CHAPTER SIXTEEN

Making the Impossible Easy: Usable PKI

DIRK BALFANZ, GLENN DURFEE, AND D.K. SMETTERS

THE WIDESPREAD PERCEPTION THAT USABILITY AND SECURITY ARE AT ODDS WITH ONE ANOTHER OFTEN leads systems designers to shun powerful security technologies. A quintessential example is provided by public key infrastructure (PKI) technology: despite the high degree of security PKI technology can provide, designers frequently avoid this technology because of its notoriously complex deployment and the incomprehensibility of such an infrastructure to end users.

This chapter explains how by designing usability in from the start, one can make PKI-based systems easy to deploy and use. The resulting systems, however, are not large, general-purpose infrastructures, but PKIs that are small, dedicated, easy to set up, and application specific. We refer to these as *instant PKIs* (iPKIs). Several case studies illustrate interaction paradigms for building such usable, secure iPKIs.

Public Key Infrastructures

Before the invention of public key cryptography, methods for secure digital communications were all but unavailable to mainstream computer users. The reason for this was the difficult problem of *key distribution*: for Alice to send a secure message to Bob, they first would have to agree on a shared secret (e.g., a key or password). The only way to do this would be for Alice (or Bob, or a third party) to generate such a key and then

deliver that key to Bob (or Alice, or both). While in transit, the key not only had to be kept secret, but it also had to be protected from tampering.

Once Alice and Bob shared a secret key, Alice could do two things: first, she could *encrypt* messages with the shared key and be sure that only Bob would be able to read those messages. Second, if Bob used a *message authentication code*, Alice could be convinced that those messages indeed came from Bob and nobody else. Because Alice and Bob both used the same key, and because this key was used both for encryption and decryption, this kind of cryptography is usually called *symmetric key cryptography*.

Public key cryptography makes the key distribution problem much easier. Bob generates a *key pair* consisting of a *private key* and a *public key*. Now, all Alice needs to do is obtain a copy of Bob's public key. As the name suggests, a public key does not need to be kept secret. For example, people could publish their public keys in a central database. Alice would then simply connect to that database and request Bob's public key to be able to communicate with Bob securely. Again, this allows two different things: first, by encrypting messages with Bob's public key, which can only be decrypted using Bob's private key, Alice can ensure that only Bob can read them. Second, by using Bob's public key to verify a *digital signature* on a message, Alice can be convinced that the message indeed came from Bob—such a signature can only be created using Bob's private key.

How does Alice know that the public key she requests from the central database is indeed Bob's (and hasn't been tampered with while in transit to Alice)? *Certification authorities* (CAs) solve this problem, and also get around scalability issues with the central database. A certification authority digitally signs a statement consisting of Bob's public key together with an assertion that this key belongs to someone named "Bob." This signed statement is called a *certificate*. We assume that everybody knows the certification authority's public key. (In practice, these public keys today are part of operating system distributions such as Microsoft's Windows or Apple's Mac OS.) Using the certification authority's public key, Alice can now verify Bob's certificate, no matter where she obtained it. If the name on the certificate is indeed Bob's, and the certification authority's signature indeed verifies, Alice can be sure that she is now in possession of an authentic copy of Bob's public key.

Certification authorities can also issue certificates for *intermediate certification authorities*, which in turn issue digital certificates either to further intermediate CAs, or to *end entities* such as Bob. A hierarchical arrangement of certification authorities and end entities to allow certification of public keys is called a *public key infrastructure* (PKI).

Public key infrastructures have a number of advantages:

- Participants in the PKI can generate their keys independently of the parties with whom they want to communicate: new keys do not need to be generated and shared for every private communication desired.
- Public keys do not need to be kept secret. This vastly simplifies the key distribution problem.

- To ensure authenticity of public keys as they are passed around among members of the PKI, a certification authority issues public key certificates that include both a member's public key and some identifying attributes such as the member's name. Every member is assumed to trust the certification authority and know the CA's public key.
- Once the certificates are issued, any two members of a PKI can establish a secure connection without anybody else's help. They use *public key protocols* to ensure the integrity and secrecy of messages.

