USENIX Association

Proceedings of the 13th USENIX Security Symposium

San Diego, CA, USA
August 9–13, 2004

Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute

Dirk Balfanz, Glenn Durfee, Rebecca E. Grinter, D.K. Smetters, Paul Stewart
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
{balfanz, gdurfee, grinter, smetters, stewart}@parc.com

Abstract

Combining effective security and usability is often considered impossible. For example, deploying effective security for wireless networks is a difficult task, even for skilled systems administrators – a fact that is impeding the deployment of many mobile systems.

In this paper we describe a system that lets typical users easily build a highly secure wireless network. Our main contribution is to show how gesture-based user interfaces can be applied to provide a complete solution for securing wireless networks. This allows users to intuitively manage the network security of their mobile devices, even those with limited user interfaces. We demonstrate through user studies that our secure implementation is considerably easier to use than typical commercially available options, even those that provide lower security. Our gesture-based approach is quite general, and can be used to design a wide variety of systems that are simultaneously secure and easy to administer.

1 Introduction

Connecting mobile computers together in a wireless network can be largely automated. Today, many default networking configurations allow computers to connect to any wireless access point, to obtain IP addresses, gateway information and DNS server locations automatically through DHCP, *etc.* Mobility of devices makes this autoconfiguration a necessity: when devices are removed from one environment and introduced into another, users should not be burdened with reconfiguring their devices manually.

This situation, however, shifts dramatically when we require our wireless network to be *secure*. A secure wireless network only admits certain (authorized) devices, and protects those devices and their communication from attack. Today, securing a wireless (or indeed, any) network usually requires substantial amounts of manual work. The burden placed on the user ranges from specifying passwords for access points and clients to managing a Public Key Infrastructure (PKI), complete with setting up a certification authority, issuing certificates and installing them on client computers, configuring client security, *etc.* The more secure the network, the more complex the configuration. This means that strong wireless security solutions are often completely out of reach for typical end users.

The difficulties of deploying secure wireless networks is perceived as just one example of the general tension between security and usability, which has led many to believe that there is an unresolvable trade-off between the two. The tension between security and ease of network configuration significantly adds to the cost of setting up and maintaining secure networks. This is aggravated in *wireless* networks, where the lack of physical barriers to access makes strong network security crucial. Consider, for example, a company that has an Ethernet-based intranet that can only be accessed from inside the company building, while the 802.11-based wireless network can be accessed from the public parking lot across the street.

In this paper, we show that security and ease of network configuration do not, in fact, have to be at odds with each other. We describe a method to set up a secure wireless network without any manual configuration. We solve the problem of introducing wireless devices to the network, distributing initial keys between them and pieces of the network infrastructure (*e.g.*, access points) – and thus establishing trust – by using *location-limited channels* [5]. As a result, a user can add a laptop to a secure wireless network by walking up to an access point and physically pointing out the access point to his laptop. The laptop and the access point exchange public keys and other relevant information through the location-limited channel, before proceeding with a completely automated

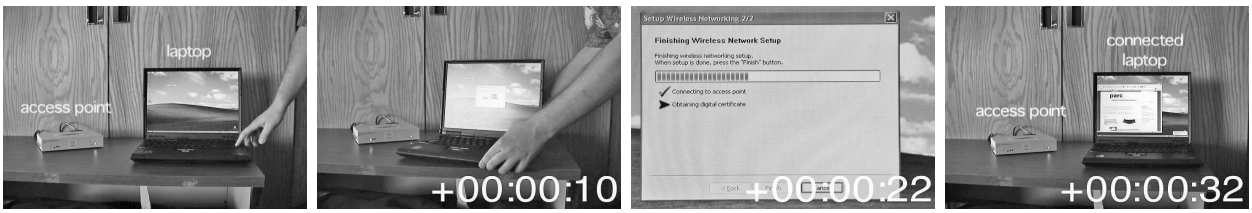

Figure 1. Connecting a laptop to a secured wireless network in 32 seconds. All the user has to do is briefly align the infrared ports of laptop and access point and press the Enter key twice. These are snapshots from a live Network-in-a-Box demonstration.

configuration of the laptop. This includes configuration of the network security settings (as well as more traditional configurations such as IP addresses, gateway information, *etc.*).

To demonstrate the utility of our approach, we designed and built a secure wireless access point and configuration software for mobile devices. Our gesture-based interface reduced the time needed for a user to enroll a laptop into a secure wireless network from over 9 minutes to under 60 seconds. At the same time, we increased the security of that network from (rather insecure) WEP to 802.1x EAP-TLS [17]. Our technology makes it possible for mobile devices to be configured quickly and to easily move to new secure networks.

The rest of the paper is structured as follows. In Section 5 we describe related work. In Section 2, we present background information on gesture-based authentication and wireless security protocols necessary to understand the rest of the paper. In Section 3 we present the design and implementation of an easy-to-use secure wireless network consisting of a single "smart" access point. We have experimentally demonstrated the usability of this system, with results shown in Section 3.4. In Section 4 we show how to extend our approach to configuring large enterprise wireless networks consisting of many commercial off-the-shelf access points. We conclude in Section 6.

2 Background

2.1 Gesture-Directed Automatic Configuration

Authentication is the most basic security problem faced by mobile networked devices: once two devices can *securely recognize* each other, they can then securely exchange data, set network configurations, issue credentials, and so on. Traditional approaches to this problem assume that both devices already participate in some manually-configured shared infrastructure, for example, that they have been configured with identical passphrases, each other's public key, or the public key of a common certification authority. For mobile devices, and new consumer devices brought into the home, this will not be the case. We need a way to allow two devices to communicate securely with each other even if they know nothing about each other *a priori*.

We solve this problem in a simple, easy-to-use manner. A user wishing to initiate communication between his device and another device in the area simply "points out" his desired communication partner, using a *location-limited channel* [5] – *e.g.,* touching the two devices together, or indicating the desired target using infrared, as with a remote control. With this simple and intuitive gesture, the user actually sends a small amount of configuration and cryptographic information – fingerprints of public keys – across this more trusted channel, and the target device sends a small amount of information back. This allows those two devices then to authenticate each other and communicate securely over the network.

There are many types of location-limited channels. As we are sending only public information over this channel, we require of such a channel only that it be very difficult for an attacker to transmit information in that channel without being detected; the channel need not be intrinsically "private" or impervious to eavesdropping. Channels such as infrared or contact give a strong intuitive feeling of "pointing out". A simple, passive USB storage token can be used to exchange authentication information between less mobile devices in a location-limited way. Audio channels allow the exchange of authentication information between a number of devices at once, thus enabling secure group communication [5, 24]. The work presented here builds on infrared location-limited channels.

Location-limited channels often have lower bandwidth and higher latency than typical network media, so both the amount of data exchanged and the number of rounds of communication must be kept to a minimum. In the work presented here, two devices exchange cryptographic digests of their public keys over the location-limited channel, plus a small amount of network infor-

mation. Subsequent communication takes place over the less secure (wireless) network, and is secured using standard public key protocols, where trust in the public keys is established by matching their cryptographic digests with those received on the location-limited channel. This allows the creation of a secure tunnel in which any number of rounds of more bandwidth-intensive network configuration and network provisioning protocols can be executed, such as protocols for requesting and delivering digital certificates.

