A Context Authentication Proxy for IPSec using Spatial Reference

Rene Mayrhofer
Lancaster University, Computing Department
Infolab21, South Drive, Lancaster, LA1 4WA, UK
rene@comp.lancs.ac.uk

Abstract:
Spontaneous interaction in ad-hoc networks is often desirable not only between users or devices in direct contact, but also with devices that are accessible only via a wireless network. Secure communication with such devices is difficult because of the required authentication, which is often either password- or certificate-based. An intuitive alternative is context-based authentication, where device authenticity is verified by shared context, and often by direct physical evidence. Devices that are physically separated can not experience the same context and can thus not benefit directly from context authentication. We introduce a context authentication proxy *that is pre-authenticated with one of the devices and can authenticate with the other by shared context. This concept is applicable to a wide range of application scenarios, context sensing technologies, and trust models. We show its practicality in an implementation for setting up IPSec connections based on spatial reference. Our specific scenario is ad-hoc access of mobile devices to secure 802.11 WLANs using a PDA as authentication proxy.*

1 Introduction

Spontaneous interaction is a desirable feature for many ubiquitous computing scenarios. It is typically seen as a process between users or devices that are in direct contact with each other, and often implies spatial proximity. However, spontaneous interaction can also be important between users or devices that are physically or virtually separated, but can communicate over some common channel like a wireless network. A similar situation arises when interacting with devices that do not feature any user interface, but only communicate wirelessly. One prominent example is IEEE 802.11 WLAN itself: users, represented by their client devices, engage in spontaneous interaction with access points that usually neither have a user interface nor are physically accessible (they might be built into building infrastructure).

The problem with such settings is to authenticate users or devices. Wireless networks are particularly vulnerable to attacks, ranging from simple eavesdropping to more sophisticated man-in-the-middle (MITM) attacks. Although there are well-known protocols to secure communication over wireless networks, they all depend on some form of authentication. Only after authenticating the communication partner, further steps to create a secure channel make sense. More specifically, the problem is to authenticate intuitively and efficiently.

From a user point of view, secure channel setup should be as transparent as possible and should cause minimal, if any, overhead to the desired spontaneous interaction. An example of a secure channel implementation is IPSec. IPSec is considered as a secure communication protocol. It supports both password- and certificate-based authentication and has been designed for cross-platform interoperability, but is daunting to set up even for technically skilled users. Although it has desirable properties from a security point of view, many users may choose not to use it for spontaneous and ad-hoc interactions. Giving credit to its wide-spread use and practical problems, also investigated by others [1, 3], we therefore use the setup of IPSec connections as our motivating example. More specifically, our demonstration application is to grant secure access to a WLAN access point – and consequently the network it manages – to new clients such as laptops.

Context based authentication, or *context authentication*, allows secure and intuitive authentication without introducing unreasonable overhead that would be incompatible with spontaneous interaction. It uses shared context between devices to create shared secrets. These shared secrets can consequently be used as cryptographic tokens for creating secure channels. However, devices such as WLAN access points that are physically separated from user devices or that have no sensors or user interfaces are unable to experience the same context.

Our approach to allow such devices to authenticate via shared context is to introduce a *context authentication proxy*. The proxy is pre-authenticated to the device that does not have sensors or a user interface itself, and authenticates to other devices on behalf of it. This concept is independent of the underlying infrastructure for expressing trust, and can work in online and offline settings and with existing password- or certificate-based authentication mechanisms. Our example application uses a mobile context authentication proxy in the form of a personal digital assistant (PDA) for better ease of use.

The contribution of this work is twofold: we examine the general concept of a context authentication proxy in more detail, discussing different options of implementing it, and we show a specific application for a widely used protocol. This application confirms a user study presented in related work [1], anecdotally showing a significant improvement of ease of use in setting up IPSec connections due to use of context authentication. We also argue that, although demonstrated by an application for securing wireless networks, context authentication proxies are of wider applicability.

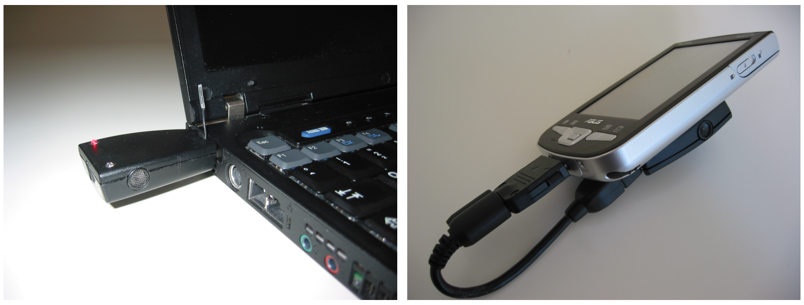
In the following, we first discuss related work in Section 2 and the previously introduced concept of context authentication in Section 3. Our main contribution is the general notion of an authentication proxy presented in Section 4 and our specific implementation for authenticating IPSec connections shown in Section 5. Finally, we briefly discuss further alternatives for implementing context authentication and for using the established shared secrets in secure communication protocols in Section 6.

