enterprise and home environments, respectively. `wpa_supplicant` was instrumented to provide timing information for the exchange of authentication messages to remove the variability in time it takes for the supplicant to associate with the AP and prepare for authentication. Thirty authentications were performed for each method. WPA-Personal takes the least amount of time, as expected. EAP-TLS is on average 62ms slower than WPA-Personal because of the additional computational complexity. WARP, which starts by establishing an EAP-TTLS tunnel between the supplicant and AP, takes on average 2.5 times longer than EAP-TLS. However, since these times are dwarfed by the total time needed for the supplicant to complete authentication (anywhere from 0.4 seconds to 8 seconds finding and associating with the access point), these results suggest that the WARP qualifies as a practical authentication method in a wireless environment.

WPA-Personal is implemented in firmware within the access point and does not require interaction with the authentication server. Although this avoids latency

costs that accumulate through repeated communication with the AS, as is the case in both WARP and EAP-TLS, it also means that WPA-Personal is carried out using the limited processing power of the access point.

Concentrating on the scalability and performance of the sSRP service is valuable since an individual IDP may service authentication requests for any number of RPs. Computation time per authentication on the prototype sSRP service was estimated by profiling the service to measure the computation and ignore communication cost. A single authentication attempt requires approximately 0.044 seconds of computation on the experimental hardware. To measure maximum throughput, several machines on the same network, running special software that simulate the user and relying party portions of the protocol, authenticated continously against a single sSRP service. At 100% CPU utilization, the service fulfills approximately 2700 authentications per minute (about 44 authentications per second). This result is consistent with the computation cost measurement of 0.044 seconds. A dual-core machine should be able to handle approximately 2727 authentications per minute.

WARP lends itself well to load balancing since a secure connection is not necessary between the RP and IDP; the sSRP service could be distributed among several machines to balance incoming requests.

The sSRP service that was evaluated is a research prototype; this means that these numbers likely represent a conservative estimate on performance. Commercial grade implementations should be capable of increased performance.

Chapter 5 — Threat Analysis

This section contains a threat analysis of WARP and the underlying sSRP protocol to enable proper risk evaluation by those deploying WARP.

SRP and SAW are the parent protocols of sSRP. SRP is already resilient to passive eavesdropping and active modification or impersonation attacks. However, introducing new elements into a protocol's messages, as has been done with sSRP, carries great risk; caution must be exercised to not create additional attack vectors and security holes.

sSRP purposefully inserts a middle party in between the user and identity provider in SRP. This creates two channels for attackers to target: 1) One between the user and relying party; and 2) Another between the relying party and identity provider.

We discuss security on both channels individually and together. We then discuss impersonation, denial-of-service, and covert channel attacks.

5.1 Channel between U and RP

The channel between the user and relying party must provide confidentiality, integrity, and authentication of the relying party in order to protect the transmission of KS_{user} and the keyshare proof, P_{KS}. A man-in-the-middle attack is still possible over this channel, depending on how authentication of the relying party is implemented. It is therefore necessary for sSRP to provide protection if the user connects to an attacker instead of the intended relying party.

For example, WARP uses EAP-TTLS to provide security on this channel and authenticate the AS. In order to prevent man-in-the-middle attacks, the supplicant needs to verify the AS's certificate before accepting the connection. A careless

supplicant choosing not to verify the certificate would allow a man in the middle to place himself between the supplicant and AS. The supplicant would establish a TLS session with the attacker, who would then establish a TLS session with the AS. The supplicant would be oblivious to such an attack. All traffic would now flow through the attacker. The attacker can now observe both KS_{user} and P_{KS}.