This gesture-based approach to authentication supports a variety of trust models. Participants can be configured to allow only the last party with whom they performed a location-limited exchange, to set up a secure, authenticated network connection to them; or the trust established through the location-limited exchange may "expire" in a short amount of time, as appropriate for the application.

In summary, a user gesture briefly establishes a location-limited channel, which in turn allows software to bootstrap a secure tunnel for the provisioning and automatic installation of network and security configuration information. We refer to this process as *gesture-directed automatic configuration*.

2.2 Wireless Security Protocols

The most popular standard for wireless networks is IEEE 802.11.[1] Adoption of 802.11 wireless networks for critical applications has been hampered by high-profile reports of security flaws. The security and encryption component of the original 802.11 standard, WEP ("Wired Equivalent Privacy"), requires clients and access points to share a single secret passphrase or key which they use to encrypt all wireless traffic. Unfortunately, highly-publicized attacks on WEP [7, 11, 36] have completely discredited WEP as an effective security mechanism.

To address these problems, industry and standards bodies are proposing new security protocols for 802.11, which will appear over the next few years. These protocols – "Wi-Fi Protected Access" (WPA [2]) and the 802.11i standard [18] combine use of an existing standard protocol, 802.1x [17] which is used to authenticate devices and users wishing to join the wireless network, with the ability to automatically derive and update encryption keys to secure wireless data. These protocols differ primarily in how the data is encrypted with these keys, and provide different degrees of backward compatibility with deployed hardware. The 802.11i standard

[1] 802.11 comes in a variety of subtypes with differing frequency and bandwidth characteristics. Although the implementation discussed here uses only 802.11b, all results are general and apply equally well to 802.11a, 802.11g, *etc.*.

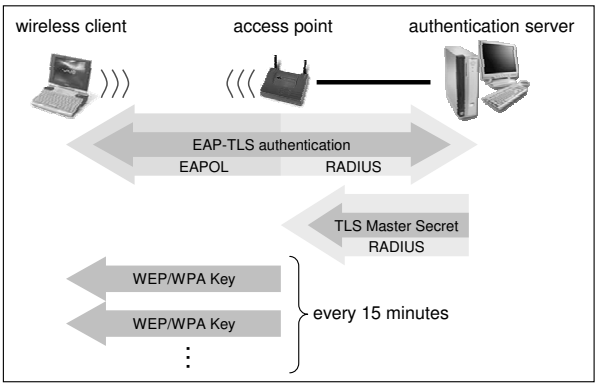

Figure 2. Roles and message flows with 802.1x authentication using the EAP-TLS protocol.

is intended as the final standard for security in 802.11 wireless networks, while WPA is intended as a temporary backward-compatible intermediate step, and is based on a draft of 802.11i.

The core of these two protocols – 802.1x-based authentication combined with automatic frequent update of the keys used to secure wireless data – has begun to see widespread use in advance of WPA and 802.11i deployment. As 802.1x is currently the most secure available option to transparently secure an 802.11 wireless LANs, we chose it for our implementation. As our work focuses on configuring trust for the 802.1x authentication protocol common to all three, our results immediately generalize to both WPA and 802.11i. In fact, our approach generalizes to allow easy configuration of IPsec-based wireless LAN security (see Section 3.2.2), or any other security mechanism authenticated using a Public Key Infrastructure.

The 802.1x Protocol. The 802.1x security standard [17] defines a mechanism for providing access control for any IEEE 802 LAN, including 802.11 wireless networks. In an 802.1x-compliant wireless network (as shown in Figure 2), each access point plays the role of an *authenticator* forcing every client to authenticate before allowing access to the wireless network. Prior to successful authentication, a client may only send 802.1x protocol messages to the access point (AP); it is not allowed to send any data frames. The access point itself is not capable of making any authentication decisions. Instead, it forwards all 802.1x messages from the client to a back-end *authentication server* (AS) via the RADIUS [31] protocol; the AS checks the credentials of the client and indicates to the access point whether the client is authorized to access the network. Optionally, the AS provides

the AP information which allows it to securely transmit unique short-lived data encryption keys to each client [8]. This latter step is used in the wireless context to protect client data transmitted over the air.

The 802.1x protocol is "pluggable", allowing any one of a wide number of authentication protocols to be run under the wrapper it uses – EAP, the Extensible Authentication Protocol [6]. The most secure of these is EAP-TLS [1], a wrapper around the widely deployed TLS (SSL) key exchange protocol [9], which requires all wireless clients and the network infrastructure itself to use digital certificates for authentication. Keying material exchanged as part of this TLS handshake is then used to automatically give authenticated clients encryption keys that they can use to secure their traffic on the wireless network. These keys are updated frequently without human intervention.

Current 802.1x deployments frequently opt for password- or shared secret-based authentication options, in order to avoid the difficulty of issuing a digital certificates to each client device. However, in addition to its greater security, EAP-TLS has a number of desirable features. First, as every client (and the authentication server) possesses a unique digital certificate and corresponding private key, access for individual clients can be revoked in case of compromise; in contrast, shared-secret variants of EAP [6] require client machines to be rekeyed *en masse* whenever the compromise of a single machine occurs. Password-based EAP protocols are subject to the same password-guessing attacks as other password-based systems [32], and can additionally be subject to man-in-the-middle attacks [4]. Furthermore, digital certificates and private keys are usually stored on disk protected by both the operating system and the user's password; they are unlocked as soon as a user logs into a machine. This provides quick, frequent and automatic authentication of both the user and the device, desirable for security and to allow seamless roaming. This approximates more closely a single sign-on system than does a password-based authentication method, where reauthentication requires caching or frequent re-entry of the password.

Once a digital certificate and other configuration information has been installed on every client, an 802.1x-secured wireless network using EAP-TLS is extremely easy to use – it requires no user intervention. The clients (and the authentication server) use their certificates and corresponding private keys to authenticate themselves to each other, requiring no input from the user.

Unfortunately, it is the process of enrolling every device in a common PKI – provisioning the digital certificates – that is a daunting task for system administrators, and out of reach of typical end users. In our organization, for instance, the process of setting up a PKI for use with 802.1x/EAP-TLS required each user, for each device, to follow a minimum of thirty-eight separate documented steps, requiring several hours of end-user time over two days. These consist of using a very typical web-based enrollment interface to request a certificate, and then configuring Microsoft's standard 802.1x client to securely access a particular wireless network.

Clearly, a much simpler solution would not only be theoretically interesting, but practically useful. In the next section, we describe how to apply gesture-directed automatic configuration to simplify setting up a secure wireless network consisting of a single access point; in Section 4, we describe our solution for an enterprise-scale wireless network.

3 A "Network-in-a-Box"

Our "Network-in-a-Box" consists of a custom-built wireless access point (NiaB AP), providing a complete 802.1x-secured wireless network, and software for client devices to enroll in that network. After a gesture-directed automatic configuration step (during which users point their device at the access point), client devices are fully configured to participate in the wireless network.

