11-2004

A Survey of WPA and 802.11i RSN Authentication Protocols

Kwang-Hyun Baek
Dartmouth College

Sean W. Smith
Dartmouth College

David Kotz
Dartmouth College

A Survey of WPA and 802.11i RSN Authentication Protocols

Kwang-Hyun Baek, Sean W. Smith, David Kotz

Dartmouth College Computer Science*
Technial Report TR2004-524

November 2004

Abstract

In the new standards for WLAN security, many choices exist for the authentication process. In this paper, we list eight desired properties of WLAN authentication protocols, survey eight recent authentication protocols, and analyze the protocols according to the desired properties.

1 Introduction

When security experts exposed flaws in *Wired Equivalent Privacy (WEP)* [4, 9, 18, 33, 36], the built-in security protocol for *Wireless Local Area Networks (WLANs)*, many researchers and vendors proposed new security solutions to replace WEP. WEP's weaknesses include a lack of protection against malicious tampering of messages, incorrect usage of an encryption algorithm, and a replayable authentication method (that is, an eavesdropper can sniff a valid user's authentication and replay it to gain the access to the network). *The WiFi alliance*, the international association of wireless device manufacturers, responded to these weaknesses with *WiFi Protected Access (WPA)*, a new industry standard for WLAN security. IEEE has also finalized a draft of *IEEE 802.11i*, also known as *Robust Security Network (RSN)* or *WPA2*, which is specially designed to address WEP's weaknesses.

One significant difference between these new standards and WEP is that the new standards separate the user authentication process from the message protection process [14, pp. 107–108]. In the authentication process, an entity proves that it is eligible to join the network.[1] The authority that decides whether one can access the network is called the *Authentication Server (AS)*. An entity that asks to join the network by initiating the authentication process has many names in the literature: "User," "Client," "Supplicant," or "Authenticating Peer". In this paper, we use the term *client* to describe this entity. Message protection ensures that, once the client joins the network, it can communicate without risks such as interception and modification of messages.

In WEP, all clients of the network share a secret key called the WEP key, which the clients use for both the authentication process and the message protection process. The WEP authentication process is simply

*This project was supported by Cisco and Award No. 2000-DT-CX-K001 from the Office for Domestic Preparedness, U.S. Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security.

[1] In this sense, the WLAN authentication process involves two parts—authentication and authorization. Authentication verifies one's identity, and authorization verifies whether the authenticated entity has the right to access the network.

encrypting a challenge from the AS the same way the client encrypts his messages in the message protection process. In this model, the WEP key shared by many clients is hard to change because it requires changing the key stored in the mobile devices of *all* clients. On the other hand, in WPA and IEEE 802.11i, the clients do not share the same key for authentication and message protection: a client obtains the secret key to use for the message-protection process from a separate authentication process, in which it uses credentials unique to itself. Thus, the separation of authentication and message protection allows dynamic key management, making WPA and IEEE 802.11i more scalable than WEP.

The separation of the authentication process and message protection process allows many existing authentication protocols for wired LANs to be implemented also for WPA and IEEE 802.11i. Given the number of authentication protocols that can be used with these new standards, it may be hard to make the appropriate choice that will work best for a specific WLAN. For this reason, we summarize eight desired properties of WLAN authentication in this paper and survey the authentication protocols that can be used with WPA and IEEE 802.11i. To limit the scope of the paper, we focus only on the authentication protocols for the *infrastructure* mode of WLANs, in which the mobile device sends all its communications to the access point, which acts as a layer-2[2] bridge between the wired and wireless network.

We organize the rest of the paper as follows. In Section 2, we provide the background needed to understand the authentication process in WPA and IEEE 802.11i. In Section 3, we outline and discuss desired properties of the WLAN authentication process. In Section 4, we survey recently proposed authentication protocols and evaluate them according to the properties described in Section 3. In Section 5, we draw conclusions about which of these WLAN authentication protocols may be the most effective in WLANs and discuss future research topics.

2 WPA and IEEE 802.11i

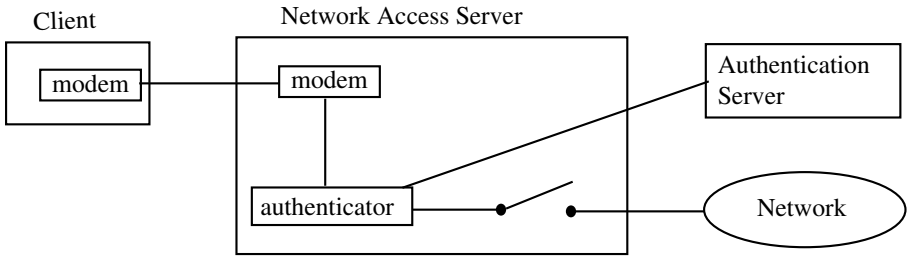
In this section, we present a more complete picture of how the client is authenticated in WLANs. We introduce another entity, the *Access Point (AP)*, also generally known as a *Network Access Server (NAS)*, and the protocol that clients use to communicate with APs during the authentication process—the *Extensible Authentication Protocol (EAP)*.

The authentication process of WPA and IEEE 802.11i adopted the three-entity model of IEEE 802.1x. IEEE 802.1x is the port-based access control protocol that was originally designed for the Point-to-Point Protocol (PPP), such as modem connections and wired LANs (see [24] for details). The three entities are the client, the AS, and the NAS (or in the case of WLANs, the AP).[3] Figure 1 informally describes the relationship among these three entities. As shown in the figure, the AS resides in the network, and the client, who initially does not have access to the network, is connected to the NAS. The NAS is the entity that initially blocks the client's access to the network and also serves as a broker between the client and the AS during the authentication process. Thus, the NAS acts as a "security guard" for the network, allowing only those who are successfully authenticated by the AS, which makes the access decisions. In WLANs, the AP and wireless links replace the NAS and modem connections, as shown in Figure 1(b). Although there have been some changes made in 802.1x to allow the WLANs to adopt port-based access control, the relationship among the three entities remains the same.

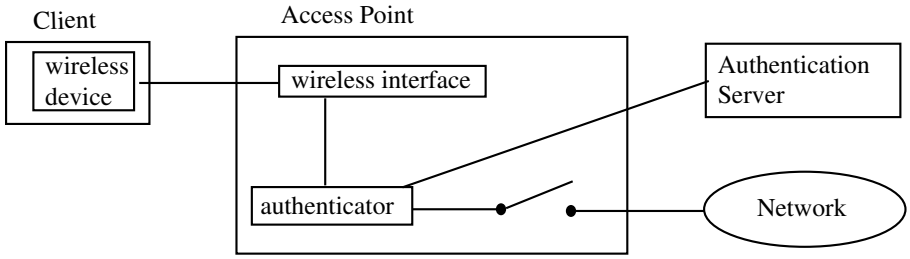
The three entities use EAP [7] to communicate during the authentication process. EAP has four message

[2]Layer 2 refers to the data link layer according to the 7-layer hierarchy of *the Open System Interconnection (OSI)* model.

[3]The client and NAS are not the IEEE 802.1x terminology. IEEE 802.1x calls the client the *supplicant* and the NAS the *authenticator* [24].



(a) Three entity model in 802.1x.



(b) Three entity model in WLANs

Figure 1: The relationship among the client, the NAS (or the AP), and the AS in 802.1x and WLANs. Note that in WLAN the AP and the wireless link replace 802.1x's NAS and modem connections; otherwise, they are the same. The switch in the AP denotes that the AP controls the client's access. The AP also brokers the authentication process between the client and the AS. Only the AS can make the final decision whether the client is admitted to the network. Once the AS makes the decision, it notifies the AP of the decision, and the AP acts accordingly to control the client's access to the network.

types: `Request`, `Respond`, `Success`, `Failure`. EAP can encapsulate other authentication protocols, such as TLS and SRP (defined in Section 4), in its `Request` and `Respond` messages. The AS uses the `Success` or `Failure` message to notify the AP whether the client authentication was successful.

The way EAP messages are used portrays the AP's role in WPA and 802.11i. Figure 2 shows the EAP message flow. Recall the security guard example. Like the security guard, the AP is not aware of the authentication process in detail. It cares only about the AS's decision whether to grant the client the access to the network. Thus, the AP simply passes along the EAP `Request` messages from the AS to the client, and EAP `Response` messages from the client to the AS. The contents of these messages are not important to the AP. Meanwhile, the AP listens for the EAP `Success` or `Failure` message from the AS. If the AS sends the `Success` message, the AP admits the client into the network; if the AS sends the `Failure` message, the AP leaves the client disconnected from the network.