KS_{user}, by itself, is useless. Without knowledge of KS_{IDP}, the attacker is unable to derive KS. KS_{IDP} is encrypted with the SRP session key K before travelling across this channel to prevent the attacker from obtaining it. The attacker, who does not know K, is unable to decrypt the keyshare and subsequently unable to impersonate the user.

sSRP is built by augmenting SRP with elements of SAW. SAW asserts successful authentication by submission of both keyshares by the browser to the website. This allows for an active impersonation attack by a man-in-the-middle who is able to eavesdrop the keyshares and submit them in place of the browser. sSRP instead sends P_{KS}, a proof of the keyshares, to the relying party.

A man-in-the-middle attack also allows an attacker to intercept P_{KS} before it arrives at the relying party. The proof could then be sent by the attacker to impersonate the user. This would be fruitless however, as a knowledge of KS would be required to further communicate with the relying party. In WARP terms, this means that the attacker would not have the correct EAP keying material to export and would therefore be incapable of communicating further with the access point.

5.2 Channel between RP and IDP

The user relies on the relying party to connect to the correct identity provider. Even if the relying party connects to a malicious identity provider, sSRP prevents that provider from learning anything about the user's password.

The channel that is established between the relying party and identity provider is insecure. SRP parameters are protected by the built-in protections provided by SRP. KS_{IDP} is sent across this channel twice in round 3 of the protocol: 1) In the clear on its way to the IDP; and 2) Encrypted with the shared SRP session key K as it is sent back to U. The AS has knowledge of KS_{IDP} and could attempt to brute-force K by encrypting KS_{IDP} with all possible values for K. Even if the relying party is able to obtain K, K is not helpful in discovering the password, as discussed in [12].

KS_{IDP} can by used by an attacker who is colluding with someone (possibly himself) who has control of the channel in between U and RP. A discussion of this vulnerability is given in Section 5.3.

5.3 Both Channels

One-time impersonation of the user is possible when two attackers that are sitting in between each channel of communication collude with each other. These attackers are: 1) the attacker in between U and RP (Mallory), and 2) the attacker in between RP and IDP (Eve). To initiate this attack, Eve passively observes KS_{IDP} as it is sent from RP to IDP. Mallory likewise obtains KS_{user} as described in Section 5.1. If Eve is able to communicate KS_{IDP} to Mallory, then Mallory can construct KS and impersonate the user. This impersonation is limited to a single authentication because the keyshares are single-use and short-lived.

Although this one-time impersonation attack is complex and unlikely, it can likewise be avoided. Encrypting KS_{IDP} with the IDP's public key before it is sent to the IDP provides confidentiality, since only the IDP would be able to decrypt it. An alternative approach would be to encrypt the entire channel between RP and IDP, but this creates unnecessary overhead as the remainder of the message

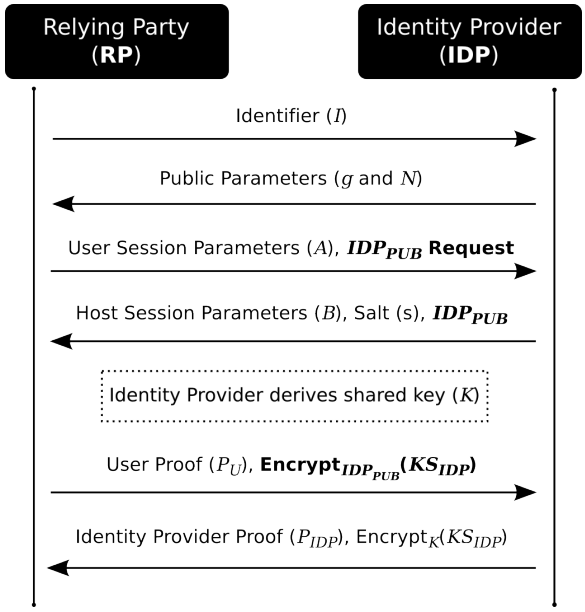

Figure 5.1: One-time impersonation resistant sSRP (augmentations shown in bold).

elements are already protected.