During this process, the access point issues digital certificates for use in the EAP-TLS-based wireless security system. The user is managing a small PKI without even realizing it. Instead of burdening the user with complicated certificate management semantics, we provide a simple and intuitive security model: A device can participate in the wireless network if and only if, during enrollment, it can be brought into close physical proximity of the access point. For example, if a NiaB AP were to be deployed in a home, then someone wishing to gain access to its wireless network would have to be able to physically enter that home. (Especially concerned users might even lock their NiaB AP in a closet.) This is a simple, intuitive trust model that seems quite effective for many situations.

3.1 User Experience

Imagine a user who wants to set up a secure wireless network to use with his laptop. Our user starts by bringing home a new NiaB AP (our prototype NiaB AP is the small white box shown in Figure 1). When he plugs it in for the first time, it initializes itself and autoconfigures the network services it will provide.

To add his laptop to the NiaB network, the user starts a NiaB enrollment application on that laptop. The application can be either pre-installed by the laptop vendor, or provided with the NiaB AP.[2] The enrollment software (shown in Figure 3) asks the user to "point out"

[2] If a USB token is used to exchange authentication information (see

Figure 3. Screenshots from the Network-in-a-Box client software for Microsoft Windows XP™.

the NiaB AP whose network he wishes to join. In our implementation, he does this using the infrared port on his laptop. For a second or two, the devices exchange a small amount of information over infrared; then, the user is prompted to separate the devices to continue the automatic configuration of the laptop. After a few more seconds, the user is informed that his laptop is ready to use. These simple steps provide a previously unconfigured laptop with everything needed to get a "network dial-tone".

If the user later wants to add his laptop to another secure network, he simply runs the client software again. As the client application contains no pre-configured information about a particular network, instead getting all the information it needs about whom to trust and what network to join via the infrared exchange, it can be used repeatedly to configure the same laptop for multiple secure networks. Once credential information for each network of interest has been configured, the user can switch between them "on the fly" using whatever location-management facilities his operating system provides. For example, our Windows XP-based implementation targets the built-in "Wireless Zero Config" service, which automatically switches between configured wireless networks as the laptop moves around, without requiring any user intervention.

3.2 System Design

As described in Section 2, an 802.1x-based wireless network contains a number of components: one or more access points, an authentication server making determinations about what clients are allowed to access the network, and, in the case of EAP-TLS, a certification authority, which issues certificates. As shown in Figure 4, our NiaB AP contains all three of these components, as well as general system services such as DHCP and a firewall, and a http-based management interface. It is therefore able to provide EAP-TLS-secured network access to clients without requiring other network components. The NiaB AP also contains a novel component – an "enrollment station". This component is responsible for handling requests for automatic client configuration made over location-limited channels like infrared or USB.

System Initialization. When the NiaB AP is switched on for the first time, the access point component automatically chooses itself a wireless network name and channel. (The network name is "NiaB Network + <number>", where the number is chosen randomly between 1 and 1000 to keep your NiaB AP from interfering with your neighbor's.) The certification authority component generates a root key pair and root certificate. By default, the NiaB AP will request its own IP address through DHCP, while providing IP addresses for its clients through DHCP from a non-routable address pool. If the NiaB AP does not have an external connection to the Internet, it still provides a useful wireless network to its clients, allowing a group of users to easily configure a secure local wireless network.

We also built in a reset feature to return a NiaB AP to a "new" state. The reset feature is activated by pressing a (physical) button on the device. Upon reset, the NiaB AP disconnects and removes all records of existing clients. It generates a new network name, root key pair, and root certificate. This automatically invalidates the certificates of previous clients, as they are signed by the old root key, rather than the new one.

Device Enrollment. When a user initiates enrollment by executing our enrollment software on a client computer, the software performs a number of steps. First, it creates a cryptographic key pair, which consists of a private key that will remain encrypted on his computer, and a public key that eventually (as described below) will be encapsulated in a digital certificate for use in authenticating to the network.

Section 2.1), it can also be used to install the software.

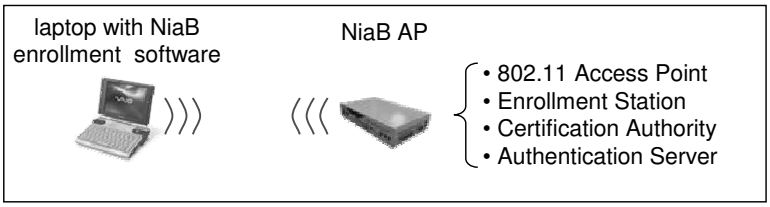

Figure 4. The Network-in-a-Box provides the functionality of several network components.

When the user makes the temporary infrared connection, his computer and the NiaB AP exchange what we call *preauthentication information*: the enrollment station component in the NiaB AP sends the name of its wireless network (802.11 SSID) along with the SHA-1 digest of its public key to the user's computer. The user's computer responds with a the SHA-1 digest of *its* (newly created) public key.

Since the user's computer now knows the wireless network name, it can contact the NiaB AP over the higher-bandwidth 802.11 network, and the infrared connection between the two devices is no longer necessary.

Next, the enrollment station component on the NiaB AP and NiaB enrollment software on the user's computer run *EAP-PTLS* over the wireless link. EAP-PTLS is an EAP plug-in protocol we designed specifically for provisioning network access for NiaB client computers, and is described in greater detail in Section 3.2.1. At this point the NiaB AP disregards all traffic from the client computer except for EAP messages, as required by the 802.1x protocol (see Section 2.2).

The EAP-PTLS exchange encapsulates a TLS handshake, in which the client computer and the NiaB AP present each other their full public keys and prove that they possess the corresponding private keys. The public keys are automatically checked to make sure they match the digital fingerprints previously exchanged over the infrared location-limited channel, allowing the client and NiaB AP to authenticate each other. The AP can confirm that the client performing the EAP-PTLS handshake over the wireless network was indeed physically present at the NiaB AP earlier, and the client can be sure it is talking to the AP of interest.

Once the TLS tunnel has been established via EAP-PTLS, the enrollment software sends a certificate request to the NiaB AP through this tunnel. The enrollment station component passes this request on to the certification authority component, which creates a certificate for the client and returns it over the TLS connection. The enrollment software on the user's computer installs the new certificate, and automatically configures the laptop's 802.1x security client software to use it.

Using the Network. At this point, the laptop has everything it needs to participate fully in the secure wireless network. Normal authentication occurs via an 802.1x authentication protocol exchange with the authentication server component on the NiaB AP. Since the client computer has just been configured with a digital certificate, it is able to use EAP-TLS to authenticate to the wireless network. This is the final "steady-state" for the client, requiring only standard 802.1x client software; our enrollment software is no longer needed and can be either removed from the client computer, or used again at a later point to add the laptop to additional secure networks.

Once the lower-level 802.1x authentication succeeds, higher-level network provisioning takes place: The DHCP server on the NiaB AP assigns an IP address to the client computer, along with addresses of DNS servers, IP gateways (both, in fact, pointing to the NiaB AP), *etc.* Again, we don't provide any special software for this step on the client side, and instead rely on standard software built into the operating systems of the client computers.