2 Related work

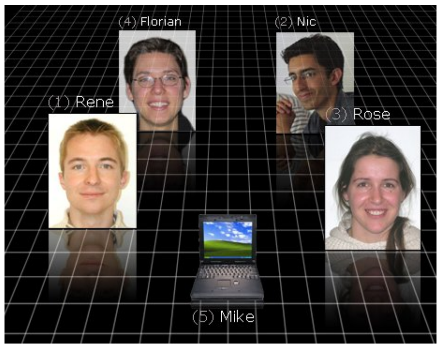
Our chosen example of securing IEEE 802.11 WLAN using context authentication has also been discussed by Balfanz et al. [1]. They present a system called "Network-in-a-box" (NiaB) that uses an infrared channel to transmit authentic cryptographic tokens and automates the set-up of secure wireless communication in much the same way as our example application does. This infrared connection is established between the client device and either the WLAN access point itself, or, in case of a distributed infrastructure, an "enrollment station", which can be regarded as a stationary instance of a context authentication proxy. Furthermore, they show in a user study that context authentication can, for end-users, significantly lower the time required to set up a secure wireless network. The major difference to our work is the role of the authentication proxy. In NiaB, the authentication proxy is described as a permanent station that authenticates all devices that are able to establish infrared connections to it. On the other hand, we specifically assign the authentication proxy an active role, in which it triggers the authentication process to a selected client, as described in more detail in Section 4. A specific advantage of our approach is that the authentication proxy can be mobile — and for our demonstration application, it explicitly is. Instead of forcing users to bring their devices to fixed stations, administrators can authenticate devices wherever it is necessary and appropriate. This can include authentication of new fixed stations, which is not possible with the less flexible enrollment station described by NiaB.

Kindberg et.al. [9] describe "channel proxies", which may be seen as a low-level implementation of a context authentication proxy. These channel proxies selectively forward messages depending on some constraints, like location of the sender or the receiver. In contrast, our concept of context authentication proxies explicitly includes high-level processing of messages. In our example, this allows the complete authentication protocol to be performed between the proxy and the WLAN client, while the WLAN access point will typically be unaware of the whole process.

Godber and Dasgupta [3] describe another implementation that is closely related to the demonstrative application we discuss in Section 5. Their system called "Secure Wireless Gateway" (SWG) uses IPSec to secure IEEE 802.11b WLAN, and also provides a captive portal to redirect unauthenticated users to a web page with instructions on how to authenticate. They suggest to use a common shared key for guest users, which is to be considered insecure against MITM attacks, and individual shared keys for registered users. However, they explicitly do not investigate generation and distribution of these individual shared keys or the use of certificate-based authentication and define it as out of scope of their work. In the present paper, we



(a) USB dongles for sensing relative spatial positions can be added to typical off-the-shelf laptops, PDAs, or even mobile phones



(b) User interface concept for spatial selection

Figure 1: Current implementation of context authentication by spatial reference

focus on this key distribution problem and present an implementation similar to SWG as an example application making use of easy key distribution.

3 Context authentication

Context authentication tries to provide intuitive means of authenticating users or devices by verifying that they are in some specific context, e.g. at some specific location. The possibilities for sharing context are obviously constrained by the sensors available to the involved devices. These sensors are used to verify some properties of the device to authenticate, i.e. to verify that the other device is in the same context. Context authentication then aims to create shared secrets for setting up secure communication, usually in the form of cryptographic key material.

In earlier work [12], we reported about using *spatial reference* for authenticating spontaneous interaction. Selecting devices based on direct line of sight has also been explored with the gesturePen [16]. The gesturePen has the intention of selecting devices by pointing at them, in much the same way as we select devices based on their relative spatial position in this work. Location is just one option for context authentication, and for the description of additional options, we refer to others [9, 2]. In the present paper, we investigate connections that are initiated in an ad-hoc manner but that might yield longer-lived security associations. Specifically, we establish IPSec connections on first contact, but continue to use these connections once established.