One type of `Request` and `Response` message is noteworthy: the type `Identity`, which is used in the beginning of the authentication process (see Figure 2). The `Request-Identity` and `Response-Identity` messages precede other `Request` and `Response` messages. The AP requests the identity of the new client; the client responds with his identity; and the AP forwards the response to the AS. Only after that

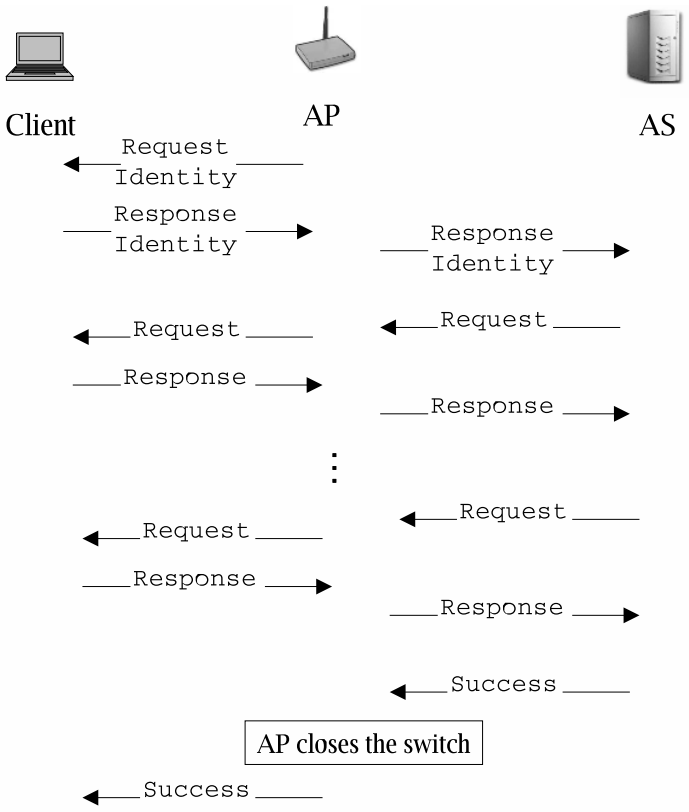


Figure 2: The EAP message flow. The three entities in WLANs use EAP to communicate among themselves during the authentication process. Note that there are only four message types: `Request`, `Response`, `Success`, `Failure`. The message `Failure` is not shown in this figure since this example shows successful authentication. (If the access to the network were denied, the `Failure` messages would replace the `Success` messages in the figure, and the AP would leave the switch open.) Also note that the Request-Identity and Response-Identity messages precede other EAP messages. The `Request` and `Response` messages can encapsulate other authentication methods, forwarding them between the AS and the client.

sequence does the EAP start encapsulating other authentication procols.

An important role of the authentication process is to establish a temporary secret that the client and the AP can use for message protection. The message protection process starts only when the authentication process finishes with the EAP message `Success`, which includes the session key from the AS to use for message protection. The client also computes the same session key from the information exchanged in the authentication protocol. (The detail of how the AS and the client compute the session key differs from one authentication method to another, and we discuss some of these methods in detail in Section 4.) The AP and the client use this session key for message protection.

In summary, the authentication process of WPA and IEEE 802.11i involves three entities: the client, the AP, and the AS. The client seeks access to the network. The AP guards the access to the network, allowing only the clients that the AS has authenticated. Finally, the AS decides whether the client is eligible to gain access to the network. They use EAP, which provides a way for the three entities to embed other

authentication protocols such as those that we discuss in Section 4.

3 Desired Properties of WLAN Authentication

In this section, we describe eight desired properties of WLAN authentication:

1. Mutual authentication,
2. Identity privacy,
3. Dictionary attack resistance,
4. Replay attack resistance,
5. Derivation of strong session keys,
6. Tested implementation,
7. Delegation, and
8. Fast reconnect.

Mutual Authentication

Informally defined, *mutual authentication* is two-way authentication between the AS and the client. The client authenticates the AS to ensure that it is not communicating with an imposter pretending to be the AS. If the authentication method does not enforce mutual authentication, an imposter pretending to be the AS may be able to act as a man-in-the-middle between the client and the AS and gather private messages from the client (this attack is called the *man-in-the-middle attack*). Borisov et al. showed that the absense of mutual authentication in WEP allowed a myriad of attacks, including man-in-the-middle attacks, which an attacker can use to decrypt WEP encrypted messages [9].

Identity Privacy

In this paper, we use the term *identity privacy* to mean hiding the client's identity (e.g., his username or email address) from eavesdroppers of the authentication process. However, by identity, we do not mean *Media Access Control (MAC)* address—a hardware address that uniquely identifies each node, such as the client's 802.11b wireless network card, of a network—since hiding such information would require major changes to the IEEE 802.11 WLAN standards. Recall from the previous section that the EAP message flow starts with the `Request-Identity` and `Response-Identity` messages. Because these EAP messages are sent in plaintext, an eavesdropper sniffing the communication in the beginning of the authentication process can easily discover the client's identity. Thus, the authentication method that extends EAP must take care to hide the identity of the client. We survey some authentication methods that protect the identity of the client in Section 4.

Dictionary attack resistance

In a dictionary attack, the victim must have some potentially guessable secret (usually a password or passphrase), and the attacker has access to some data derived from the secret in a known way, typically independent of the context. Thus, the attacker can verify guesses—and, if the derivation is independent of context, the attacker can pre-compute a dictionary of likely passwords.

In WLANs, an eavesdropper can obtain such derived data, such as the encrypted timestamp of the authentication process and the encrypted username of the client. Some of the legacy protocols, such as MSCHAP—used in Lightweight EAP (LEAP)—are vulnerable to dictionary attacks [12]. Therefore, the authentication protocol must take care not to reveal such data.

Replay attack resistance

In a *replay attack*, an eavesdropper records the authentication process of a legitimate client and replays it to gain the access to the network. Note that replay attacks are possible even when the eavesdropper does not know the secret required for the authentication process. An authentication protocol can resist this attack by including a *nonce*—a timestamp or a sequence number—in the authentication process so that the authenticating parties can detect that a replayed authentication session is not fresh.

Derivation of strong session keys

Any secret is not likely to remain secret indefinitely: If an eavesdropper sniffs many messages encrypted with the same secret, he may eventually be able to derive the secret key from the messages. One major weakness of WEP is that the client and the AP use a static WEP key stored in the wireless device for multiple sessions. The only way of changing the key is manually entering a new key in the AP and in all its clients. As a result, once an eavesdropper discovers the secret key via a key-discovering attack such as a dictionary attack, he can decrypt any message that is encrypted with the discovered key, including the past messages that the eavesdropper has sniffed.

Most authentication protocols that we survey in the next section derive a different *session key* for the client and the AP to use for each session's message-protection process. Thus, even if an eavesdropper discovers the secret key that the AP and the client use for message protection, the eavesdropper cannot decrypt the messages from past or future sessions using the stolen key. Moreover, the longer-lived, multi-session secret is only used to derive the session key during the authentication process. Because the authentication process is shorter, it is less likely for the attackers to gather enough messages encrypted with the same secret.

Tested Implementation

If the authentication protocol is new, the protocol is likely to have more flaws in its design and implementation than existing protocols that have been tested. For an authentication protocol to be used with confidence, its design and implementation need a rigorous security analysis and its limitations need to be thoroughly understood. For example, the IEEE 802.11 work-group initially considered WEP to be as secure as the wired LAN (hence, the name Wired Equivalent Protocol). As mentioned eariler, however, after its implementation, WEP was discovered to be quite insecure.

Delegation

Sometimes it is necessary to have services act on the client's behalf. For example, it is useful to tell the print server to print a remote file that only the client is authorized to read. Although the client's user can log in to each resource it is difficult to arrange a login for scheduled tasks at times when the user is not around. Thus, it would be convenient if the client can give a permission to the printer to read the file on the client's behalf even if the client is not logged in at the time. This permission to act on someone's behalf is known as *delegation*.

Goffee et al. [23] show that delegation can be used for managing guest accounts on a WLAN. It is often a burden for the network administrator to manually manage temporary accounts, especially when there is a big conference at a university where hundreds of temporary accounts have to be created. Moreover, it is not wise to have just a generic guest login with a generic passwords, since secrets shared by many people are generally insecure. Delegation solves this problem because a valid client can delegate the right to access the network to the guests, lifting the burden from the network administrator.

Fast Reconnect

Unlike a wired LAN, a WLAN gives the client the freedom to move from one access point to another while maintaining his connection to the network. Recall that the AS and client authenticate each other to establish trust, and as a by-product of that trust, the client and the AP can trust each other. When the client changes location and associates with another AP, the new AP, which did not broker the authentication process, may not be aware of the trust that the client and the AS established. In such cases, the client may lose connection to the network until he reauthenticates via the new AP.

New applications for WLAN, such as *Voice over IP (VoIP)*, are appearing in the market, and they require seamless connections to the network. Thus, if the client is moving from one access point to another (when a *handoff* occurs between two access points), it is not desirable for the client to go through a lengthy authentication process whenever it associates with a new AP. *Fast reconnect* is a feature that provides a lightweight version of the authentication protocol, so that the client can reuse the credentials from the previous access point for faster reconnection to the network.

4 Proposed Authentication Protocols for WLANs

In this section, we survey many recent WLAN authentication protocols. We group the protocols into three categories: secret-key methods, public-key methods, and tunneled methods. We evaluate each authentication protocol according to the desired properties from the previous section.

4.1 Secret-Key Approach

In *secret-key* authentication methods, the AS and the client have the same secret and establish trust by proving to each other the knowledge of the shared secret key. (Because the same secret key is shared between the authenticating parties, secret-key methods are also known as *shared-key* or *symmetric-key* methods.) Secret-key authentication protocols are efficient and require little computational power. This advantage is especially important in WLANs because many wireless devices, such as PDAs and mobile VoIP phones, have little computational power.