3.2.1 EAP-PTLS Protocol

The EAP-PTLS protocol is a simple variant of the standard EAP-TLS authentication protocol [1]. Like EAP-TLS, the wireless client and authentication server (in this case, the NiaB AP itself) perform a standard TLS exchange: exchanging certificates, demonstrating that they each possess the private key corresponding to the public key in that certificate, and establishing a secure tunnel for further communication. However, in the case of EAP-PTLS, the certificates used are self-signed, and are simply carriers for the public keys whose fingerprints were previously exchanged over the location-limited channel. Both the client and server are satisfied with the authentication exchange only when, at the conclusion of a successful TLS handshake, the key used by each party matches the fingerprint previously received over the location-limited channel.

In EAP-TLS, the TLS handshake is only used for authentication and for agreement on keying material that is used to derive keys to protect future wireless traffic. No data is actually sent down the secure tunnel. In EAP-

PTLS, we do send data over the secure TLS tunnel. We use the Certificate Management Protocol (CMP) [3], a standard protocol for requesting and retrieving certificates, to allow the new client to send a certificate request to the authentication server through this tunnel. This request is forwarded to a Certification Authority (CA) internal to the NiaB and immediately approved and signed by the root key in the NiaB. Although the design choice of using CMP may seem like overkill in this scenario (the internal NiaB CA always approves incoming certificate requests), it allows us to easily generalize our solution to the enterprise scenario described in Section 4.

3.2.2 "Phone Home" Service

Our solution for provisioning 802.1x-based security generalizes well to other security applications such as Virtual Private Networks (VPNs). Consider a user who has successfully configured a home network, and then wishes to access the devices and services on that network while he is away from home. This is a feature not typically provided by consumer home gateway devices. A more sophisticated gateway device or firewall machine might allow the user to configure a password-based approach to providing remote access to his home; access to which is often not encrypted.

We added features to the NiaB AP to make it easy to allow devices to access the home network using an IPsec-based VPN. This VPN uses the same certificate as was issued by the NiaB AP when the device enrolled in the home network. We have implemented automatic configuration of such a service, which we call "Phone Home", as part of our Windows XP™ NiaB client enrollment software. The Phone Home service is set up at the same time as the enrollment in the wireless network, requiring no additional effort from the user.

As part of device enrollment, the NiaB enrollment software configures appropriate IPsec policies and Remote Access Service (RAS) Phonebook entries to allow the client computer to initiate a standard Windows-style L2TP-based IPsec VPN connection back to the external IP address of the enrolling NiaB [38]. This new "Phone Home" VPN connection appears as a standard "Network Connection" on the user's desktop. Clicking on it moves their machine virtually "inside" their home network in a secure fashion. All communication with the home network is encrypted and authenticated, and the remote device automatically receives a new, "virtual" IP address inside the home network's address space. All communication now goes through the firewall provided by the NiaB.

Provisioning and access to the "phone home" service is controlled using the NiaB management interface (see Section 3.2.3 below). Although clients are configured to be able to use the service by default, such access can be disabled on a client-by-client basis at the NiaB, independent of the access those clients have to the NiaB's local wireless network. This is useful, for example, in the case where a home user decides that a guest may get access to the wireless network, but should not have access to network resources from outside of the home.

3.2.3 System Management and Client Revocation

In order to allow the user to monitor the status of his NiaB system, alter any of the autoconfiguration parameters inappropriate for his situation (*e.g.*, to turn on PPPoE if his ISP requires it, or to configure a static IP address for the external interface of the NiaB AP), we provide a simple web-server based management interface. This interface is largely similar to that provided by standard commercial access points, though is easier for users to find, and does not require the use of a password for secure access.

Through the NiaB's DNS proxy (configured to be the DNS server for all NiaB clients), we redirect any entries in the ".niab" domain to the NiaB AP itself. Therefore, entering "http://www.niab" in a web browser automatically takes you to the NiaB management page, without requiring the user to enter a specific configuration IP address. A shortcut to this page is provided as part of client configuration. Since individual NiaB clients can be recognized by their certificates, policy configuration can be used to allow only a subset of those clients to access the NiaB management interface (*e.g.*, the first client to join the network, and any other clients he specifically enables). This use of client authentication to control access also saves the user from having to change and remember an administrator password.

Most importantly, this configuration interface allows a user to permanently revoke access to NiaB services (wireless network and VPN) to any client, by revoking that client's certificate – this is done simply by clicking a button in the management interface next to the name of the device to be removed from the network. At that point, a new Certificate Revocation List is automatically generated by the NiaB AP, and all currently-connected clients are asked to reauthenticate. To temporarily restrict client access, the management interface allows wireless and VPN access to be separately enabled and disabled for each enrolled client. While this interface may not be as directly intuitive as our gesture-directed enrollment interface, it does provide useful functionality – revocation of individual devices. We have not yet experimentally studied how usable this simple interface is, but we may explore more intuitive alternatives in future work.

3.3 Implementation Details

3.3.1 NiaB Access Point

We have implemented our NiaB access point on the OpenBrick platform [13] – a small, x86-based computer providing most of the ports and peripherals standard to larger PCs. We have modified the hardware to add an infrared port to the front of each device, along with a red LED that is used to inform the user when the NiaB AP is transmitting infrared data. This LED provides valuable feedback to the user as to whether they have lined the device infrared ports up properly.

The NiaB access points are running a modified distribution of RedHat Linux 9.0, and release 2.6.4 of the Linux kernel. This version was selected for its greater stability and for its native support for IPsec. The Linux kernel provides native firewalling capabilities, which we configure to provide protection from the Internet for the hosts on the NiaB-provided wireless network. The NiaB access points also act as DHCP servers and DNS caches for their clients.

Implementations of our base protocols – gesture-directed authentication using location-limited channels, the EAP-PTLS enrollment protocol, certificate issuance functions, and Certificate Management Protocol (CMP) messaging used to request certificates, *etc.*, are written as libraries in C++ to facilitate reuse. All cryptographic operations are implemented using OpenSSL 0.9.7.

Access point functionality for the NiaB access points is provided using the HostAP project's access point software [29], slightly modified to provide the additional logging and management interfaces we require. We set up the HostAP access point software to be a 802.1x passthrough to an internal RADIUS server for authentication decisions (see Figure 4).

The RADIUS server we use is a modified version of FreeRADIUS 0.8.1 [28]. FreeRADIUS itself provides a pluggable implementation of the EAP protocol architecture, making it easy to add implementations of new EAP subtypes. We use that architecture to add a C++ implementation of EAP-PTLS (see Section 3.2.1). We modified the implementation of EAP-TLS provided by FreeRADIUS to add additional configuration and support for the use of Certificate Revocation Lists (CRLs), and to access the client authentication information controlled through our management interface.

To provide "phone home" functionality, we use the IPsec support present in the Linux kernel (version 2.6), and modified a version of the IKE daemon raccoon to check both CRLs and our client authentication information database to verify clients.