These are just the most important advantages of public key infrastructures, as compared to shared key systems. There are many more details and subtleties to PKIs that cannot be covered in this short introduction. What is important to remember is that these are not simply theoretical ideas. The concepts of public key cryptography and public key infrastructures have been implemented, and many implementations follow established standards. For example, we can use *X.509*[1] certificates to exchange public keys, and can use public key protocols such as *Secure Socket Layer*[2] (SSL) or *Transport Layer Security*[3] (TLS) to secure communication between web browsers and web servers.

Problems with Public Key Infrastructures

PKI technology was originally designed to be deployed globally. The idea was to have a single certificate infrastructure in which all users and devices would participate. Despite all the advantages of such a system, this has proven impractical: the amount of coordination and trust required to establish a global infrastructure is enormous, and no one has been able to do it in the 25 years since a global PKI was originally proposed.

Apart from such political problems, usability—or, rather, a lack of usability—is the main reason for the failure of public key infrastructures. Usability issues hamper PKI deployment even at much smaller scales, such as within a single company or within a supply chain. Here is a list of usability problems that people are confronted with when trying to use a PKI:

- Users don't have an intuitive understanding of public key cryptography. A study conducted by Whitten *et al*. shows that users have trouble understanding the difference between public and private keys, as well as the role that certificates play.[4] There is no intuitive model that explains the security properties of a PKI. (In contrast, in the physi-

1 R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," IETF—Network Working Group, The Internet Society, RFC 2459, Jan. 1999.

2 Alan O. Freier, Philip Karlton, and Paul C. Kocher, "The SSL Protocol Version 3.0," IETF—Transport Layer Security Working Group, The Internet Society, Nov. 1996.

3 T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF—Network Working Group, The Internet Society, RFC 2246, Jan. 1999.

4 Alma Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Proceedings of the 8th Usenix Security Symposium* (1999), 169–184. See also Chapter 34, this volume.

cal world, we understand that, say, those who have the key to our house can open the front door.)

- Users don't understand the connection between the PKI and the application goal they are trying to achieve. For example, the application goal might be to send a confidential email message. It is not clear to the average user what this has to do with certification authorities, key pairs, etc. Lamentably, the standard practice of reusing certificates across applications ensures that users will be exposed to gory PKI details: they must configure their applications to use certificates, understand when to request and how to install certificates, and so on.
- Not only do users not understand why they need certificates in the first place, but tasks such as requesting and installing certificates are often too cumbersome[5,6] (see the sidebar, "Case Study: Traditional PKI Deployment for an Enterprise Wireless Network").
- As mentioned earlier, certificates often attest that a certain public key belongs to a certain name (usually referred to as *identity certificates*), as names are considered easier for users to work with. However, in large PKIs , this changes the problem of finding the correct certificate for someone into a naming problem—e.g., knowing the "John Smith" in marketing from the one in accounting. It also means that for a CA to issue a certificate to the "right" person, it must have the ability to tell that the recipient is indeed the rightful owner of a particular name, not just a particular key pair.

As a result, organizations simply don't even bother issuing certificates to end users, limiting the use of certificates to servers that have been painstakingly configured by trained administrators.

Making PKI Usable

Can PKI technology be made usable? The following examples argue strongly that with new approaches to design, it is possible to build "instant" public key infrastructures that are both easy to use and, through the use of public key cryptography, highly secure.

Case Study: "Network-in-a-Box"

Wireless local area networks are becoming increasingly popular in homes and small offices. Unfortunately, such networks have been subject to a number of high-profile security problems, leaving users' machines vulnerable to attackers—not only those on their block, but also those accessing their network from quite a distance using high-gain antennas. Such networks can be secured, using new standards. However, the more secure the network, the more difficult it usually is to set up and configure (see the earlier sidebar for an example).