Secret-key authentication methods have several drawbacks, however. Unlike in wired LANs, in WLANs it is easy to eavesdrop on the communications between the authentication server and the client. Because

most secret-key authentication protocols derive the shared secret from the user's password, and because most users choose bad passwords, it is easy for the attacker to gather enough encrypted messages extract the secret key from them, using dictionary attacks [41, 12]. Although some secret-key authentication methods, such as EAP-SRP, do protect the client's password from dictionary attacks, these methods require much greater computational power than other secret-key methods. Moreover, it is hard to securely distribute the shared secret to both parties.

We discuss and compare three secret-key authentication protocols that are being used for WLAN authentication: *Lightweight Extensible Authentication Protocol (LEAP)*, *Kerberos*, and *EAP over Secure Remote Password (EAP-SRP)*.

Lightweight Extensible Authentication Protocol (LEAP)

LEAP [30, 11] was developed by Cisco to address WEP's weaknesses. Because LEAP uses WEP for the message protection process, LEAP does not meet the level of security that WPA and 802.11i RSN provide. LEAP's authentication process, however, is noteworthy because it is the first commercial use of IEEE 802.1x and EAP for WLANs [14, pp. 184–186]. LEAP's authentication protocol also includes mutual authentication and temporary session keys, two of many missing features of WEP authentication.

Figure 3 shows the LEAP authentication message flow. Initially, the client and the AS share a secret MSCHAP[4] key. Following the initial EAP introduction message exchanges, the client and the AS perform MSCHAP challenge-response protocol, originally developed for authenticating modem users, to accomplish mutual authentication. First, the client sends a random challenge to the AS, and the AS responds to the challenge by encrypting it with the secret key that is shared between the AS and client. The client authenticates the AS by decrypting the response from the AS and comparing it to the challenge. If the decrypted response matches the challenge, the AS is authenticated. Similarly, the AS authenticates the client with a challenge. If the mutual authentication is successful, the client and the authentication server derive a temporary session key from the information exchanged during the authentication process. The client and the AP use this session key for the message protection process.

Although LEAP supports mutual authentication and session key derivation, LEAP has some flaws. LEAP does not protect the client's identity because the EAP identity messages are sent in plaintext. Moreover, because an eavesdropper can easily sniff the challenge-reponse pair sent between the client and the AS during the MSCHAP authentication, LEAP is vulnerable to dictionary attacks [12]. LEAP also does not consider other desired properties such as delegation and fast reconnect. Even with these missing features, LEAP has been tested thoroughly and its strengths and weaknesses are well understood.

Kerberos

Developed at MIT in 1993, Kerberos is an authentication protocol designed for TCP/IP. Many places that use wired LAN already use Kerberos for authenticating its clients for services, such as accessing the email server. In this paper, we focus on Kerberos Version 5, whose protocol is explained in detail in elsewhere [28]. We cover the basic concept of the Kerberos authentication model, and discuss how it can be applied to the WLAN.

In the Kerberos model, every service requires a *ticket* and a session key. The tickets and the session keys are issued by the Kerberos AS and its *Ticket Granting Server (TGS)*. Each ticket contains enough information about the client so that the server controlling the service can verify that the client is indeed

[4]MSCHAP is Microsoft's extension to *Challenge-Handshake Authentication Protocol (CHAP)*

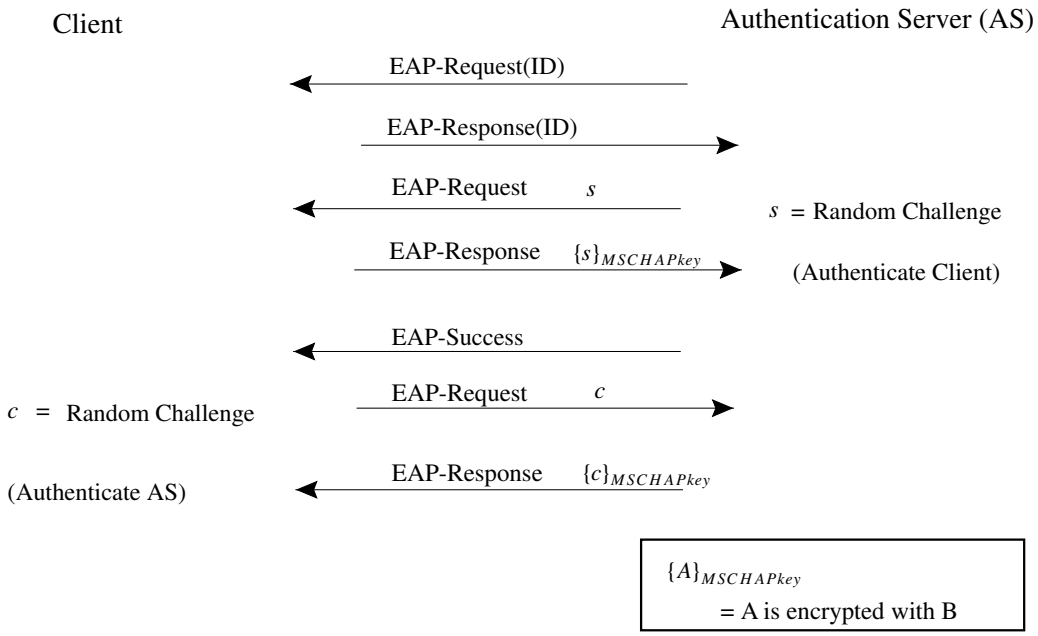


Figure 3: Message Flow for LEAP. The authentication begins with EAP identity messages. The authentication is mutual with a separate challenges from the authentication server and the client. If the mutual authentication succeeds the access point receives the session key from the authentication server and the client calculates the session key individually.

the one to whom the ticket is issued. This information may include the server's name, the client's name, the network address of the client, the beginning and ending validity time for a ticket, and the session key. Tickets are encrypted with the secret shared between the TGS and the server controlling the service so that any tampered information can be detected.

There are two kinds of tickets: *Ticket Granting Tickets (TGTs)* and service tickets. Service tickets are the tickets that grant the access to services to the client. A TGT allows the client to gain service tickets, so the client must first obtain a TGT to obtain service tickets.

There is one more credential that Kerberos uses to prevent replay attack: *authenticators.*[5] An authenticator may include the client's name, the client's network address, and a timestamp. The client presents an authenticator (with a fresh timestamp) along with the ticket and encrypts them with the session key of the ticket so that an attacker who sniffs the authenticator and the ticket cannot replay them.

Figure 4 describes the message flow in the Kerberos model. The AS issues a TGT and a corresponding session key to the client. This session key is encrypted with the secret shared between the client and the AS. Thus, if an imposter pretending to be the client receives the encrypted session key, he will not be able to decrypt the encrypted session key and gain the service ticket because the server can check that the session key value in the ticket does not match the session key that the imposter used to encrypt the authenticator. Moreover, an imposter pretending to be the AS will not be able to generate a correctly encrypted ticket and session key to the client. Thus, in the Kerberos model, the usage of the encrypted session key allows the

[5]Note that Kerberos *authenticator* is different from the *authenticator* in the IEEE 802.1x terminology. In IEEE 802.1x, the NAS is called the *authenticator*.

client and the AS to mutually authenticate each other.

One way of implementing Kerberos in WLAN authentication is to treat the AP as a service that passes data packets to and from the network (see Figure 5). In this model, authentication occurs when the client obtains the TGT and the AP service ticket. This idea is consistent with the port control model of IEEE 802.1x since the AP acts as the server that provides access to the wireless network. But because the Kerberos AS resides within the IP network, which an unauthenticated client would not be able to access, the client would have no way to obtain the TGT and the AP service ticket from the Kerberos AS. To solve this problem, Kerberos authentication in WLANs requires the AP to have a *proxy Kerberos application server*, which is sepcifically designed to help connecting the AS and the TGS with clients who do not have the access to the AS and the TGS. One Internet draft [37] describes the proxy Kerberos application server in detail, and another Internet draft [3] defines how Kerberos messsages can be used over EAP.[6]

Kerberos offers many desired properties. It supports mutual authentication. It generates a session key. Its implementation has been well tested and analyzed[7]. Moreover, the Kerberos authentication protocol can support delegation. To support delegation, Kerberos TGTs also include a field `AUTHORIZATION-DATA`, which specifies application restrictions on the delegatee, and flags such as `FORWARDABLE` (the receiver can further delegate TGTs to another client), `FORWARDED` (this TGT was obtained through delegation), and `PROXYABLE` (the receiver can delegate service tickets to another client). The service tickets that were obtained through a `PROXYABLE` TGT are marked `PROXY`, which specifies that the tickets were issued through delegation. Thus, applications can enforce their own rules about when to honor the delegated tickets. Finally, recall that the tickets are encrypted with the secret key shared between the AS and the server controlling the service. Kerberos can provide fast reconnect for handoffs by having all the APs share the same secret with the AS. Then, after the client obtains an AP service ticket for a particular AP, it can reuse the ticket at any APs until the ticket expires. However, security experts may be uncomfortable about such a widespread usage of a secret [14].

Kerberos also has a few disadvantages. It is vulnerable to dictionary attacks, because an eavesdropper can sniff the encrypted ticket that contains the ticket's timestamp, which the eavesdropper can easily guess [41]. Moreover, since EAP identity messages are not protected, Kerberos does not protect the client's identity from an eavesdropper.