Our management interface is provided using mini_httpd 1.17 [33], a small web server, chroot'ed for greater protection, coupled with a variety of Perl and Python scripts. Standalone certification authority and CRL generation functionality is implemented in a C++ library which uses OpenSSL to handle cryptographic operations and certificate and CRL formatting.

3.3.2 NiaB Client Software

The client application described above is implemented for Microsoft Windows XP™, and has been tested successfully on a wide variety of laptop hardware using a number of different 802.11 client cards, both internal and external. A command-line client for Linux has been tested on a somewhat smaller range of laptop hardware.

We implemented client-side protocols and routines for CMP, EAP-PTLS, location-limited channels, and certificate handling as portable libraries written in C++. All cryptographic operations are implemented using OpenSSL 0.9.7.

To support frame handling for sending and receiving raw Ethernet packets (necessary for executing EAP-PTLS over the wireless connection), we use the NDISUIO API [34] for Microsoft Windows™ and libdnet [22] and libpcap [23] for Linux. The user interface of our Microsoft Windows™ uses Microsoft Foundation Classes. For basic wireless support, we use the NDISUIO API for the Microsoft Windows XP™ client, and the Linux Wireless Extension and Wireless Tools [12] for Linux. Our NiaB client software sets up 802.1x support for Windows XP™ using the Wireless Zero Configuration Service [16] and for Linux using the xsupplicant 802.1x client software package [27].

Again, our software is used only when a client is enrolling in a new wireless network and does not interfere with the normal day-to-day use of such networks. Though our software supports being used repeatedly to add the client device to more than one secure wireless networks, switching between those networks in operation is then done using the mechanisms provided by the operating system in use.

3.4 User Studies

We undertook a series of usability studies [10] both to test if our system actually makes it easier to set up a secure wireless network, and to obtain feedback to iteratively improve our design.

3.4.1 Procedure

Our usability tests had two objectives. First, we wanted to know whether our system, NiaB, allowed users to easily and securely connect to a wireless network. Our second objective was to compare our solution against a com-

	Commercial AP			NiaB		
	Time (min)	Steps		Time (min)	Steps	
Avg	9:39	14		0:51	2	
	Ease	Satisfaction	Confidence	Ease	Satisfaction	Confidence
Avg	3	3	2	1	1	1

Table 1. User studies comparing the stand-alone Network-in-a-Box to a commercial access point.

mercially available alternative. We picked a commercially available access point based on several criteria: it should be designed for end users, not enterprises; it should be a market leader; and, enrollment should occur on the same operating system as our solution (to avoid confounding variables by switching platforms).

Usability tests assign tasks to users to see whether they can complete them successfully and to learn what errors they make. The task we chose was to ask a user to connect a laptop to a secure wireless network. Though practical deployments of our client software were done on a wide variety of laptop hardware and wireless network cards, we asked all participants to use the same laptop in order to limit setup time and variability in our quantitative studies. The laptop came pre-loaded with the setup software for both NiaB and the commercial AP. In both cases, the setup software provided a "wizard"-style interface for the user. The commercial AP's wizard required 10 steps, the NiaB's wizard required two (each stage in the connection process where the user has to make a decision was counted as a step).

Subjects were asked to connect to both access points, one after the other. Users were timed how long it took them to complete each connection task. We also counted how many errors they made and how many steps they took.

We selected our subjects from a pool of our coworkers. To avoid a bias in our data towards people with significant computer science skills we recruited broadly from both the research and administrative staff. Further, we administered a screening questionnaire to ensure that we selected subjects with a broad range of backgrounds. We screened for education (technical vs. non-technical) and experience (wireless network owned and administered, no wireless network and never set up). We selected the broad range of subjects to avoid an emphasis towards reduced times (for both the commercial and NiaB AP) that would derive from either educational background or experience with wireless networking technologies.

To avoid potential learning bias (being able to connect more quickly the second time than the first based on new knowledge) we selected an even number of participants, split them into two groups, one of which connected to the NiaB AP and then the commercial AP and the other who connected to the commercial AP and then the NiaB AP. After each connection activity participants were asked to rate their experience in terms of ease, satisfaction, and confidence.

Our testing was done in two iterations. We recruited six subjects for the first iteration. We picked six subjects because previous research shows that five subjects reveal approximately 80% of the usability errors in a system [26]. In addition to comparing the commercial AP and NiaB, the first iteration was also used to refine the NiaB user interface. In the second iteration subjects evaluated a revised NiaB interface using the same task as the first iteration. By keeping the task design the same we were able to compare across the two studies, and the results from the second iteration increased our chances of finding almost all of the usability errors.

3.4.2 Results

The results from the two iterations are shown in Table 1. They show that users took much less time (approximately a 10x speed-up) on average to connect to NiaB AP than the commercial AP. NiaB also required fewer steps – points where the user has to make decisions – than the commercial AP. More significantly, on average users took two steps to join the NiaB network, the same number needed to enroll correctly. This was not true for the commercial AP, for which users took an average of 14 steps – 4 more than intended. In other words, users were making errors in the set up process for the commercial AP and frequently having to repeat steps and recover from mistakes.

The likelihood of making errors and the number of steps involved in the commercial AP set up task contributed to users ratings of ease, satisfaction, and confidence. On scales of 1 (most positive) to 5 (most negative) users rated ease of task, satisfaction in the experience, and confidence that they could do it again, more highly for NiaB. These positive feelings towards NiaB were borne out in qualitative interviews, too.

One advantage that iterative usability testing offers is

the opportunity to identify and fix problems with the interface. The first iteration uncovered two usability issues. First, although people managed to successfully use the location-limited channel, they did not realize that they could move the laptop away from the access point once the initial data was exchanged. Second, people did not always know when they had actually finished the task and could use the network. These findings allowed us to redesign the interface, and retest it with users. Results from our second iteration show that we provided more appropriate feedback to let users know to unalign the infrared ports after the location-limited data exchange, and communicated more effectively when they were completely finished.

4 Securing Enterprise-Scale Networks

We extended our easy-to-use approach for enrolling in secure wireless networks to enterprise-class networks with many access points. We deployed our enterprise solution to handle enrollment in the wireless security system of a small enterprise consisting of approximately 250 users.

There are two important differences between enterprise-class networks and the small networks we have considered previously. First, their architecture is different – enterprise networks have many access points communicating with a central backend authentication infrastructure. Second, enterprise networks have considerably more complex security requirements than are present in the home.

We address these differences by encapsulating our gesture-directed enrollment functionality in one or more "enrollment stations" – usually a simple box, or PC, configured to allow gesture-directed user authentication over one or more location-limited channels, but not itself an access point. Depending on the security needs of the enterprise, this enrollment station can implement security requirements considerably more sophisticated than that used in the stand-alone case.

By placing the enrollment station in a locked room to which only employees of the enterprise have access, one ends up with an intuitive security model very similar to that used in the stand-alone case. By adding security cameras monitoring the enrollment station, an enterprise adds the ability to audit its use after the fact. Or an enterprise may want a member of the IT staff to approve each user enrollment manually, *e.g.,* after checking the user's employee badge, or entering additional configuration information to be added to the enrolling device.