Building upon our current implementation [12], we assume devices to be equipped with sensors in the form of USB dongles. These dongles provide accurate sensing of relative spatial positions using ultrasound. Figure 1a shows two of them attached to a laptop and attached to a PDA — both can sense each others position with an accuracy of better than 10 centimeters in distance and 25° in angle [6].

In an office space with many laptops, PDAs, and other devices communicating over the same wireless network within a small area, this fine-grained sensing of shared context offers distinct advantages in selecting specific devices. If an administrator wants to allow "that device over there" to access the wireless network[1)], then other devices in the same room should not automatically be allowed too. Solutions based on infrared connections can not easily provide such a fine-grained selection because infrared beams often span the whole area.

Our context authentication protocol integrates secure authentication transparently and seamlessly with device selection, as shown in Figure 1b. Simply by selecting a visualized device position, corresponding to the physical device as visible to the user, the authentication process is triggered. User interaction is thus changed from selecting devices from a network-discovered list to a spatially-discovered environment; authenticating selected devices happens automatically without further user interaction. This seamless integration makes the protocol well suited for spontaneous interaction. Security properties of our authentication protocol have been discussed previously [12], and here we simply assume the protocol to provide a shared secret to both devices that perform the spatial authentication.

Although we build upon this specific authentication protocol for our demonstrative application, the concept of authentication proxies is independent of the underlying sensing platform for context authentication.

4 Authentication proxy

Previous work on context authentication assumes that those devices that authenticate each other can experience the same context, but this is not always possible. Figure 2 shows a device A, e.g. owned by Alice, trying to interact securely with a device B, e.g. a WLAN access point. Because the access point is physically inaccessible, Alice can not benefit from direct context authentication with it to secure her communication. By introducing a *context authentication proxy* P, we give her this option. To facilitate an authentication between two devices, the authentication proxy experiences the same context as one of the devices, i.e. it shares some aspect of the context. With the other device, it is pre-authenticated. It will usually be desirable that context be shared with the more volatile side, i.e. with mobile devices, changing environments, or, generally speaking, with transient connections. Since we assume a more permanent relationship with the other end of the authentication, in this example between P and the access point, the necessary pre-authentication only needs to occur once during set-up of these devices. Any standard authentication protocol, e.g. password- or certificate-based ones or any means of conveying trust of B in P can be used. Due to this trust relationship, the possibly mobile authentication proxy P is assumed to be used or maintained

[1)]The same method can be used to allow access to private parts of the network, or, more generally speaking, to specific resources. We use access to the wireless network only as an example.

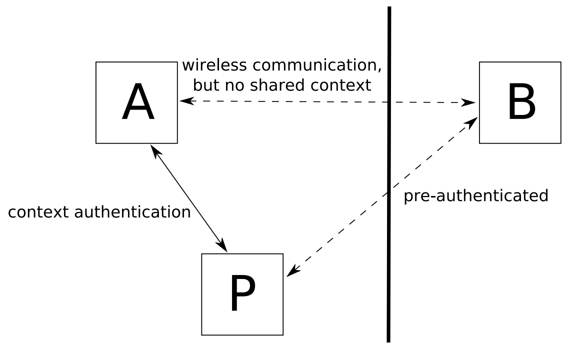


Figure 2: Using a context authentication proxy P allows physically separated devices A and B to benefit from context authentication even when they can not experience the same context

by a trusted person, such as a system administrator.

The main task of the authentication proxy is to create a shared secret between A and B, to enable secure communication between them over a wireless network. Depending on the initiator of the authentication, we can distinguish between two different approaches for user interaction with the proxy:

- We speak of a *passive authentication proxy* when P acts as an authentication service and simply waits for clients to initiate an interaction. The *client* takes the active role, starts context authentication with P to obtain a shared secret for communicating securely with B, and may need to engage in another authentication procedure with B over the now-authenticated wireless network. Instances of this approach are the closely related NiaB [1], and one of our previous works [13], which describes the use of RFID tags to secure communication over wireless ad-hoc peer-to-peer networks. The former requires a further offline authentication step performed by the in-house certificate authority when used for "enterprise" WLAN access, or relies on the infrared channel authentication for the simpler "home" WLAN setting. In the latter, we store public keys of network peers on associated RFID tags that can be read for secure spontaneous interaction. Note that in this previous work, we termed the RFID tags "objects" and the associated devices with which the interaction takes place the "proxies" because of a slightly different focus on the interaction.