5 P. Doyle and S. Hanna, "Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage," 2003; *www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf.*

6 P. Gutmann, "Plug-and-Play PKI: A PKI Your Mother Can Use," *Proc. 12th Usenix Security Symposium*, 2003, 45–58.

CASE STUDY: TRADITIONAL PKI DEPLOYMENT FOR AN ENTERPRISE WIRELESS NETWORK

When we set up a wireless local area network (WLAN) at the Palo Alto Research Center (PARC), we opted for a PKI-based solution. This promised better security and better ease of use than a password-based system. The idea was to give each of about 200 users an X.509 certificate and to use the *802.1x* wireless standard's *Extensible Authentication Protocol* in TLS mode (802.1x EAP-TLS[1,2]) to authenticate to the wireless network.[3] Users requested and retrieved their certificates via a fairly automated, standard web-based interface, and configured their devices via the GUI-based 802.1x configuration software supplied with Microsoft Windows XP. To help users enroll, our administrators produced an elaborate set of instructions.

We studied eight subjects' experiences enrolling in the wireless PKI. Our subjects were sophisticated computer users, typically holding Ph.D.s in Computer Science. Despite using the GUI-based interface for enrollment and configuration of their machines, the process involved a total of 38 distinct steps. Each of these presented an opportunity for end users to make frustrating mistakes. The average time that it took them to request and retrieve their certificate and then configure their system was 140 minutes.[4]

Almost all of the subjects printed the instructions, and even those determined to understand what they were doing soon began following the instructions mechanically. In the end, many test subjects described enrollment as the most difficult computer task that PARC had ever asked them to do. All subjects had little idea of precisely what they had done to their computers. Several commented that if something were to go wrong, they could not perform even basic troubleshooting. For several subjects, this was the first time that they had ever experienced the inability to administer their own machines. Ironically, while PKI technology may have secured their machines for wireless use, it simultaneously reduced these end users' ability to configure and maintain their own machines.

1 B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol (EAP-TLS)," IETF RFC 2716 (Oct. 1999); *http://www.ietf.org/rfc/rfc2716.txt*.
2 ANSI/IEEE Std. 802.1x, Port-Based Network Access Control, IEEE (2001).
3 D. Balfanz, G. Durfee, R. Grinter, and D. Smetters, "In Search of Usable Security—Five Lessons from the Field," *IEEE Security & Privacy* 2, no. 5 (Sept./Oct. 2004).
4 Dirk Balfanz, Glenn Durfee, R.E. Grinter, D.K. Smetters, and Paul Stewart, "Network-in-a-Box: How to Set up a Secure Wireless Network in under a Minute," *Proceedings of the 13th Usenix Security Symposium* (2004), 207–221.

A recent study investigates how to build a wireless network that is extremely easy to set up and that, at the same time, provides the strongest grade of wireless security currently available. The authors of the study call their system a "Network-in-a-Box" (NiaB).[7]

7 *Ibid*.

This high level of security is provided by the 802.1x protocol,[8] which authenticates each wireless client before it can access the network, and then provides them with encryption keys that change rapidly and automatically to protect their data. Networks that use 802.1x ask clients to authenticate themselves using a choice of mechanisms. The most secure of these is a protocol called EAP-TLS,[9] which asks clients and the wireless infrastructure to mutually authenticate each other using digital certificates via the standard TLS[10] key exchange protocol. Deploying such a network means setting up a certification authority and an authentication infrastructure, and issuing certificates to each new client. As the example in the earlier sidebar shows, this is extremely difficult in an enterprise staffed by professional administrators. It is normally inconceivable for an average home user.

What was needed was a small-scale PKI (an instant PKI), one that covered a single wireless network and its clients, and one that could be configured and set up automatically. Users also needed a secure way to indicate which clients should be able to join that network, and a way to automatically have those clients obtain a certificate and be set up to use it.