An important design goal for our enterprise solution is to integrate with existing off-the-shelf commercial devices and software. In our system, the access points, authentication server, and certification authority can all be commercial off-the-shelf, knowing nothing about any of our configuration protocols. At the same time, we provide opportunities for system administrator control and intervention that are desirable in the enterprise setting.

4.1 User Experience

The user experience of enrolling in the enterprise version of our system is similar to that of the stand-alone NiaB system. The user physically brings her new mobile device to one of perhaps many enrollment stations distributed throughout the enterprise.

A user accessing the enrollment station performs a brief location-limited exchange, and is then told than an enrollment request has been submitted on her behalf. She then leaves the enrollment station. As mentioned above, an IT staff member may want to further review and approve her request off-line, or perform additional configuration or processing, so it may take some time for her certificate request to be fulfilled.

All further device configuration takes place over the wireless network, using any of the enterprise's access points. The user does not need to visit the enrollment station again. She may at will re-run the client enrollment application (possibly prompted by an email from an administrator) to check whether or not her enrollment request has been approved; eventually the software indicates that the request has been approved, and the certificate and configuration information is installed automatically on her device. She may then begin using the secure wireless network normally.

4.2 System Design

In this section, we describe the changes in design from the stand-alone NiaB required to make an enterprise solution (as shown in Figure 5). We separate the components built into the stand-alone NiaB AP. The access points, certification authority, and authentication server functionalities are standard solutions that do not need to be aware that they are participating in our system. The enrollment station is designed to handle our EAP-PTLS protocol and speak with a Certification Authority for certificate management.

Client Enrollment Software. The client configuration software is similar to what we use in the stand-alone NiaB case. The most significant change is that the client software is written to be aware of the fact that a request for a certificate may not be approved immediately – waiting for the approval of a system administrator could delay enrollment.

In the stand-alone NiaB system, the Certification Authority either immediately grants or rejects the certificate,

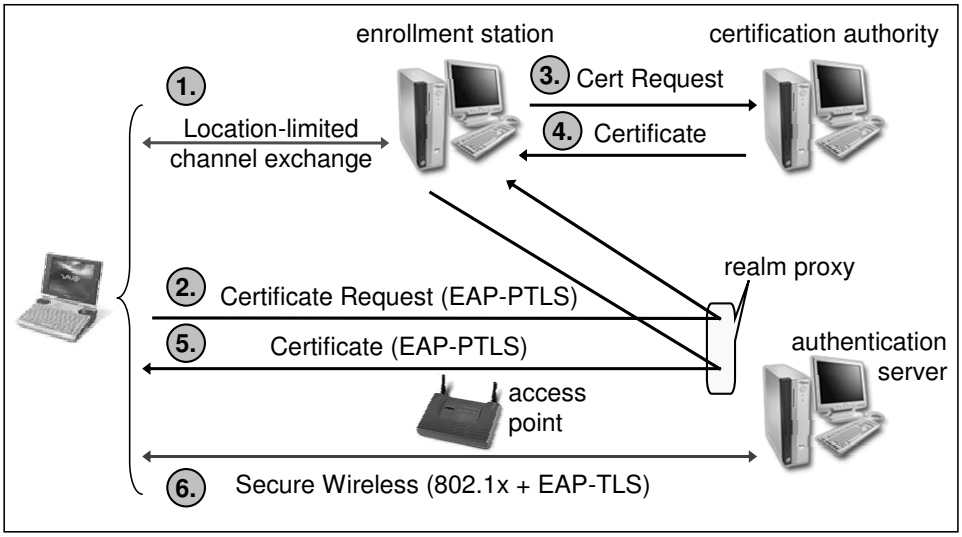

Figure 5. Message flows in the enterprise version. Only the enrolling laptop and the "enrollment station" are aware that they are participating in a NiaB-enabled network.

and the authentication server returns the resulting certificate or rejection message to the client. In the enterprise version, however, the certificate request enters a database of pending requests that might need to be examined, approved, or rejected by an administrator.

Because the client and enrollment station are speaking the Certificate Management Protocol inside of the EAP-PTLS tunnel, the enrollment station is able to send to the client a request number and a result code indicating that the request is pending. Subsequent execution of the client enrollment software polls for this request number, also using CMP messages in the EAP-PTLS tunnel. Until the request has been approved, the client receives an indication that it is still pending, and can repeat the request later (see Figure 5).

We note that as long as the server and client cache the information they exchanged over the location-limited channel, they do not need to repeat a location-limited exchange. These later attempts to retrieve a certificate that has already been requested can be performed over the wireless network without any intervention by the user. Though the authentication exchange over a location-limited channel must be done in physical proximity to the enrollment station, all further interaction with each client can be done using any access point in the infrastructure.

Once approved, the certificate is returned to the client the next time the client polls for it; the client enrollment software automatically configures the client device to use the new wireless network just as in the stand-alone case.

Enrollment Station. We configure the standard off-the-shelf authentication server to forward EAP-PTLS traffic to the enrollment station. We accomplish this by taking advantage of RADIUS proxying (during enrollment the client claims to belong to a recognizable special realm, "host@preauth"). Authenticated clients then engage in the normal EAP-PTLS enrollment protocol with the radius server running on the enrollment station, but the resulting certificate requests are then forwarded to an enterprise CA. Since we are using EAP-PTLS, the enrollment station does this only for clients it trusts due to prior interaction over the location-limited channel. When the client checks to see whether its certificate has been issued, its EAP-PTLS exchanges are again forwarded to the enrollment station, which retrieves the issued certificate from the enterprise CA.

4.3 Implementation Details

In this section we describe the particular implementation we are using in our organization; obviously, given the focus on interoperability with commercial software, many of these components would vary from installation to installation.

Access to the wireless network is provided by standard commercial access points; the Authentication Server we use is a commercial RADIUS server (Funk Software's Steel Belted Radius^{TM}). This AS is configured, using standard RADIUS proxying facilities, to forward EAP-PTLS messages from clients requesting enrollment in the system to the RADIUS server running on the enrollment station.

	Standard Enrollment			"Enterprise NiaB" Enrollment		
	Time (min)	Steps		Time (min)	Steps	
Avg	140	38		1:39	4	
	Ease	Satisfaction	Confidence	Ease	Satisfaction	Confidence
Avg	5	4	4	1	1	1

Table 2. User studies comparing our "Enterprise NiaB" solution to a typical commercial alternative.

For our current deployment, we are using a modified stand-alone NiaB as our enrollment station. It runs a copy of FreeRADIUS that responds to only one EAP type, namely our EAP-PTLS protocol. It listens for client authentication requests over location-limited channels, thus limiting initial requests for enrollment to devices with physical access to the enrollment station. It matches the authentication information it receives over these location-limited channels with the public keys used in requests for PTLS authentications forwarded by the main Authentication Server.