EAP-Secure Remote Password (EAP-SRP)

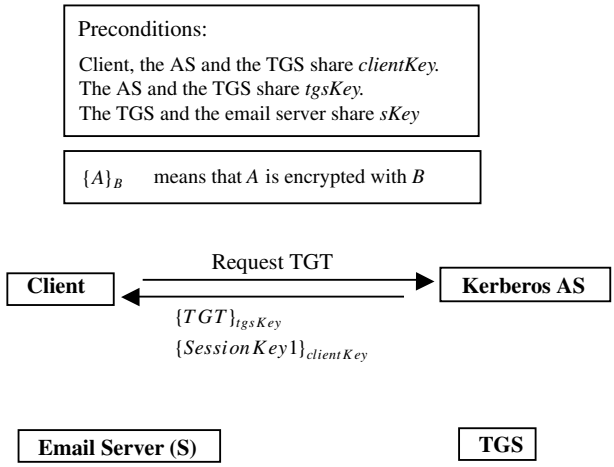
The secret-key methods so far are vulnerable to dictionary attacks. The SRP protocol, proposed by Wu, is one example of secret-key protocols known as *Strong Password Protocols*, which resist dictionary attack [40, 42]. Strong Password Protocols include *Strong Password Encrypted Key Exchange (SPEKE)* and *Augmented Encrypted Key Exchange (AEKE)*.

Informally speaking, SRP is a Diffie-Hellman variant. Diffie-Hellman key exchange is a way for two entities to agree on a secret key by using asymmetric keys [13]. In SRP, the client and the AS compute this asymmetric keys from their shared secret. They exchange these keys and compute a session key based on them. Rather than checking to see if they share the same master secret (the secret they used to compute the asymmetric keys), they check to see if they agree on the value of the computed session key.

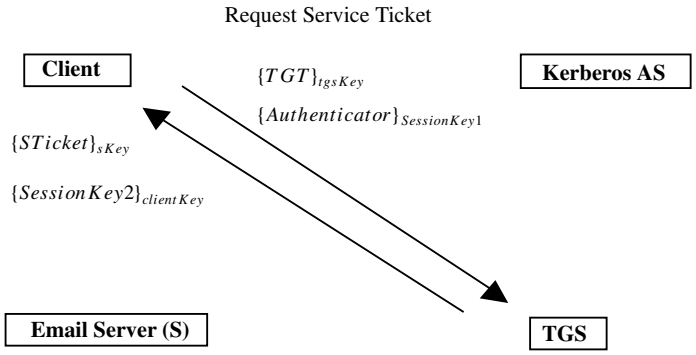
EAP-SRP, still in a draft form, explains one way to use the SRP protocol within EAP [10]. Figure 6 shows message flows for EAP-SRP. Initially, the AS and the client share the following values:

[6]The draft specifies how to use EAP with *General Security Service Application Programming Interface* (GSSAPI), which provides the interface for the proxy Kerberos application server.

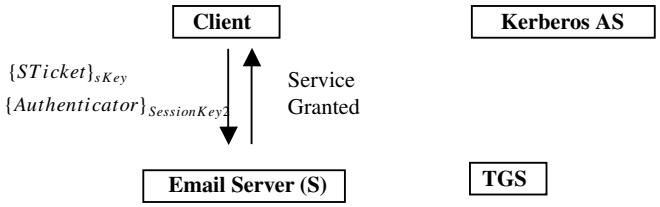
[7]MIT posts security advisories [26] for their Kerberos implementation.



(a) Requesting the TGT and authentication

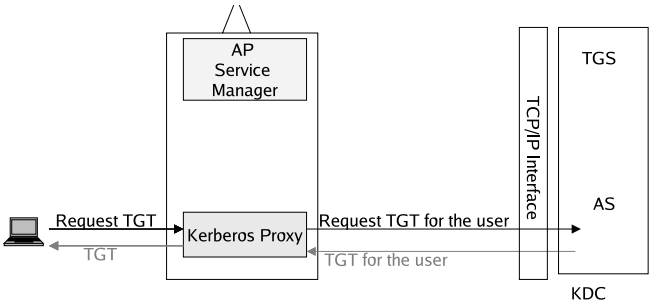


(b) Requesting the Service ticket

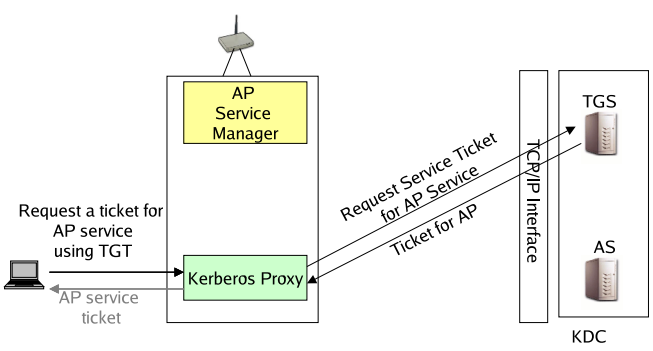


(c) Requesting the email service

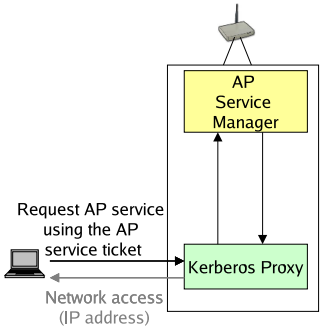
Figure 4: Kerberos model for email service. We present the *general* Kerberos model to explain the detail of the protocol. We chose the email service example since it is being used at Dartmouth. (a) The client first obtains the TGT and $\{SessionKey1\}_{clientKey}$ for the TGT. Only the legitimate client can extract $SessionKey1$ from the message, because it is encrypted with the client's secret, $clientKey$. (b) After extracting the session key, the client hands over the TGT and an authenticator that is encrypted with $SessionKey1$ to the TGS to obtain the service ticket. (c) The client finally presents the service ticket to check his email. Note that each ticket contains its session key so that whenever the ticket is handed, the receiver can verify whether the authenticator is encrypted with the right session key.



(a) Requesting the TGT.



(b) Requesting the AP service ticket



(c) Requesting the AP service

Figure 5: Kerberos authentication in WLAN. The message sequence is the same as Figure 4 except the client does not have access to the Kerberos AS and TGS. Thus, the client must communicate through the Kerberos proxy application server, which simply brokers the Kerberos messages. The client must obtain the ticket for the AP service to access the network. (a) The client requests a TGT through the proxy application server in the AP. (b) Then, using the TGT, the client requests the service ticket for the AP. (c) Finally the user presents the service ticket to the AP service to gain access to the network.

	LEAP	Kerberos	EAP-SRP
Mutual Authentication	Yes	Yes	Yes
Identity Privacy	No	No	Limited
Replay Attack Resistance	Yes	Yes	Yes
Dictionary Attack Resistance	No	No	Yes
Strong per-session key	Yes	Yes	Yes
Tested Implementation	Yes	Yes	No
Delegation	No	Yes	No
Fast Reconnect	No	Limited	Limited

Table 1: Summary of secret-key methods

- $salt$ = a random number,
- g = some constant,
- N = large prime number,
- $v = g^x \bmod N$, where $x = H(salt, username,$ the client's password).

The client chooses a nonce a and computes Message A, where $A = g^a \bmod N$, and the AS chooses two nonces b and u and computes Message B, $B = g^b + v \bmod N$. They exchange A and B and compute the session key based from two messages and the value x, namely $g^{(b(a+ux))} \bmod N$. The client and the AS can compute this message because they know the secret value x, constant g and the prime number N. Note that an eavesdropper who sniffs the messages A and B will not be able to gain the needed information about the shared secret x to mount dictionary attacks due to the usage of modular arithmatic with a large prime number. Once they compute the key, they exchange the hash value of the key to verify that they computed the same value for the key.

In summary, EAP-SRP supports mutual authentication and resists dictionary attack using temporary asymmetric keys. Performing large modular arithmetic during authentication handshakes, however, is computationally intensive and can cause delays [25, pp. 297]. While the Internet draft for EAP-SRP mentions a method for identity privacy through use of a hidden pseudonym, it adds that the method should not be used when strong identity privacy is required [10]. Moreover, its fast-reconnect feature is intended for frequent lightweight authentication initiated by the *same* AP. We do not know how or whether this feature can be used for handoffs, during which which the client needs to be authenticated at different APs. Furthermore, the Internet draft does not mention the possibility of delegation, and we could not find any implementations of EAP-SRP.