The NiaB access point (NiaB AP), shown in Figure 16-1, is the solution to these problems. It provides a complete 802.1x-secured wireless network.[11] Clients who want to enroll in that network run a small enrollment application. In addition to being an access point, the NiaB AP has the authentication services necessary to support 802.1x, a certification authority to issue certificates to wireless clients, and an enrollment component (described later) to add new clients to its network securely. When the NiaB AP is switched on for the first time, it configures itself automatically. The access point component chooses a wireless network name and channel. The CA component generates a root key pair and root certificate. The NiaB AP is configured to automatically use an upstream Internet connection if one is available, and to act as a gateway for its clients—it provides all of the services necessary for its wireless clients to safely access the Internet such as a DCHP server, a firewall, etc. Even 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.

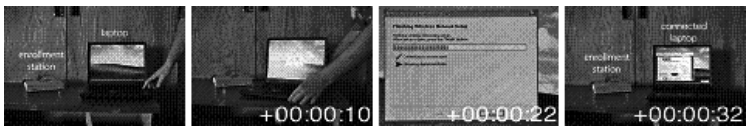

FIGURE 16-1. *Network-in-a-Box makes it easy to enroll in a secure wireless network*

8 ANSI/IEEE Std. 802.1x.

9 B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol (EAP-TLS)," IETF RFC 2716 (Oct. 1999); *http://www.ietf.org/rfc/rfc2716.txt*.

10 T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF RFC 2246 (Jan. 1999); *http://www.ietf.org/rfc/rfc2246.txt*.

11 Balfanz, Network-in-a-Box.

The process to easily and securely add new clients to this network is called *gesture-directed automatic configuration*: a user wishing to add his laptop to the NiaB network runs a small enrollment application, which tells him to "point out" the NiaB AP whose network he wishes to join. In the NiaB implementation, he points out the AP using the infrared port on his laptop (although other mechanisms could be used[12]). 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."

This gesture-based approach to configuration is simple and intuitive for the user and, at the same time, highly secure. When the user's laptop and the NiaB AP communicate over infrared (IR), they are actually exchanging digital fingerprint information with each other that will allow them to securely recognize each other's public keys, and the AP tells the client the name of the wireless network it should join. The infrared channel is an example of a *location-limited channel*.[13] Location-limited channels are channels that can be used to bootstrap secure communications between devices.

The client enrollment software on the laptop then makes a secure connection over the wireless network to the NiaB AP, authenticating each other by proving they have the private keys corresponding to the public key fingerprints they exchanged over IR. This takes the user's clear intention to have his laptop talk to that *particular* NiaB AP, and turns it directly into a secure connection over which the laptop can be configured. Over that secure connection, the enrollment software on the laptop requests a certificate from the NiaB AP's CA, and then retrieves the new certificate and installs it on the laptop. The enrollment software then configures the laptop to use the new certificate to access the NiaB network securely in the future. From then on, standard software in the laptop will automatically access and authenticate to the NiaB network whenever it is in range, without further intervention from the user.

By setting up and joining a NiaB network, the user is managing an "instant" public key infrastructure—a small, dedicated PKI—without even realizing it. Instead of burdening the user with complicated certificate management semantics, this iPKI provides 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 enough physical proximity of the access point to exchange information over infrared. For example, if an NiaB AP were to be deployed in a home, 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

12 Dirk Balfanz, D.K. Smetters, Paul Stewart, and H. Chi Wong, "Talking to Strangers: Authentication in ad hoc Wireless Networks," *Proceedings of the 2002 Network and Distributed Systems Security Symposium* (NDSS'02), Internet Society (2002), 23–35.

13 *Ibid*.

their NiaB AP in a closet.) This is a simple, intuitive trust model that is quite effective for many situations.

The gesture-based interface for enrolling in a NiaB network is designed to be as simple and intuitive as possible. Most importantly, it asks the user to perform only an action that he would have to perform anyway—to indicate *which* network he would like to join. User studies confirm that this approach is very easy to use—it took a population of 12 users on average only 2 steps and 51 seconds to set up a laptop to use a secure NiaB network, and they rated the system's ease of use very highly.[14] The interface also follows Yee's guideline:

> "Grant authority to others in accordance with user actions indicating consent."[15]

The user action indicating consent is the physical pointing out of the laptop to the access point. The authority granted is the network access given to the laptop. Without this explicit user action, no authority is granted to the laptop.