Figure 5.1 shows a slight modification to the interaction between the relying party and identity provider in sSRP, which prevents the one-time impersonation attack. This is accomplished as follows (other interactions remain unchanged):

1. **RP**: appends an optional message, $IDP_{pub}Request$, to A as it travels to **IDP**.

2. **IDP**: fills the request by sending his public key certificate along with B and s to **U** through **RP**. The certificate contains his public key (IDP_{pub}).

3. **RP**: verifies and strips off the certificate (**U** has no need for it).

4. **RP**: encrypts KS_{IDP} with IDP_{pub} before it is sent to **IDP**.

The encrypted KS_{IDP} is only decryptable by the IDP and is therefore useless to an eavesdropper.

5.4 Impersonation By The Provider

Identity providers can impersonate any of their users. If an attacker or malicious insider is able to take control of the sSRP service hosted by the provider, they also gain the ability to impersonate the user to gain wireless connectivity. The attacker may choose to replace the password verifier of an authorized identifier with his own or alter the sSRP service to always grant successful authentication.

WARP relies on an SRP password file on the identity provider to provide verifiers, salts, and public parameters for authorized users. Because the password verifiers are not plaintext equivalent to the password, an attacker who steals the password file is unable to use the verifier to impersonate the user without first determining the user's password.

SAW, and therefore WARP, are built around existing trust given to identity providers. An organization deploying WARP must make judgments regarding the personal messaging providers they choose to trust. The organization can alternatively require use of identifiers within its own messaging systems (such as organizational email addresses) in order to satisfy the required level of trust.

5.5 Denial-Of-Service (DoS)

SRP is resilient against attackers modifying the information being exchanged between the user and host. sSRP does nothing to compromise this resilience. At most, an attacker could cause authentication to fail. This could be used to deny service to an otherwise authorized user. As simpler, jamming-based denial-of-service attacks (DoS) already exist, the potential for DoS attacks using WARP is negligible.

5.6 Covert Channels

A covert channel is a method of communication that uses another channel's bandwidth to transmit data without knowledge or consent. Covert channels take

many different forms (e.g., steganography, timing between transmissions, text manipulation). WARP could be used to send data between the supplicant and IDP. Because WARP only opens up a channel of communication between the supplicant and the IDP of an authorized identifier, it cannot be used as a covert channel to any arbitrary party.

Using WARP as a covert channel would not be an effective means of transferring large amounts of data; the number and size of WARP's messages are quite small. Similar authentication request throttling used to limit DoS attacks could also be employed to greatly reduce the amount of information that could be exchanged over the covert channel.

Since the link between the AS and IDP is unencrypted, the supplicant could communicate information to a passive eavesdropper on that link. Encrypting this link would disallow many types of covert channels (a timing covert channel may still work) except to those parties colluding with the IDP.

Chapter 6 — Deployability

Various issues of WARP deployability are discussed in this section. Deployability of WARP can be discussed from three points of view: wireless users, organizations providing wireless access, and identity providers.

6.1 Users

Users are very familiar and comfortable with password-based systems. WARP does not remove password usage but limits the scope of its usage by leveraging existing login credentials. WARP's user interface is as intuitive and familiar as current password authentication methods, thus maintaining user convenience.

6.2 Organizations

WARP provides convenient wireless authentication to organizations. Adoption is painless because administrators are already familiar with access control lists. WARP softens the burden of password and account management. Providing access to regular organizational staff could be automated by generating the access control list using existing knowledge of staff identifiers. Guest access could then be maintained by manually populating a second access control list.

WARP is unusable in organizations where the authentication server is unable to communicate with identity providers (e.g., ad-hoc networks without Internet connectivity).

WARP assumes the same level of trust extended to identity providers by SAW and may be inappropriate for use in organizations where the existing trust extended to third-party identity providers is insufficient.

6.3 Identity Providers

WARP requires identity providers to host an sSRP service. Until incentives outweigh the cost, it is unlikely that every identity provider will be willing to meet this requirement. Fortunately, sSRP is simple and easy to implement and therefore comes at a minimal cost.