Our enterprise CA was developed in-house. It is written in Java, and provides a web-based interface used by both people and the EAP-PTLS enrollment protocol to post certification requests. Each certificate request that comes in from the NiaB enrollment station must be reviewed, edited and approved by a human before the certificate is issued. Once the certificate has been issued, the user owning the device receives an e-mail message, indicating that they should re-run the NiaB enrollment software on their device to retrieve their certificate and finish device configuration. This step can be done at any physical location within our enterprise.

4.4 Enterprise User Studies

We deployed this software as the primary enrollment mechanism for our secure enterprise wireless network. Anecdotally it seemed a success – not only were end users happier with the process, but our IT staff preferred to use the system to enroll laptops they were configuring for other users. To confirm these perceptions, we performed quantitative user studies.

4.4.1 Procedure

In order to see whether our system made enrolling in an enterprise secure wireless network easier, we undertook a comparative usability test. Like the stand-alone NiaB test described in Section 3.4, we wanted to see whether our enterprise solution was easier to use than a currently commercially available alternative, described briefly below. We observed five individuals conducting both types of enrollment. We followed the same subject selection protocol as previously described, but although we used the same subject pool we recruited different subjects for this second test.

4.4.2 Control – Standard Enterprise Enrollment

We looked at users performing standard procedures for requesting a digital certificate, installing that certificate, and then configuring a standard commercial 802.1x client to use that certificate to authenticate to a particular network. The interface for requesting and installing certificates was a web-based one, very similar to that used by commercial Certification Authorities such as Verisign, or Microsoft's Certificate Server software. The 802.1x client software was provided with Microsoft Windows XP™, and comes with a dialog-based graphical configuration interface. Users were provided with extensive documentation as to how to perform all enrollment and configuration steps, complete with screen shots. The total procedure required 38 steps.

4.4.3 Results

The results from two iterations of study are shown in Table 2. The first interesting result was the sheer amount of time end users took to enroll in the 802.1x-secured wireless network using the standard interface – an average of 140 minutes (2 hrs and 20 mins). We found this result very surprising. This observation underscores the fact that the intuitions of domain experts – who could perform the same steps in minutes, if not seconds – are not always useful in evaluating system usability, and the importance of obtaining direct feedback about users' experiences with security systems.

Using our gesture-directed enterprise solution, the time to enroll dropped dramatically, from 140 minutes to under 2 minutes. The total number of steps to request and install a client certificate and configure the client device was reduced from a total of 38 steps to 4 steps. More significantly, users reported making a variety of errors in the 38-step process, unlike the enrollment procedure of our enterprise solution, where they made none.

The reduction in time and the fact that users did not make errors contributed to the users' ratings of ease, satisfaction, and confidence. On scales of 1 (most positive) to 5 (most negative) users rated ease of task, satisfaction in the experience, and confidence that they could do it again more highly for our "enterprise NiaB" wireless enrollment system.

4.5 Discussion

While the enterprise version of our system is designed to meet the fundamental architectural and security constraints of almost any corporate network, we have only been able to experimentally test it in a relatively small enterprise consisting of about 250 users. It remains to be seen whether such a system could scale to meet the demands of a community of 10,000 users supported by a significant IT staff. Traditionally, approaches like ours are thought inappropriate for enterprise environments: large enterprises often prefer completely automated configuration approaches without any per-machine interaction. They also usually have all machine configuration performed by administrators, rather than end users, thereby potentially putting less of a premium on usability.

To counter these arguments, we first point out that the use of certificate-based authentication methods and EAP-TLS provides greater security than both password-based approaches and automatic configuration approaches without per-device authentication. The latter approach usually encodes all necessary authentication information in a static software install replicated on each machine. This can include things like the certificate of the access point or authentication server, allowing one-way authentication of the infrastructure by the client (which presumably authenticates using a password). In more dangerous approaches, such installers can include secret keys shared across all devices in a network, or even a certificate and private key for the user.

We find this approach unsatisfactory for a number of reasons. First, it requires the user to download customized software for each network they want to join. In contrast, our client obtains all the information it needs to configure a particular network from the AP or enrollment station – it can be re-used to enroll a device in multiple networks. The fact that our client software is "generic" in this way means that it could be pre-installed by an operating system vendor, without requiring further customization for a particular network. Second, blindly downloading enrollment information or keys to a potential new client in a customized installer may make it easier to configure that client to use a network, but it doesn't solve the fundamental trust assignment problem – authenticating that a particular device ought to be in fact the one to receive those keys. At best, this problem can be sidestepped by requiring that client to authenticate using a previously-existing password infrastructure. Our goal was to allow easy, secure enrollment in a wireless network even in the case where no pre-existing trust infrastructure existed.

Our approach can also be very appealing even in enterprises where all new machines are initially configured by an administrator. First, we find anecdotally that even the experienced systems administrators in the small enterprise in which we deployed our system vastly preferred to use it to configure new devices for other users than the previous, manual option. Given the increasing demands on administrators and the fact that many of them are not security experts, increased usability of security can be valuable to them as well.

Second, in large organizations, administrative tasks such as WLAN enrollment will happen repeatedly over the lifespan of a given device – returning it to an administrator every time is inconvenient. Increasingly, users introduce personal devices, such as PDAs, into their workplace. System administrators do not have the time and facilities to configure each device that an employee may need to use at work. In all of these cases, it may be easier to have employees enroll their own devices into a wireless network as needed, rather than expecting them to be preconfigured by an administrator.

5 Related Work

The use of a gesture-based user interface to communicate a small piece of information for bootstrapping a larger data exchange is a relatively recent idea. In response to increasing realization that ubiquitous computing will demand that users select among many computers around them, systems such as gesturePen [37] have used infrared-based pointing mechanisms to allow users to select desired targets. The role of gesturePen is to help users establish data communications with computers around them, and the infrared channel is used to exchange IP address information. Our work relies on the same intuitive user experience as gesturePen, but builds on it by providing a secure information exchange.

Gesture-based user interfaces have also found other applications, some with security in mind [5, 30], some without [19]. To our knowledge we are the first to apply this idea to provide a complete solution for securing wireless 802.11 networks.

The idea of location-limited channels originated in [35] (although not under that name). In [5], this idea was expanded to use public key cryptography, enabling a much wider range of potential types of location-limited channels. The list of possible location-limited channels, and their uses, continues to expand [20, 21, 30].

Our system reduces network security to the physical security at the time of enrollment – a simple, intuitive model accessible to non-technically-savvy users. This is in contrast to Microsoft's CHOICE network [25] and the Secure Wireless Gateway (SWG) [14], which do not require physical proximity, but are much harder to use. For example, in Microsoft's CHOICE network, users enter an existing Passport password into a web page every time they want to use a public wireless network. The SWG asks a user to log in to a secure web site using an existing password and execute additional configuration steps. While entering a password may not seem like a burden, adding seemingly simple steps like this actually has large impact for non-technically-savvy users [10]. Furthermore, gesture-based automatic configuration can be used with a wide variety of embedded devices that may not allow users to enter passwords.