Summary of Secret-Key Approaches

Table 1 shows whether LEAP, Kerberos, and EAP-SRP satisfy each of our desired properties. Even though LEAP and Kerberos are well understood and widely deployed, both are vulnerable to dictionary attacks. EAP-SRP overcomes this vulnerability to dictionary attacks using temporary asymmetric keys that are based on the shared symmetric key. EAP-SRP, however, lacks implementation in WLANs. Kerberos can also offer fast reconnection if all the APs share the same secret with the TGS, so that a client's ticket that is issued

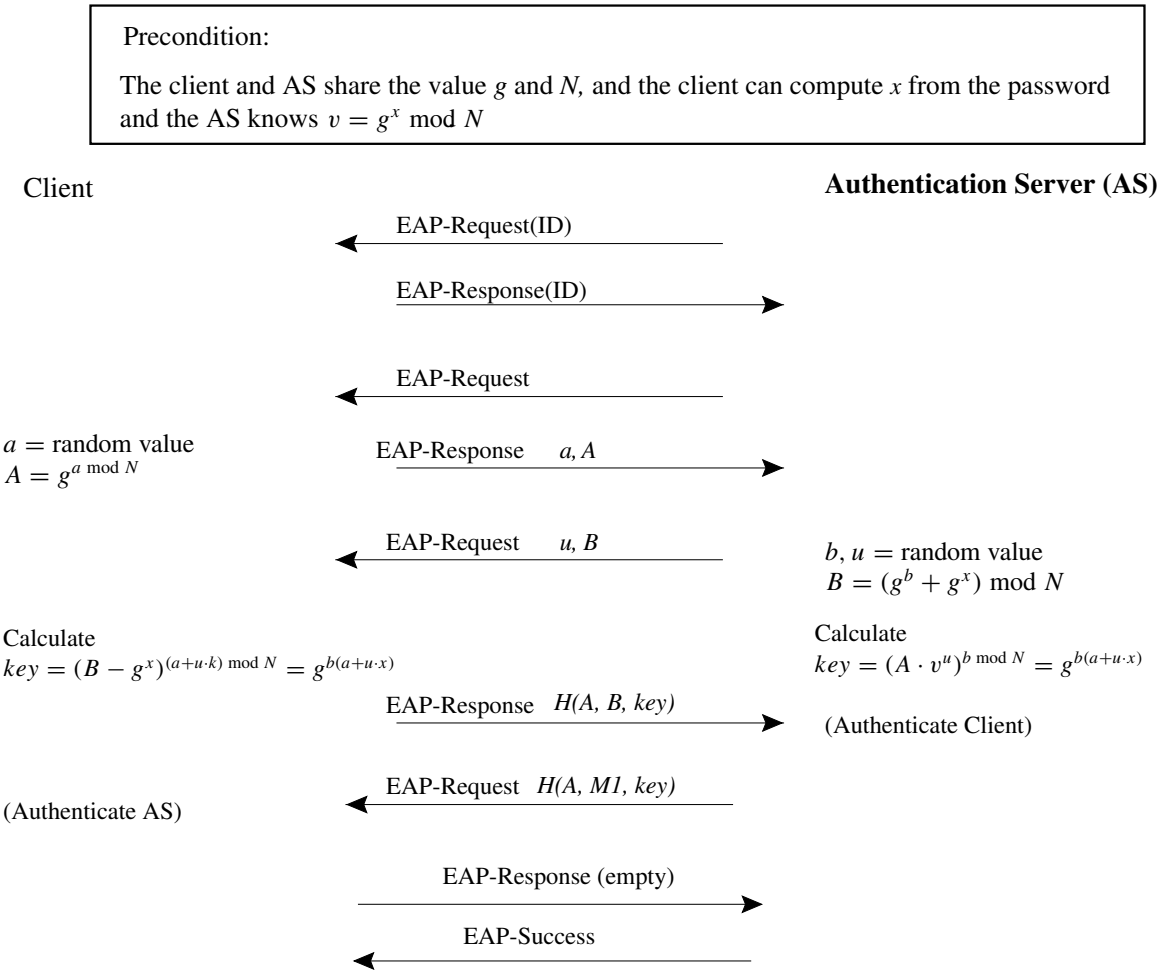


Figure 6: Messages flow in EAP-SRP. The value $x = H(salt, username,$ the client's password). Note that eavesdropper cannot feasibly extract the secret value x from A and B due to the usage of modular arithmetic with a large prime number. After exchanging A, B, u, the client and the AS can compute the same session key, $key = g^{(b(a+ux))} \bmod N$. After computing key, they exchange the hash value of the key to mutually authenticate each other.

for one AP can be accepted by all other APs. As we mentioned, however, sharing secrets in this way is a questionable idea.

4.2 Public-Key Approaches

Unlike the secret-key approach, the public-key approach uses a mathematically connected key pair, a public key and a private key.[8] If a message is encrypted with the public key, it can be decrypted only with the corresponding private key. For example, if the client wishes to authenticate the AS, the client encrypts a challenge with the AS's public key and challenges the AS to prove its identity by decrypting the challenge with the AS's private key. After the the AS decrypts the challenge, it encrypts the challenge with the client's public key so that only the client, who has the corresponding private key, can decrypt it.

To insure that a client's public key is legitimate and to prevent an imposter from advertising his public key as a legitimate client's key, the AS and the client need to establish trust, typically through *Certification Authorities (CAs)*, trusted independent third parties that issue *certificates*. CAs sign their certificates using their private key so that one can verify the validity of the certificate using their public key. Clients are assumed to have, in advance, a copy of the CA's public key to use for validating certificates.

The requirement of well-implemented CAs makes most public-key methods considerably more complicated to deploy than the secret-key methods, however [25].[9] In the absence of proper CAs, an imposter might be able to advertise his public key as the AS's public key since there is no CA to verify that the key belongs to the AS.

We discuss three public-key authentication protocols: EAP over Transport Layer Security (EAP-TLS), Identity-based authentication, and Greenpass, which is based on the Simple Public Key Infrastructure (SPKI).

4.2.1 EAP-Transport Layer Security (EAP-TLS)

IETF RFC 2716 [1] defines EAP-TLS. It is based on a certificate approach, and requires trusted CAs. TLS is a standardized version of the Secure Socket Layer (SSL) protocol, which was developed by Netscape. EAP-TLS extends EAP to provide certificate-based authentication for WLANs.

Figure 7 depicts the EAP-TLS authentication handshake. The client sends a random number c to the AS. Then AS responds by sending its certificate, $cert_{AS}$, and another random number s. If the AS wishes to authenticate the client, it also sends a `certificate request` message at this stage, notifying the client that it should send the client's certificate and digital signature in response. Receiving the certificate from the AS, the client verifies the certificate using the CA's public key. If it is valid, the client selects another random value, p, encrypts it with the AS's public key, and sends it back to the server. This third random value is called *pre-master secret* to reflect that the value is secret and that it will be used to create the session keys. If the network requires mutual authentiation, the client also sends its certificate, $cert_{Client}$, along with the `certificate verify` message. The former contains the client's public key and the latter is the digital signature of the handshake messages signed by the client's private key, so that the AS can authenticate the client by verifying that the client knows the private key that corresponds to the public key in the certificate. The AS and the client derive the same session key using the random numbers they exchanged and the pre-master secret. At the end of the handshake message, the AS sends `TLS-Finished` message

[8] The public key approach is also known as the *asymmetric* key approach.

[9] Note that this issue is part of the general problem of public-key methods and not unique to WLAN.

which contains the *message digest*[10] of the handshake messages, including the pre-master secret. The client authenticates the AS by checking to see if the message digest that the AS sent matches the one the client computed. If the AS does not know the private key that corresponds to the server's certificate, then it would not have been able to obtain the pre-master secret and compute the same message digest as the client.

TLS is well understood and well tested, and the security it provides is trusted by many network security vendors. EAP-TLS supports mutual authentication between the client and the AS if the client also has a certificate signed by a CA that the AS trusts. EAP-TLS resists most attacks, including replay and man-in-the-middle attacks. EAP-TLS also derives a per-session key between the AP and the client after successful TLS authentication.

There are some disadvantages of TLS, however. Some argue that most users do not understand or use the certificates properly. Moreover, EAP-TLS alone does not provide a way to delegate one's access to the network to others. Finally, by itself, EAP-TLS does not provide a way to authenticate clients who do not have a certificate that are signed by the CAs that the AS trusts.

4.2.2 ID-based (ID-based) Cryptography

As mentioned earlier, a certificate-based protocol, such as TLS, is hard to implement due to the requirement of CAs. ID-based cryptography takes advantage of public-key authentication without the complication of certificates. ID-based cryptography is a form of public-key encryption for which the public key can be one's email address or any arbitrary string that identifies the one who holds the associated private key. Gagne in [21] surveys ID-based cryptography and discusses possible applications of ID-based cryptography.

ID-based cryptography works as follows. An ID-based encryption scheme consists of four algorithms:

1. `Setup` generates the system parameters and a master key.
2. `Extract` uses the master key to generate the private key corresponding to an arbitrary string ID, which is the public key.
3. `Encrypt` encodes plaintext using the public key ID.
4. `Decrypt` decodes ciphertexts using the corresponding private key.

A trusted authority, namely the *Private Key Generator (PKG)*, can run the algorithm `Setup` to get the master key. The PKG then runs `Extract` at the request of a user who wishes to obtain the private key corresponding to some strings that belongs to him such as his email address. After obtaining the private key from the PKG, users can then run `Decrypt` to decode the encrypted messages.

ID-based cryptography can ease the revocation of public keys by using public key strings that include expiration date. Moreover, including actual delegated rights in the public key strings can simplify the delegation process.

The protocol proposed by the Lee et al. extends EAP to use ID-based cryptography in the WLAN authentication process [29]. Figure 8 shows the message flow of this protocol. The client and the AS send each other nonces signed with their private keys, and verify the received signature. Session-key derivation, implementation and identity privacy concerns are not mentioned in their paper. The authentication protocol is vulnerable to replay attacks because an eavesdropper can simply sniff the random number and the corresponding digital certificate and replay it to gain access to the network. Moreover, many of the advantages

[10] A message digest is a compressed data created with one-way hash function. Also it is impossible to change a message digest back into the original data from which it was created.