- For an *active authentication proxy*, the roles of waiting for and of initiating the context authentication are swapped between A and P. That is, the *proxy* takes the active role, starts context authentication with A to generate a shared secret for letting A communicate securely with B, and may take additional steps to register A with authorization databases. In this case, A only waits to be authenticated and does not need to take any additional steps. This requires even less user interaction by offloading some steps to the proxy and thus can further decrease the burden placed on the user for setting up secure communication. We point out that the interaction between A and P, and subsequently between A and B, is still spontaneous. However, the change in roles relieves the client from going through additional steps after the initial context authentication and shifts this task to the proxy. P is in a better position to perform them, because it is part of the existing network and is thus assumed to know more about it than the new client.

Choosing between a passive and an active authentication proxy also depends on the respective trust model. If the trust model can express transitive trust, i.e. delegating trust from one entity to another, then B can delegate authorization decisions to P. Without the ability to delegate trust, an active authentication proxy can still initiate the context authentication, but a subsequent authorization step might be necessary before A can access resources on B. In this case, the choice of authentication proxy should match the interaction style of the application, i.e. who initiates the spontaneous interaction. Note that arbitrary trust models can be used, including the sharing of passwords, which is clearly not recommended from a security point of view, and that most can be used to delegate trust in some way. A concept for delegating restricted trust over potentially multiple hops is described e.g. by Steffen and Knorr [15] and could be used in combination with context authentication proxies.

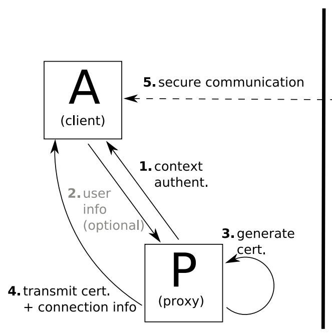
One secure and standardized option to delegate trust is to use X.509 certificates signed by a certificate authority (CA) managed by P and trusted by B. Every certificate that P creates and signs will be trusted by B, allowing P to make decisions about authorizing clients to use B's services. In this sense, our approach of a context authentication proxy is an implementation of the plug-and-play PKI [5]: a client device is automatically provided with an X.509 certificate that allows it to use some services. But instead of initially authenticating with the suggested username/password combination, we authenticate client devices based on context, specifically on their relative spatial position. This makes the approach more usable for spontaneous interaction.

5 Application for establishing IPSec connections

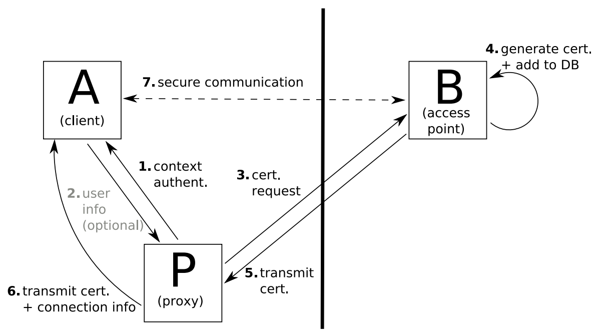
In this section we show *IPSecME* (IPSec made easy), an application to delegate trust for authorizing IPSec connections that uses an active authentication proxy and standard X.509 certificates. It uses our secure spatial authentication protocol described in earlier work [12] and does not depend on software being pre-installed on the client like NiaB [1].

5.1 Concept

Our application can be used for setting up arbitrary IPSec connections by providing appropriate connection details in the form of an XML configuration file to the authentication proxy. IPSec tunnels over an otherwise open 802.11 WLAN are a practical example without loss of generality. For simplifying the discussion, we also assume the access point to act as an IPSec gateway, but it could be easily split over different devices without any change to our work.



(a) The proxy implements the CA — no connection between the proxy and the access point is necessary



(b) The access point (or infrastructure) implements the CA — the proxy needs an online connection to request certificates for and forward them to the client

Figure 3: Two options for delegating trust with a context authentication proxy

The application consists of two parts, one running on the client and one on the proxy device. Figure 3 shows two options for implementing this application using an active context authentication proxy P: the CA can either run directly on P, or it can run on the access point B (or any other infrastructure device). In the former case, B delegates trust about authorization to P by allowing all clients A that present a certificate signed by the proxy's CA to establish IPSec tunnels. As illustrated in Fig. 3a:

1. P authenticates A via shared context, in this application via spatial reference.
2. A can optionally send information about the logged in user, the machine name, etc., if this should be encoded in the X.509 certificate.
3. P generates a new X.509 certificate with the information provided by A and/or locally entered data and signs it with its CA key. Note that the certificate includes the matching private key.
4. P forwards the new certificate, the private key, its CA certificate, and details about the IPSec connection, i.e. the IP address of the gateway, the remote subnet, etc. to A. The private key is encrypted with the shared key generated in step 1.
5. A uses its new certificate and the IPSec connection description to establish a secure connection to B.