Both SWG and Microsoft CHOICE require the user to have an existing trust relationship with the network provider, while our approach allows mutual authentication between users and network providers that share no preexisting relationship. We also point out that our system conveys all of the security advantages of using digital certificates in a Public Key Infrastructure. This is in contrast to SWG, which secures wireless access using IPSec configured with a shared secret.

The perceived difficulties associated with managing Public Key Infrastructures often lead users to look for other, less secure alternatives. There has been some work, however, to make PKIs more usable (see, for example, [15]). The location-limited channels we use allow us to side-step much of the bootstrapping problems usually found when trying to build a global Public-Key Infrastructure. We also show with our work that one can quite effectively use a "small-scale" PKI without inheriting the usability problems usually associated with larger PKIs.

6 Conclusions

Security and usability are typically thought to be at odds with each other: highly secure systems are thought to be necessarily difficult to use, and systems that can be easily managed by end users are thought to be inherently insecure. Yet deploying systems of mobile devices, in which ease of administration and security are both pressing concerns, requires a resolution to this apparent conflict. We have demonstrated, by way of example, that security and usability are not always irreconcilable. Our "Network-in-a-Box" system provides an example of how gesture-based user interfaces can lead to systems that are both secure and easy to use.

We have implemented our NiaB system, both in a stand-alone and an enterprise version, to test out our design. Through user studies, we experimentally measured a significant decrease in the time required by users to set up a secure wireless network as compared to a typical commercial access point. This improvement – from approximately ten minutes down to under a minute – is especially favorable when one notes that our solution sets up the highly secure 802.1x/EAP-TLS standard, versus the significantly less secure password-based WEP standard used by the commercial access point.

Our approach, gesture-directed automatic configuration, relates digital security to physical security in a way that users find intuitively easy to understand. Although we have applied this technique to address the particularly pressing problem of securing 802.11 wireless networks, the approach is quite general and can be used to design a variety of systems that are both secure and easy to administer.

7 Acknowledgments

We like to thank the anonymous reviewers and Tim Diebert for their helpful comments. Alp Simsek built the "phone home" service described Section 3.2.2. Raghu Gopalan provided valuable feedback about our enterprise deployment.

References

[1] B. Aboba and D. Simon. *PPP EAP TLS Authentication Protocol (EAP-TLS)*. IETF - Network Working Group, The Internet Society, October 1999. RFC 2716.

[2] Wi-Fi Protected Access. WPA. `http://www.wifialliance.org/opensection/protected_access.asp`.

[3] C. Adams and S. Farrell. *Internet X.509 Public Key Infastructure Certificate Management Protocols*. IETF - Network Working Group, The Internet Society, March 1999. RFC 2510.

[4] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in tunnelled authentication protocols. In *11th Security Protocols Workshop*, Cambridge, United Kingdom, April 2003. Springer-Verlag.

[5] Dirk Balfanz, D.K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the 2002 Network and Distributed Systems Security Symposium (NDSS'02)*, San Diego, CA, February 2002. The Internet Society.

[6] L. Blunk and J. Vollbrecht. *PPP Extensible Authentication Protocol (EAP)*. IETF - Network Working Group, The Internet Society, March 1998. RFC 2284.

[7] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11, 2001.

[8] P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese. *IEEE 802.1x Remote Authentication Dial-In User Service (RADIUS) Usage Guidelines*. IETF - Network Working Group, The Internet Society, September 2003. RFC 3580.

[9] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. IETF - Network Working Group, The Internet Society, January 1999. RFC 2246.

[10] Joseph S. Dumas and Janice C. Redish. *A Practical Guide to Usability Testing*. Ablex Publishing Corporation, 1993.

[11] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Eight Annual Workshop on Selected Areas in Cryptography*, August 2001.

[12] Wireless Tools for Linux. `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`.

[13] The OpenBrick Foundation. OpenBrick. `http://www.openbrick.org/`.

[14] Austin Godber and Partha Dasgupta. Secure wireless gateway. In *Proceedings of the ACM Workshop on Wireless Security (WiSe-02)*, pages 41–46, New York, September 28 2002. ACM Press.

[15] Peter Gutmann. Plug-and-play PKI: A PKI your mother can use. In *Proceedings of the 12th USENIX Security Symposium*, pages 45–58, Washington, D.C., August 2003.

[16] The Cable Guy. Windows XP wireless auto configuration. `www.microsoft.com/technet/columns/cableguy/cg1102.asp`, November 2002.

[17] IEEE. ANSI/IEEE. 802.1x: Port-based network access control, 2001.

[18] IEEE. ANSI/IEEE. 802.11i: MAC enhancements for enhanced security, 2003.

[19] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, Celine Pering, John Schettino, Bill Serra, and Mirjana Spasojevic. Places and things: Web presence for the real world. In *3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, 2000.

[20] Tim Kindberg and Kan Zhang. Secure spontaneous device association. In *UbiComp 2003*, 2003.

[21] Tim Kindberg and Kan Zhang. Validating and securing spontaneous associations between wireless devices. In *Proceedings of the 6th Information Security Conference (ISC03)*, 2003.

[22] The Dumb Networking Library. libdnet. `http://libdnet.sourceforge.net/`.

[23] The Packet Capture Library. libpcap. `http://www.tcpdump.org/`.

[24] C. Lopes and P. Aguiar. Aerial acoustic communications. In *Proceedings of the 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2001.

[25] Microsoft. The CHOICE network. `http://www.mschoice.com/`.

[26] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *ACM Conference on Human Factors in Computing Systems (INTERCHI '93)*, pages 206–213, 1993.

[27] Open Source Implementation of IEEE 802.1x. xsupplicant. `http://www.open1x.org/`.

[28] The FreeRADIUS Server Project. FreeRADIUS. `http://www.freeradius.org/`.

[29] The HostAP Project. HostAP. `http://hostap.epitest.fi/`.

[30] Jun Rekimoto, Yuji Ayatsuka, Michimune Kohno, and Hauro Oba. Proximal interactions: A direct manipulation technique for wireless networking. In *Proceedings of INTERACT 2003*, 2003.

[31] C. Rigney, A. Rubens, W. Simpson, and S. Willens. *Remote Authentication Dial-In User Service (RADIUS)*. IETF - Network Working Group, The Internet Society, June 2000. RFC 2865.

[32] Robert Moskowitz. Weakness in passphrase choice in WPA interface, 2003.

[33] Acme Software. mini_httpd. `http://www.acme.com/software/mini_httpd/`.

[34] Network Driver Interface Specification. NDIS. `http://www.ndis.com/`.

[35] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th Security Protocols Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194, Cambridge, United Kingdom, 1999. Springer-Verlag, Berlin Germany.

[36] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. In *Proceedings of the 2002 Network and Distributed Systems Security Symposium (NDSS'02)*, San Diego, CA, February 2002. The Internet Society.

[37] Colin Swindells, Kori M. Inkpen, John C. Dill, and Melanie Tory. That one there! pointing to establish device identity. In *ACM Conference on User Interface Software and Technology (UIST 2002)*, pages 151–160, 2002.

[38] W. Townsley. *Layer Two Tunneling Protocol (L2TP)*. IETF - Network Working Group, The Internet Society, December 2002. RFC 3438.