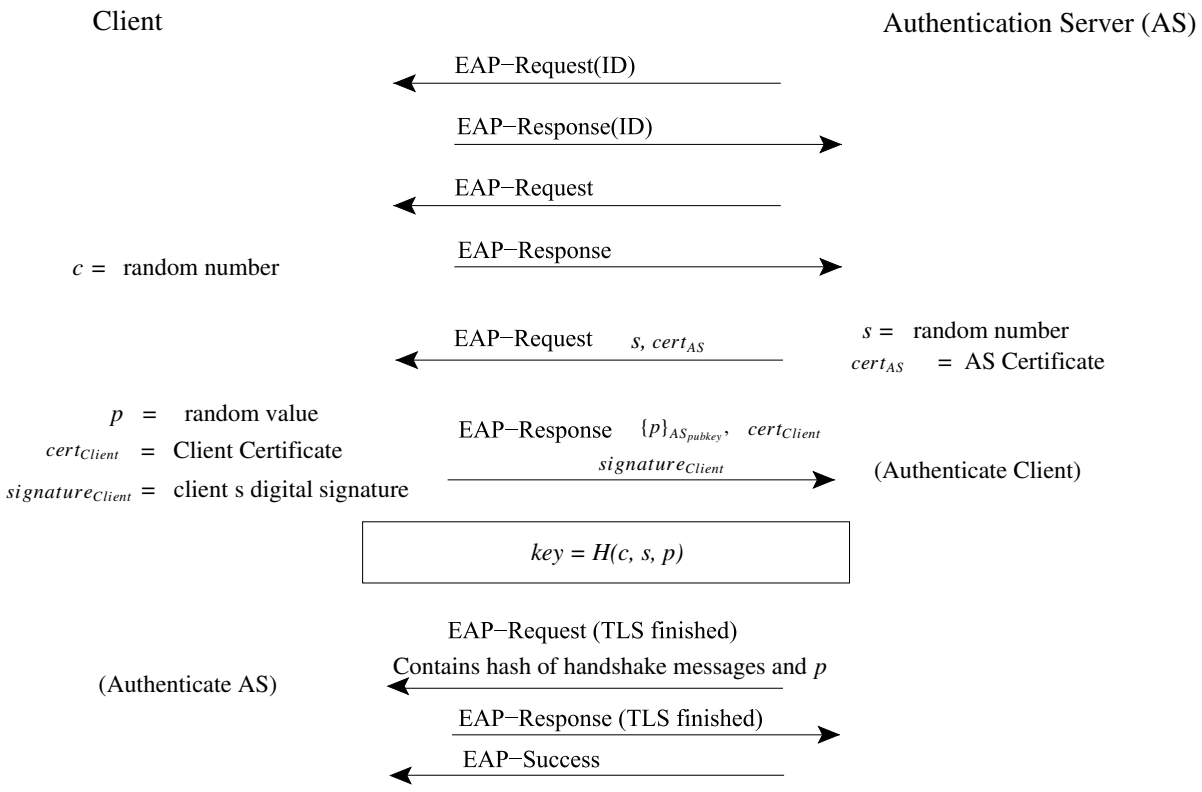


Figure 7: Message Flow for EAP-TLS. The client and the server verify each other's certificate and exchange the pre-master secret. The session key computed from EAP-TLS is $H(c, s, p)$, where c and s are nonces that the client and the AS choose, and p is the pre-master secret. The client and the AS exchange c, s in the clear. If the AS's certificate is valid, the client chooses p, encrypts it with the AS's public key, and sends it back to the AS. Because p is encrypted with AS's public key, only the AS can decrypt p with its private key. After exchanging c, s, and p, the AS and the client compute the session key by computing the hash of these three values.

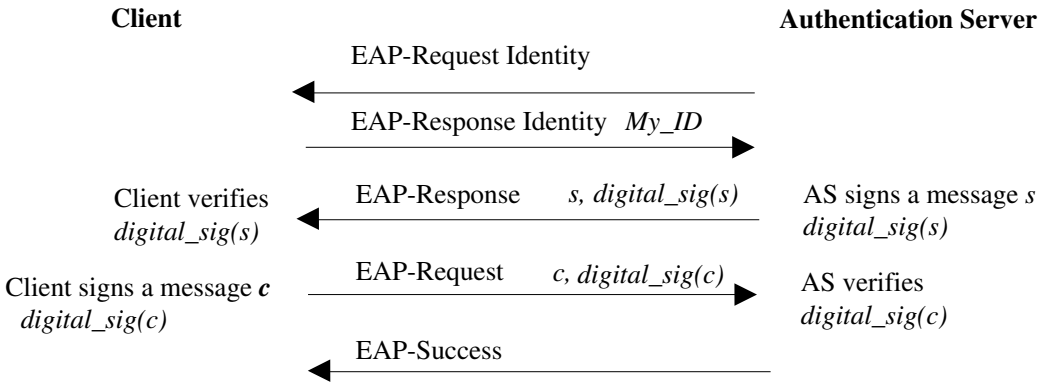


Figure 8: ID-Based Cryptography in WLAN. The client and the AS exchange nonces and corresponding digital signatures. They authenticate each other by verifying each other's signatures.

of ID-based cryptography, such as delegation, have not been implemented yet, at least in any application to WLANs.

4.2.3 SPKI Certificate and Greenpass

SPKI certificates [15, 16, 17] focus on *authorizations* rather than authentication. Rather than relying on a centralized source of trust, trust can be built if there exists a chain of SPKI certificates that expresses a sequence of authorizations from a trusted source.

A SPKI certificate is a 5-tuple:

$$(I, S, D, A, V),$$

where I stands for the issuer of the certificate, S stands for the subject who receives the authorization carried by the certification, D is a flag indicating whether the subject S is allowed to delegate authorization to others, A is the list of authorized rights associated with the certificate, and V is a specification of dates or conditions under which the certificate is valid. When there are two certificates $(I_1, S_1, D_1, A_1, V_1)$ and $(I_2, S_2, D_2, A_2, V_2)$, where $S_1 = I_2$ and D_1 indicates that I_1 can delegate authorizations, the two certificates form a chain, namely, $(I_1, S_2, D_1 \cap D_2, A_1 \cap A_2, V_1 \cap V_2)$. In this way, SPKI's certificate chain replaces certificates. Thus, trust in SPKI is built from certificate chains.

Dartmouth College is currently developing Greenpass [23, 22, 27], an authentication protocol based on SPKI certificates and EAP-TLS. Recall that EAP-TLS provides strong security but lacks the desired property of delegation. Moreovever, client authentication for EAP-TLS requires the client to have a certificate issued from CAs that the AS trusts. By using SPKI certificates in conjunction with EAP-TLS, one can take advantage of both strong security of EAP-TLS and a flexible means of delegation through SPKI certificates. Greenpass aims to ease the complication of guest authorization, especially when the guest's institution does not have CAs or a CA that has not yet established trust with the AS of the host institution. In the Greenpass model, chosen delegators can delegate the right to access the network to guests who cannot use EAP-TLS authentication.

Figure 9 shows how a guest (who does not possess a certificate trusted by the AS) can gain access to the network via a SPKI certificate that is issued and signed by a delegator. It is assumed that the guest and the delegator have already established trust out-of-band. This assumption is reasonable since the guest and the

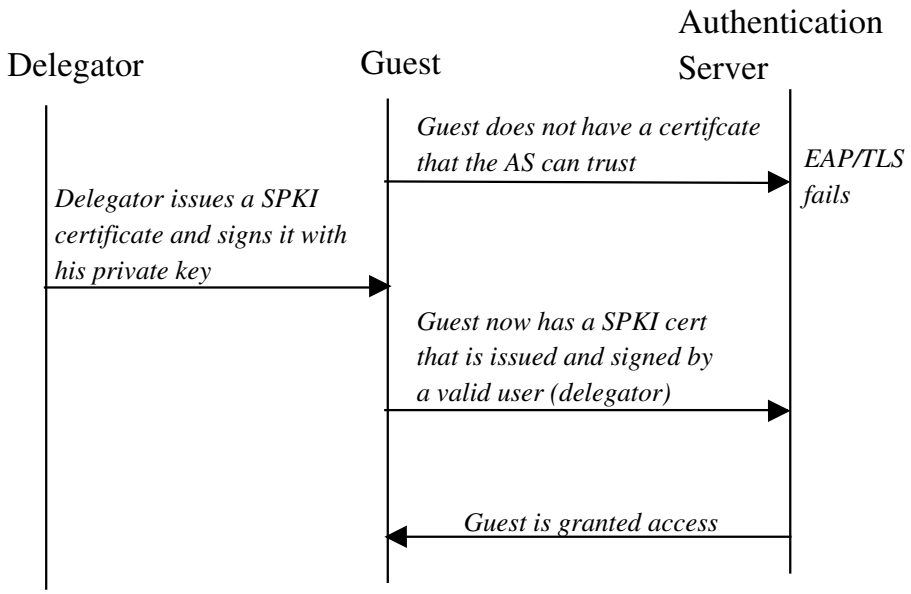


Figure 9: Guest authorization using Greenpass. This figure illustrates the case when the guest does not have a certificate that the AS trusts. The delegator is assumed to have a valid certificate that the AS trusts. A SPKI certificate, signed with delegator's private key, binds the guest's certificate with the delegator's certificate, allowing the guest to gain access to the network.

delegator may already be acquaintances or can check each other's identity through other means. The figure only illustrates the case when the guest does not have a certificate that the AS can trust[11]. Since there is no trusted certificate that can authenticate the client, the client cannot carry out EAP-TLS client authentication. Then the delegator, who is a valid user and holds a certificate that the AS trusts, issues a SPKI certificate and signs it with his private key. The SPKI certificate binds the guest's certificate with the delegator's certificate. This SPKI certificate authorizes the guest to access the network.