This option has the advantage that no online connection between the P and B is required. The trust between them is formed by B importing P's CA certificate. After this, no further communication between B and P is necessary for authenticating arbitrary clients[2)].

In the latter case, P requests certificates from the CA running on B using an online connection. As illustrated in Fig. 3b:

[2)]Note that revoking a certificate that P generated will require an update of its associated certificate revocation list (CRL) on B, and consequently communication between B and P. However, for spontaneous interactions, short-lived certificates can be used to alleviate the need for CRL updates.

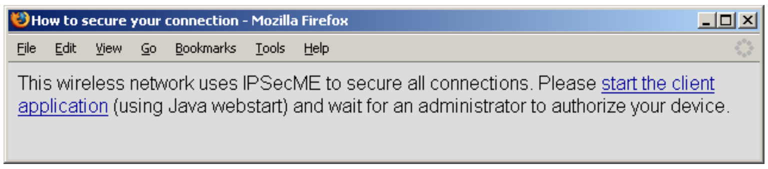
1. equal to step 1 in the former case
2. equal to step 2 in the former case
3. P generates a certificate request with the information provided by A and/or locally entered data and sends it to B.
4. B decides if A should be authorized and, if yes, signs the certificate request with its CA key and adds the new certificate to its authorization database.
5. B sends the new certificate to P.
6. equal to step 4 in the former case
7. equal to step 5 in the former case

The necessarily secure connection between B and P forms the pre-authentication between them with a slightly different trust model. B trusts P to authenticate A based on shared context and to forward machine information and certificates, but keeps decisions about authorization local. For spontaneous interaction, the first option has the advantage that no connection between B and P is necessary, thus we implement this one.

5.2 Implementation

The implementation currently runs on a standard Laptop running Windows XP SP2 as the client A and a PDA running Pocket PC 2003 as the authentication proxy P. Because our context authentication protocol using spatial reference and this new application have been implemented in Java, other platforms can be used as well. All platform-specific parts, i.e. managing certificates, establishing IPSec connections, and access to the ultrasound sensing devices, have been implemented for Windows XP, Linux, and Mac OS/X. The combination of access point and IPSec gateway, depicted as B in the above concept, has been implemented in two different versions. A standard access point connected to a PC running Gibraltar firewall [11] represents an enterprise scenario where the functionality of B is distributed in the infrastructure. An embedded implementation using the OpenWrt distribution [14] on an Asus WL-500GP access point represents the home/small office scenario with a single, combined device. Both implementations use Openswan [17] as IPSec implementation and ChilliSpot [7] to provide the captive portal. These two scenarios show that our approach can be used with arbitrary implementations of WLANs and IPSec gateways as long as they support external X.509 CAs.

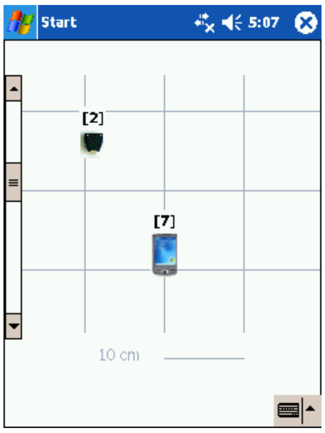
Figure 4 shows how users experience the whole process. The client does not need to have any special software pre-installed and does not need any a priori information about the environment. When it first connects to the WLAN, which is publicly accessible, its web browser gets redirected to a local web page in the same way as it is used by the currently popular WLAN hot spots (see Fig. 4a). From this web page, the user can start the client part of the application via Java Webstart and then simply waits for the proxy to initiate authentication. We assume that devices are either equipped with ultrasound sensing or that the USB dongles


How to secure your connection - Mozilla Firefox

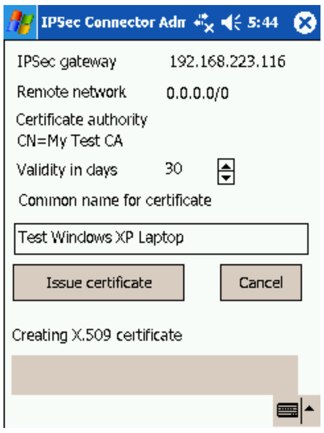
File Edit View Go Bookmarks Tools Help

This wireless network uses IPSecME to secure all connections. Please start the client application (using Java webstart) and wait for an administrator to authorize your device.