As mentioned previously, sSRP relies on an SRP password file to provide verifiers, salts, and public parameters for authorized users. These password file entries are created during an enrollment process. It may be unfeasible for identity providers to force a re-enrollment process for their users. Identity providers will therefore have to migrate their existing password files.

Migration problems arise because password files generally contain non-plaintext forms of the passwords[1] (e.g., password hash); the SRP verifier is generated using the plaintext password and the salt. In this case, the verifier may have to be generated using the non-plaintext form. When generating verifiers this way, it is *not* recommended that the existing password file be maintained side-by-side the SRP password file. This would allow an attacker who steals the existing password file to impersonate the user at the provider (since he could use the non-plaintext form of the password to impersonate the user).

Identity providers who currently use services that receive a plaintext password from the user (e.g., Unix logins) could modify the service to intercept the password and generate an SRP password file entry. This removes the need for a formal re-enrollment process; users log in once to enable sSRP. The modified service could be deployed well in advance of sSRP deployment to generate SRP password entries for most users. Since existing password files are not used to generate the verifiers,

[1] To prevent easy discovery of the password if the password file is stolen.

they can be safely maintained side-by-side the SRP password file; easing gradual deployment.

Chapter 7 — Related Work

Many authentication systems rely on pre-established shared secrets. System-wide passphrases used in mechanisms like WEP [6] and WPA-PSK [3] can be difficult to distribute. If the shared secret is compromised a new one must be deployed to each device. Guest access can only be achieved by disclosure of the shared secret, a step that should make security conscious administrators cringe. It is also difficult to audit access because each user connects using the same passphrase (MAC addresses could be used but are not a reliable means of identity since they are trivially spoofed). Many of these protocols leak information that can be used to mount an off-line dictionary attack against the pre-established secret; if a weak shared secret is chosen the wireless network could be trivially compromised.

Individual user accounts with passwords are employed by challenge/response-based systems such as MSCHAPv2 [13]. Account-level password-based authentication provides reasonable deployability with a static group of users. It also lends itself to auditing because authentication is tied to a set of credentials. However, these systems suffer from the same issues of other password-based authentication mechanisms: password re-use and difficulty in remembering strong passwords. It is also difficult to configure guest access in organizations where account creation incurs significant overhead or time. Delegation of wireless access is difficult and requires a user to disclose his/her credentials to the delegate.

Recent authentication mechanisms rely on PKI to provide authentication. These systems, when deployed correctly, are quite secure. However, managing certificates and securing public/private key-pairs are challenging tasks for even savvy end-users. The sign-up process can be resource intensive, adding additional overhead on IT staff

who must verify user identity and issue certificates according to stringent company policies. Each authenticating device must also have access to a private key tied to a certificate, which decreases deployability.

Greenpass [5] leverages EAP-TLS [2] for authentication. EAP-TLS relies on PKI to provide the client and server certificates used in the TLS handshake. Greenpass also provides decentralized delegation of access by allowing delegators to sign an SDSI-SPKI certificate belonging to the guest with their X.509 certificates. Guests can then use the SDSI-SPKI as their EAP-TLS client-side certificate. Greenpass suffers from the inherent difficulties of PKI as enrollment of regular users still involves a CA issuing X.509 certificates.

Network-in-a-Box (NiaB) [4] enrolls devices in the wireless network by employing location-limited communication channels (e.g., infrared or a USB key) to securely distribute keys necessary for PKI. Although key distribution is simplified, NiaB still requires significant management for enterprise environments where security restrictions require IT staff to scrutinize each certificate request made during enrollment. In an organization that supports thousands of users this overhead can be significant. Typically only one device may be serviced at a time per enrollment station due to the location-limited channel on which enrollment is done. This one-at-a-time approach creates a bottleneck in circumstances where large amounts of new devices need to be enrolled within a short time period (i.e., wireless connectivity at a conference where most delegates arrive within a short period of time).