Greenpass [23] provides mutual authentication, delegation, and derivation of session keys. It has no known vulnerabilities to dictionary attacks or replay attacks.

4.3 Tunneled Approaches

Finally, we discuss two tunneled methods, which are still being revised as Internet drafts: *Protected EAP (PEAP)* [32] and *EAP-Tunneled TLS (EAP-TTLS)* [20]. These authentication protocols have two phases (Figure 10). In the first phase, the client authenticates the AS using EAP-TLS, and use the resulting session key to establish an encrypted *tunnel* to encrypt their communication. In the second phase, the AS authenticates the client through the encrypted tunnel. The choices of the client-authentication methods separates EAP-TTLS from PEAP. While PEAP supports any EAP methods, EAP-TTLS supports not only EAP methods but also legacy password protocols such as MSCHAP.

The tunnel has two purposes. First, as mentioned earlier, it allows use of a less secure legacy protocol

[11] Greenpass EAP-TLS AS is implemented with FreeRADIUS, and to our knowledge, it only supports oligarchy model. It does not support certificate chains.

	EAP-TLS	Greenpass	ID-based Crypto	PEAP/EAP-TTLS
Mutual Authentication	Yes	Yes	Yes	Yes
Identity Privacy	No	No	No	Yes
Replay Attack Resistance	Yes	Yes	No	Yes
Dictionary Attack Resistance	Yes	Yes	Yes	Yes
Strong per-session key	Yes	Yes	No	Yes
Implementation	Yes	Yes	No	Yes
Delegation	No	Yes	Yes	Maybe
Fast Reconnect	No	No	No	No

Table 2: Summary of public-key and tunneled methods. PEAP and EAP-TTLS support delegation if the authentication protocol the client uses in the second phase implements delegation.

for client authentication in the second phase. Recall that for mutual authentication, EAP-TLS requires the client to have a certificate issued by CAs that the AS trusts. Because the encrypted tunnel from the first phase hides the content of the messages sent during the the second phase, the client and the AS can be sure that the client authentication is as secure as EAP-TLS, but without requiring a CA that supports client with. Thus, PEAP and EAP-TTLS can provide mutual authentication that is as secure as EAP-TLS even when only legacy client-authentication methods are available.

Second, using the tunnel hides the client's identity from an eavesdropper by hiding the EAP `Response-Identity` message in the encrypted tunnel. To do so, in the first phase of the authentication process, the client's EAP `Response-Identity` message contains a generic domain name instead of the username. Since the AS does not authenticate the client in the first phase, the AS ignores the client's identity in the EAP `Response-Identity` message. The client authenticates the AS by standard EAP-TLS method. When the TLS handshake is finished with `TLS Finished` message, the client initiates the second phase by sending his username through the encrypted tunnel.

PEAP and EAP-TTLS have many advantages. Not only does the tunnel provide identity privacy, but it can also provide delegation if the client authentication method that is used in the second phase provides delegation. Moreover, even when the client authentication protocol is vulnerable to dictionary attacks or replay attack, in the tunneled second phase it becomes no longer vulnerable to these attacks because the eavesdropper sniffing the tunneled session must break the secure EAP-TLS tunnel to mount these attacks on the client authentication. Finally, PEAP and EAP-TTLS are available in commercial products (PEAP is developed by Microsoft and Cisco, and EAP-TTLS is developed by Funk Software), and their security concerns are going through public scrutiny.

Recently, however, Asokan et al. discovered a man-in-the-middle attack in these tunneled protocols [5]. Because EAP-TTLS and PEAP support legacy authentication methods that may not create session keys, the protocols require that the session key from the first phase—the key they use to encrypt the tunnel—to be the session key for the message protection process. Morever, because some clients using backlevel OS and software may not be able to perform EAP-TLS authentication, EAP-TTLS and PEAP allow the client to forego the tunneling and proceed to the second phase. These two conditions together introduce a man-in-the-middle attack with which an attacker can steal a legitimate client's session. Nevertheless, there are possible solutions to resist the attack, and an Internet draft was issued by IETF concerning this problem [34]. The working group expects that the drafts specifying EAP-TTLS and PEAP will apply appropriate changes to prevent the attack.

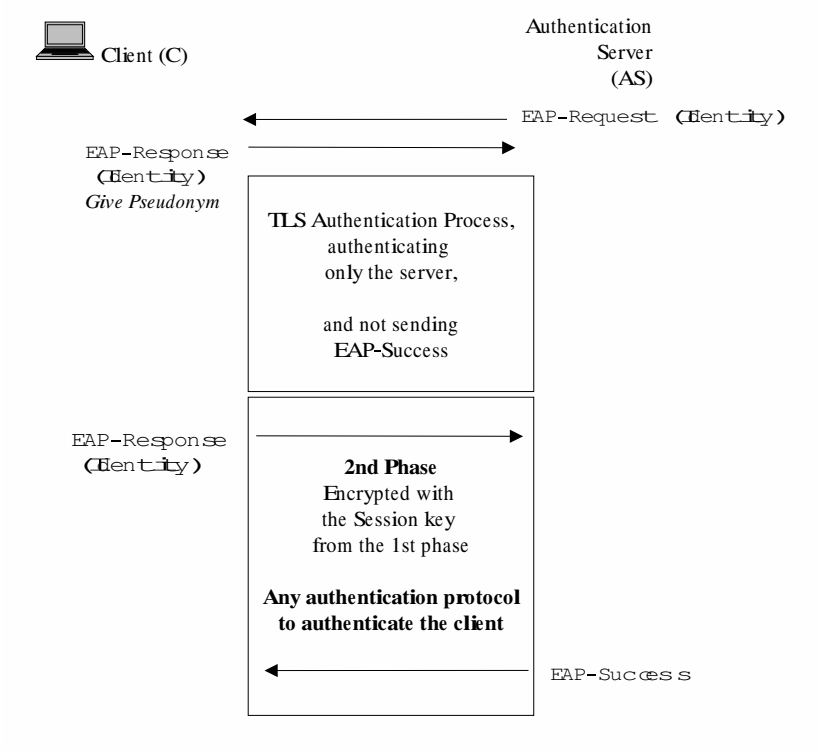


Figure 10: Message Flow for PEAP. There are two phases of authentication. During the first phase, the client authenticates the AS using EAP-TLS, and the client and the AS create an encrypted tunnel using the session key that resulted from the EAP-TLS authentication process. In the second phase, all communications are encrypted, and the AS authenticates the client during this phase. If the client wishes to hide its identity, it can give a pseudonym instead of its identity in the EAP Response-Identity message in the first phase and reveal its identity to the AS in the second phase when the communication is tunneled.

Summary of Public-Key and Tunneled Approach

Table 2 shows whether EAP-TLS, Greenpass, ID-based Crypto, PEAP and EAP-TTLS satisfy our desired properties. EAP-TLS satisfies all properties, except identity privacy, delegation and fast reconnect. Greenpass and PEAP/EAP-TTLS provides the security of EAP-TLS without requiring that the client's CAs to be trusted by the AS's CAs. Moreover, the tunneled protocols satisfy additional desired properties such as identity privacy and delegation. ID-based Crypto has the potential to simplify revocation and delegation, but missing features, such as lack of implementation and lack of session key derivation, make it an inappropriate choice for securing WLANs.

5 Related Work

Welch and Lathrop give a taxonomy of threats against WLANs and surveys security technologies in general [39]. They present eight types of attacks against WLANs and conclude that a secure WLAN that can protect its infrastructure against these threats must provide the following security technologies: (1) mutual

authentication, (2) a link-level layer encrypted tunnel with a strong encryption method, and finally, (3) strong cryptographic integrity verification.

Welch also discusses two-level client authentication. The client has two entities: one is the mobile device and the other is the human user that uses the device. If the AS only authenticates the device, then anyone who has the access to the device has the access to the network. For example, if the mobile device is stolen, the thief automatically gains the access to the network. Moreover, the device can be shared among different users, who may or may not eligible to access the network. Human users are limited in their ability to remember long, complicated secrets. The result is that users may choose easy-to-crack passwords. Although the two-level client authentication is a desired property, none of protocols in this paper makes it a requirement. It may suffice to authenticate the user's access to the device (although many current operating systems support autologin) or to the network service (such as a web site or file server). Still this issue warrants further study.

Edney and Arbaugh present a comprehensive information on WPA and 802.11 RSN, as well as the security threats and open source implementations [14]. We recommend Chapters 7–13 of this book to readers who wish to learn the details of WPA and 802.11i RSN. Aboba maintains a frequently updated website with an unofficial guide to 802.11 security [2] where he lists the relevant papers and standards with links to their sources.