(a) When first trying to access the network, the client is redirected to a page that delivers the client application



(b) Authentication proxy: selecting the client to authenticate based on spatial reference

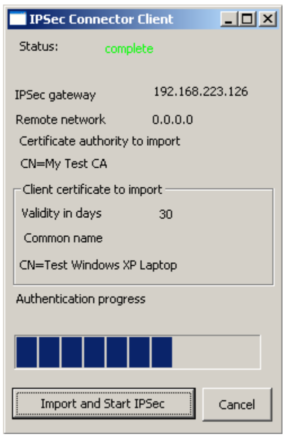

IPSec Connector Adr 5:44

IPSec gateway 192.168.223.116
Remote network 0.0.0.0/0
Certificate authority
CN=My Test CA
Validity in days 30
Common name for certificate
Test Windows XP Laptop

Issue certificate | Cancel

Creating X.509 certificate

(c) Authentication proxy: after setting a name and the validity period for the new certificate, the client is authenticated


IPSec Connector Client

Status: complete

IPSec gateway 192.168.223.126
Remote network 0.0.0.0
Certificate authority to import
CN=My Test CA
Client certificate to import
Validity in days 30
Common name
CN=Test Windows XP Laptop
Authentication progress

Import and Start IPSec | Cancel

(d) Client: accepting the new certificate and the CA of the IPSec gateway to establish the connection

Figure 4: Screenshots of the IPSecME application

are attached at this stage. For ease of use, we skip the optional step 2 and omit to use client-provided information for generating the certificate. With spontaneous interaction, any such information tends to be meaningless anyway due to the lack of a globally accepted naming scheme. An administrator using the context authentication proxy can then select the client based on spatial reference (see Fig. 4b) and specify the validity period of the certificate and optionally a name describing the client for later use (see Fig. 4c). This name only needs to be meaningful within this environment, e.g. to the administrator. After initiating the context authentication protocol, the certificate is generated and signed automatically, and the IPSec connection details along with the certificate are sent to the client. Note that the private key contained in the PKCS#12 format used for transmitting the certificate is encrypted with the shared secret that has been established between the client and the proxy during context authentication. Thus, they can communicate over the public, insecure WLAN without worrying about attacks. Finally, after receiving the certificate and connection details, the client can, when accepting them, immediately import its new certificate and establish the IPSec connection (see Fig. 4d). Further communication is automatically secured by the IPSec tunnel, which in our case includes all traffic to and from the client.

6 Discussion

The concept of an authentication proxy is generally applicable to arbitrary ways of authentication via shared context, and NiaB has already shown that the use of a special instance of an authentication proxy with infrared works well. It has yet to be investigated how well this concept integrates with other options such as cameras or microphones for sensing shared context. Our software has been designed to make the context authentication protocol exchangeable. It is a simple task to change our application to use IrDA like in NiaB, for example, or to use something different like authentication over an audio channel [4] or with mobile phone cameras [8]. Even though practical applications have not yet made use of authentication proxies in those cases, we do not anticipate any major obstacles.

Also, there are other options for implementing the secure channel after successful authentication. In this work, we use the well-known IPSec protocol, but the different TLS suites, IEEE 802.1x, or IEEE 802.11i are also considered to be secure protocols and may be more appropriate for different application scenarios. Securing WLANs has been chosen as a scenario due to its practicality and wide applicability. By leaving the WLAN itself open and publicly accessible, we can provide public services usable without authentication, and additional access to authenticated users. We already use this possibility to deliver the authentication application to new clients, thus making it unnecessary to require any pre-installed software. This combination of two (or multiple) levels of service is more difficult to achieve with IEEE 802.1x. For purely spontaneous interaction, IPSec transport connections can be used between just two hosts instead of tunnel connections for securing all traffic a host generates.

For trust delegation, there are again multiple possibilities. In our application, we rely on standard PKI techniques, but shared passwords, OpenPGP keys, or even hardware tokens are other examples that can be used with the same concept. The decision of using online or offline relationships between the service and the authentication proxy is also highly dependent on both the application and the trust model. If the trust model allows delegation of trust, then an active authentication proxy can have distinct advantages, especially when a wireless connection to the actual service is not available ubiquitously. The trust relationship then allows pre-authentication of a client to the service, via the authentication proxy, even before any wireless contact to the actual service is possible. This gives more freedom in performing the authentication, because it can be done at any time for later use. Our application demonstrates this by pre-authenticating IPSec connections for accessing a private network securely over an otherwise public WLAN or from the Internet. This use of IPSec connections is often termed “road warrior” support, because the home network can be accessed from anywhere.