Chapter 8 — Conclusions

Wireless Authentication using Remote Passwords is a secure and usable wireless authentication mechanism. WARP is simple and provides administrators with an easy way to manage wireless networks. Managing access is done by supplying administrative software with a list of personal messaging identifiers. Minimal account provisioning is necessary. Users authenticate by proving ownership of an authorized identifier. This is done by providing the supplicant software with a password for an email or other personal messaging account.

WARP is secure and protects against well-known cryptographic attacks. No information is leaked that would allow an attacker to impersonate or otherwise compromise the user's credentials.

sSRP is a general surrogate authentication protocol that can be used to prove authentication between user and host to a relying party. sSRP is based on SRP and inherits many of SRP's cryptographic assurances. sSRP has many uses outside the scope of WARP.

A large amount of research surrounding WARP and sSRP remains to be done.

The usability of WARP could be confirmed by conducting a user study. The study would help to evaluate and improve both the supplicant and administrative software. It could also provide insight into how users react to obtaining wireless access by using their personal messaging identifier credentials.

sSRP could replace the use of email or instant messages in the original SAW protocol to provide website authentication. Using the sSRP protocol in this manner would increase the security of SAW, as it has the potential to thwart active impersonation attacks by third parties, and eliminate latency issues associated with

personal message delivery. SAW can also suffer from usability problems due to latency in email delivery; having to wait more than a few seconds to log in to a website can be uncomfortable for users. If for some reason the email never arrives, as could be the result of an over-protective spam filter, the users are denied authentication capability. sSRP-based SAW would not be subject to these problems and therefore is more convenient and usable than the original SAW.

SAW enables intuitive delegation between personal messaging identifiers and natural client-side auditing capabilities. Further research could provide both of these useful abilities to sSRP.

sSRP need not be limited to personal messaging identifiers only. It can be deployed in conjunction with any username/password-based system.

An interesting combination would be the use of sSRP with OpenID identifiers. OpenID does not require any client-side changes to the browser. Consequently the protocol has to operate in a way that opens a few security holes. Although the benefits of no client-side changes are clear, third-party authentication can be achieved more securely with sSRP. Further research would be needed to fully explore and understand the potential uses of sSRP with OpenID.

References

[1] B. Aboba, L. Blunk, J. Vollbrecht, and J. Carlson. RFC 3748: Extensible Authentication Protocol.

[2] B. Aboba and D. Simon. RFC 2716: PPP EAP TLS Authentication Protocol, 1999.

[3] W.-F. Alliance. *Wi-Fi Protected Access*, 2003. http://wifialliance.com.

[4] D. Balfanz, G. Durfee, R. Grinter, and D. Smetters. Network-in-a-Box. In *USENIX Security Symposium*, 2004.

[5] N. Goffee, S. Kim, S. Smith, P. Taylor, M. Zhao, and J. Marchesini. Greenpass. In *PKI Research and Development Workshop*, pages 26–41, 2004.

[6] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999 (Reaffirmed 2003).

[7] J. Malinen. wpa_supplicant. Available from http://hostap.epitest.fi/wpa_supplicant/.

[8] The FreeRADIUS Server Project. FreeRADIUS. Available from http://www.freeradius.org/.

[9] The OpenSSL Project. OpenSSL. Available from http://www.openssl.org/.

[10] T. van der Horst and K. Seamons. Simple Authentication for the Web. In *Security and Privacy in Communication Networks*, September 2007.

[11] T. Wu. The Secure Remote Password Protocol. In *Network and Distributed System Security Symposium*, pages 97–111, 1998.

[12] T. Wu. SRP-6. Technical white paper, Stanford University, October 2002.

[13] G. Zorn. RFC 2759: Microsoft PPP CHAP Extensions, Version 2, 2000.