There are numerous proposed authentication protocols other than those described in this paper. They include EAP-Message Digest Challenge (EAP-MD5 Challenge) [19], EAP-Archie [38], and EAP-One Time Password (EAP-OTP) [8]. In particular, we did not discuss methods using tokens such as EAP-OTP. It may be valuable to compare these token-based solutions with the methods that we discuss in this paper.

Another solution to secure WLAN is through the use of a *Virtual Private Network (VPN)*. Most VPNs use *IP-Security (IP-Sec)* to tunnel the authentication messages and packets. Wireless gateway vendors such as Bluesocket, Reefedge, and Trapeze Networks have published "white papers" evaluating their VPN solution for WLAN [6, 31, 35]. They argue that the VPN solution does not protect the IP addresses as the WPA or 802.11i RSN solutions do; thus, an attacker can learn about the destination address of the traffic. VPN solutions are also expensive compared to the WPA and 802.11i RSN solutions, because VPN solutions require expensive VPN concentrators to provide authentication, access control, and a tunnel endpoint for all data traffic. Moreover, most implementations of VPNs on the market are proprietary, so interoperability is poor. Finally, many VPN implementations have security flaws. A detailed discussion of the VPN security solution for WLANs is outside of the scope of this paper.

6 Conclusion

In this paper, we present eight desired properties of WLAN authentication protocols. We survey and compare eight authentication protocols: LEAP, Kerberos, EAP-SRP, EAP-TLS, Greenpass, ID-based cryptography authentication, EAP-TTLS, and PEAP. We find that LEAP and Kerberos are not sufficiently secure due to their vulnerability to dictionary attacks. EAP-SRP and ID-based Privacy lack current implementations for WLANs. EAP-TLS provides strong security if the network users are not concerned with delegation and identity privacy. We find Greenpass, EAP-TTLS and PEAP to be the most promising approaches since they provide the strong security offered by EAP-TLS as well as additional features, including delegation (Greenpass) and identity privacy (EAP-TTLS and PEAP). Moreover, these protocols overcome some of the difficulty of authenticating the client in EAP-TLS (that is, requiring the client to possess certificates issued by CAs that the AS trusts).

In the future, it would be interesting to examine token-based authentication methods and compare them to the secret-key and public-key protocols mentioned here. Moreover, after studying these eight protocols, it was not obvious whether their fast reconnect features will support sufficiently fast handoffs between APs. In general, it would be useful to characterize how these authentication protocols handle a fast-roaming client.

References

[1] B. Aboba and D. Simon. PPP EAP TLS Authentication Protocol. IETF RFC 2716, October 1999.

[2] Bernard Aboba. Wireless LAN Security Site. Available at `http://www.drizzle.com/~aboba/IEEE/`.

[3] Bernard Aboba and Dan Simon. EAP GSS Authentication Protocol. IETF Internet Draft, draft-aboba-pppext-eapgss-12.txt, April 2002.

[4] William A. Arbaugh, Narendar Shankar, Y. C. Justin, and Kan Zhang. Your 802.11 Wireless Network Has No Clothes. *IEEE Wireless Communications*, 9(6):44–51, December 2002.

[5] N. Asokan, Valtteri Niemi, and Kaisa Nyberg. Man-in-the-Middle in Tunnelled Authentication Protocols. Cryptology ePrint Archive, Report 2002/163, 2002.

[6] Bluesocket. Wireless Gateways: Going Beyond VPNs for WLAN Security and Management Solutions. Bluesocket Bluepaper: BSI-VPNS Bluepaper 121702, December 2002.

[7] L Blunk and J. Vollbrecht. PPP Extensible Authentication Protocol (EAP). IETF RFC 2284, March 1998.

[8] L. Blunk, J. Vollbrecht, and B. Aboba. The One Time Password (OTP) and Generic Token Card Authentication Protocols. IETF Internet Draft, draft-ietf-eap-otp-00.txt, October 2002.

[9] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of 7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001. ACM Press.

[10] J. Carlson, B. Aboba, and H. Haverinen. EAP SRP-SHA1 Authentication Protocol. IETF Internet Draft, draft-ietf-pppext-eap-srp-03.txt, July 2001.

[11] Cisco. Wireless LAN Security White Paper. Available at `http://www.cisco.com/warp/public/cc/pd/witc/ao1200ap/prodlit/`, 2002.

[12] Cisco. Dictionary Attack on Cisco LEAP. Tech Note, available at `http://www.cisco.com/warp/public/707/cisco-sn-20030802-leap.shtml`, August 2003.

[13] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 19(3):644–654, 1976.

[14] Jon Edney and William A. Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley, 2003.

[15] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Examples. IETF Internet Draft, draft-ietf-spki-cert-examples-01.txt, March 1998.

[16] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. Simple Public Key Certificate. IETF Internet Draft, draft-ietf-spki-cert-structure-06.txt, July 1999.

[17] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory. IETF RFC 2693, September 1999.

[18] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *8th Annual Workshop on Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, Toronto, Canada, August 2001. Springer-Verlag, Berlin Germany.

[19] Paul Funk. The EAP MD5-Tunneled Authentication Protocol (EAP-MD5-Tunneled). IETF Internet Draft, draft-funk-eap-md5-tunneled-00.txt, March 2003.

[20] Paul Funk and Simon Blake-Wilson. EAP Tunneled TLS Authentication Protocol (EAP-TTLS). IETF Internet Draft, draft-ietf-pppext-eap-ttls-03.txt, August 2003.

[21] Martin Gagne. Identity-Based Encryption: A Survey. *RSA Laboratories Cryptobytes*, 6(1):10–19, 2003.

[22] Nicholas C. Gofee. Greenpass Client Tools for Delegated Authorization in Wireless Networks. Master's thesis, Dartmouth College, June 2004.

[23] N. Goffee, S. Kim, S.W. Smith, P. Taylor, M. Zhao, and J. Marchesini. Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *3rd Annual PKI Research and Development Workshop*, pages 26–41. NIST, April 2004.

[24] IEEE. IEEE Standards for Local and Metropolitan Area Networks: Standard for Port Based Network Access Control. IEEE Std 802.1x-2001, available at `http://standards.ieee.org/getieee802/download/802.1X-2001.pdf`, 2001.

[25] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security, Private Communication in a Public World*. Prentice Hall PTR, 2nd edition, 2002.

[26] MIT Kerberos. Kerberos security advisories. Available at `http://web.mit.edu/kerberos/advisories`.

[27] Sung Hoon Kim. Greenpass RADIUS Tools for Delegated Authorization in Wireless Networks. Master's thesis, Dartmouth College, June 2004.

[28] John Kohl and B. Clifford Neuman. The Kerberos Network Authentication Service (Version 5). IETF RFC 1510, September 1993.

[29] Byang-Gil Lee, Doo-Ho Choi, Hyun-Gon Kim, Seung-Won Sohn, and Kil-Houm Park. Mobile IP and WLAN with AAA authentication protocol using identity-based cryptography. In *10th International Conference on Telecommunications ICT*, volume 1, pages 597–603, 2003.

[30] Cameron Macnally. Cisco LEAP protocol description. Available at `http://www.missl.cs.umd.edu/wireless/ethereal/leap.txt`.

[31] Trapeze Networks. The Illusion of Security: Using IPsec VPNs to Secure the Air. White Paper, available at `http://www.trapezenetworks.com/tech/tech_IPsecVPN.asp`.

[32] Ashwin Palekar, Dan Simon, Glen Zorn, Joe Salowey, Hao Zhou, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. IETF Internet Draft, draft-josefsson-pppext-eap-tls-eap-07.txt, 26 2003.

[33] N.L. Petroni, Jr and W.A. Arbaugh. The Dangers of Mitigating Security Design Flaws: A Wireless Case Study. *IEEE Security & Privacy*, 1(1):28–36, 2003.

[34] Jose Puthenkulam, Victor Lortz, Ashwin Palekar, and Dan Simon. The Compound Authentication Binding Problem. IETF Internet Draft, draft-puthenkulam-eap-binding-04.txt, October 2003.

[35] Sandeep K. Singhal. Understanding Wireless LAN Security. White Paper, available at `http://home.en2004.com/en2004/pdf/UnderstandingWirelessLANSecurity.pdf`, 2003.

[36] A. Stubblefield, J. Ioannidis, and A. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. Technical Report TD4ZCPZZ, ATT Labs, August 2001.

[37] Jonathan Trostle, Michael Swift, Bernard Aboba, and Glen Zorn. Initial and Pass Through Authentication Using Kerberos V5 and the GSS-API (IAKERB). IETF Internet Draft, draft-ietf-cat-iakerb-09.txt, October 2002.

[38] J. Walker and R. Housley. The EAP Archie Protocol. IETF Internet Draft, draft-jwalker-eap-archie-01.txt, June 2003.

[39] Donald J. Welch and Scott D. Lathrop. A Survey of 802.11a Wireless Security Threats and Security Mechanisms. Technical Report ITOC-TR-2003-101, United States Military Academy, 2003.

[40] Thomas Wu. The Secure Remote Password Protocol. In *Proceedings of the 1998 Internet Society Symposium on Network and Distributed Systems Security*, pages 97–111, San Diego, CA, 1998.

[41] Thomas Wu. A Real-World Analysis of Kerberos Password Security. In *Proceedings of the 1999 Internet Society Network and Distributed System Security Symposium*, February 1999.

[42] Thomas Wu. The SRP Authentication and Key Exchange System. IETF RFC 2945, September 2000.