The security of our approach builds upon three parts: First, our context authentication protocol is considered secure against known attack scenarios; it uses multiple rounds of an interlock protocol to verify that only a device at a specific relative position can successfully authenticate. Ultrasound sensing is used as a side channel for transmitting information, in a way that is tightly interwoven with the spatial relationship between devices and that prevents man-in-the-middle attacks on the wireless channel (see [12] for a more detailed analysis). Second, IPSec as a protocol for secure channels is currently considered as one of the most secure standards. Third, well-known PKI techniques delegate trust to the context authentication proxy. We explicitly point out that the security of our proposed use of authentication proxies relies on the physical security of the proxy devices; when attackers can access these proxies physically, they can access resources as defined by the respective trust model. This is not a new restriction — the security of most protocols relies on physical security of some of its components. An active authentication proxy, like the PDA in our example application, might be small and mobile, and thus even more care needs to be taken to protect it.

Currently, no formal user study has been done to verify if our application eases the task of securing a WLAN or setting up other IPSec connections. We only have anecdotal evidence that IPSec connection set-up is significantly eased by our use of an active authentication proxy: For comparable security using e.g. the web administration interface of Gibraltar firewall, an administrator first needs to log in and navigate to the certificate management module (4 steps), create a new certificate for the client (10 fields in a web form), and download it. Then this certificate needs to be imported on the client machine (manual transfer of the file, e.g. with a USB storage device, followed by 14 steps under Windows XP) and an IPSec connection needs to be created (8 steps with the Windows XP wizard). In contrast, using our demonstration application, a new client needs to start the application (1 step, Fig. 4a), an administrator needs to spatially select the client device (1 step, Fig. 4b) and enter the certificate details (2 fields, Fig. 4c). After automatically transmitting the new certificate to the client and importing it, the user only needs to start the IPSec connection (1 step, Fig. 4d). Intuitively it seems clear that it is a considerable improvement over manual configuration. By explicitly assigning the authentication proxy an active role, the end user is relieved from dealing with the connection set-up details at all. This combines into a single step two tasks that are usually separate: the selection, often called *identification*, of a device followed by a proper authentication, and the *authorization* to use some service. We argue that only one step, namely deciding about authorization, is necessary from an administrator point of view and that the authentication step should be made implicit for spontaneous interaction to become viable.

It might become difficult to distinguish devices on the visualized map when too many are presented at once. However, this is an implementation issue and is not inherent to the concept of an authentication proxy. We point out that the use of spatial reference for context authentication assumes the availability of appropriate sensors, either built into a device, or attached to it. For example in a meeting scenario, spatial reference is a generally useful tool [10] and using it for granting temporary access to resources – with the approach described in this paper – thus integrates seamlessly. In other scenarios, ultrasound sensing might not be readily available for current mobile devices. Although our USB dongles make it easy to attach them, it is an additional step that needs to be done. But, as mobile devices begin to include more sensors, context authentication will be more easily possible in the near future.

7 Conclusions

In this work, we argue that context authentication is more intuitive then the typical password- or certificate-based methods, especially for spontaneous interaction. The example of setting up secure WLAN connections shows clearly that these often-used methods do not scale with regards to the number of wireless connections used by a single person. A direct comparison between the number of steps that need to be executed by a user and an administrator for creating such a secure connection between a password-, a certificate-, and a context-based authentication procedure is obviously biased; our demonstration application has been designed specifically to make this as easy as possible, while other methods are usually not aimed at supporting spontaneous interaction. Nonetheless, practical experience shows that those WLANs where simple, spontaneous interaction is desired, such as WLAN hot spots in hotels or airports, either do not use any authentication at all or tend to be seen as awkward by most users. Context authentication allows to provide secure wireless connections without demanding user attention "just for security". Our main contribution is the general concept of a context authentication proxy, which allows devices to use context authentication when they can not actually experience the same sensor values for any suitable aspect of context. A first demonstration application implements this concept for a prominent example, namely WLAN access. The fact that other projects have approached this scenario shows the practical importance of the problem.

Compared to SWG, we benefit from the use of certificates to provide more security for larger scenarios, where re-keying of the whole system to disable access for a single client is not reasonable. We extend the results of the NiaB project in three areas: First, by making the context authentication proxy active, we give both the clients and the administrator more flexibility in the authentication process. By running a CA on the proxy, the decisions about authentication and authorization can be condensed into only one spatial device selection step to improve ease of use. Second, the proxy is made mobile and supports offline authentication where connectivity to the target network is not available. Third, ultrasound sensing provides more fine-grained selection of devices, and the same granularity is used in the spatial authentication protocol. This allows multiple devices in the same area to be distinguished better, e.g. to grant temporary network access in a meeting scenario with multiple laptops and PDAs on one desk. With an infrared channel like the one used in NiaB, there is no protection against active man-in-the-middle attacks. Therefore, the context authentication needs to be run in a secure environment where such attacks are prevented by organizational restrictions (e.g. that only one device is allowed to enter the authentication room at any time). With our proposed spatial authentication protocol, context authentication is secure even in public and untrusted environments.

Complete source code of our client and proxy implementations is available at `http://www.openuat.org/spatial-ipsec-proxy`, including configuration files for the gateway using Gibraltar firewall and using OpenWrt.

Acknowledgements

This work is partly supported by the EC under contract 013790 "RELATE", and by EPSRC under grant GR/S77097/01.

References

[1] D. Balfanz, G. Durfee, R. E. Grinter, D. K. Smetters, and P Stewart. Network-in-a-box: How to set up a secure wireless network in under a minute. In *Proc. 13th USENIX Security Symposium*, pages 207–222. USENIX, August 2004.

[2] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proc. NDSS'02: 2002 Network and Distributed Systems Security Symposium*. The Internet Society, February 2002.

[3] A. Godber and P. Dasgupta. Secure wireless gateway. In *Proc. WiSE'02: 3rd ACM workshop on Wireless security*, pages 41–46, New York, NY, USA, 2002. ACM Press.

[4] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human verifiable authentication based on audio. In *Proc. ICDCS 2006: 26th IEEE Int. Conf. on Distributed Computing Systems*, page 10. IEEE, July 2006.

[5] P. Gutmann. Plug-and-play PKI: A PKI your mother can use. In *Proc. 12th USENIX Security Symposium*, pages 45–58, August 2003. published at `http://www.cs.auckland.ac.nz/~pgut001/pubs/usenix03.pdf`, shorter version appeared in IEEE Computer Magazine, August 2002.

[6] M. Hazas, C. Kray, H. Gellersen, H. Agbota, G. Kortuem, and A. Krohn. A relative positioning system for co-located mobile devices. In *Proc. MobiSys 2005: 3rd Int. Conf. on Mobile Systems, Applications, and Services*, pages 177–190, New York, NY, USA, June 2005. ACM Press.

[7] Jens Jakobsen. Chillispot web page. `http://www.chillispot.org`, 2006.

[8] Michael K. Reiter Jonathan M. McCune, Adrian Perrig. Seeing-is-believing: Using camera phones for human verifiable authentication. Technical report, CMU, November 2004.

[9] T. Kindberg, K. Zhang, and N. Shankar. Context authentication using constrained channels. In *Proc. WMCSA: 4th IEEE Workshop on Mobile Computing Systems and Applications*, pages 14–21. IEEE Computer Society, June 2002.

[10] G. Kortuem, C. Kray, and H. Gellersen. Sensing and visualizing spatial relations of mobile devices. In *Proc. UIST 2005: 18th ACM Symposium on User Interface Software and Technology*, pages 93–102. ACM Press, October 2005.

[11] R. Mayrhofer and Esys GmbH. Gibraltar firewall web page. `http://www.gibraltar.at`, 2006.

[12] R. Mayrhofer, H. Gellersen, and M. Hazas. An authentication protocol using ultrasonic ranging. Technical Report COMP-002-2006, Lancaster University, October 2006.

[13] R. Mayrhofer, F. Ortner, A. Ferscha, and M. Hechinger. Securing passive objects in mobile ad-hoc peer-to-peer networks. In R. Focardi and G. Zavattaro, editors, *Electronic Notes in Theoretical Computer Science*, volume 85.3. Elsevier Science, June 2003.

[14] OpenWrt. OpenWrt web page. `http://openwrt.org`, 2006.

[15] R. Steffen and R. Knorr. A trust based delegation system for managing access control. In *Advances in Pervasive Computing: Adjunct Proc. of the 3rd Int. Conf. on Pervasive Computing*, volume 191, pages 1–5. Austrian Computer Society (OCG), April 2005. available at `http://www.pervasive.ifi.lmu.de/adjunct-proceedings/poster/p001-005.pdf`.

[16] C. Swindells, K. M. Inkpen, J. C. Dill, and M. Tory. That one there! pointing to establish device identity. In *Proc. UIST '02: 15th ACM symposium on User interface software and technology*, pages 151–160, New York, NY, USA, 2002. ACM Press.

[17] Xelerance Corporation. Openswan web page. `http://www.openswan.org`, 2006.