The authors of the study arrived at this easy-to-use design through an iterative process. Their interdisciplinary team (consisting both of security experts and ethnographers) first designed a prototype that they then tested with users for usability flaws. The test users indeed pointed out issues with the prototype that were addressed in the final design. The timing and satisfaction results cited earlier refer to that final design.

Case Study: Casca

Casca is an application that shows how small, dedicated iPKIs can be used to secure work group applications.[16] In Casca, users create shared virtual spaces for collaborating with other users. Each space is represented by an (initially empty) canvas on the user's screen. Users can invite others to join a space, and such members can then add objects to that space (e.g., files, cameras, or speakers) by dragging objects' representations (i.e., icons) onto the space's canvas (see Figure 16-2). For example, if Alice invites Bob to join her space, and then drags her screen component onto the space's canvas, Bob (along with any other member of that space) will have access to Alice's screen. (Bob will see an icon representing Alice's screen on his canvas, and will be able to display documents on Alice's screen by simply dragging those documents onto the icon.)

The shared spaces provide a natural basis for security: only users that are members of a space can access the objects in that space. By making sharing a top-level primitive of the application, the design makes it easy for users to understand what they are sharing and with whom. Moreover, users don't have to do anything extra in order to specify who

14 Balfanz, Network-in-a-Box.

15 See Chapter 13, this volume.

16 W.K. Edwards, M. Newman, T.F. Smith, J. Sedivy, D. Balfanz, D.K. Smetters, H.C. Wong, and S. Izadi, "Using Speakeasy for ad hoc Peer-to-Peer Collaboration," *Proceedings of the ACM 2002 Conference on Computer Supported Cooperative Work* (CSCW 2002), (ACM Press, 2002), 256–265.

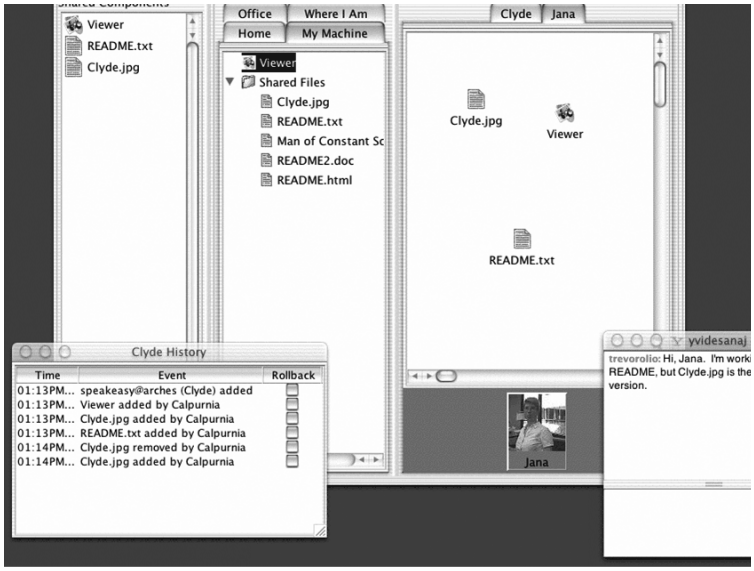

FIGURE 16-2. *A screenshot of the Casca application*

should have access to (and who should be barred from access to) shared resources: assembling a work group to participate in a shared virtual space automatically sets up a security mechanism that enforces access control along work-group boundaries.

It turns out that instant public key infrastructures are an ideal tool to implement the security requirements for Casca in an easy-to-use manner. Whenever a user creates a new blank canvas (thus creating a shared space that only he is a member of), a new root certificate is created for that shared space. The creator of a shared space essentially becomes the certification authority for that space (of course, the user is unaware of this). Inviting others to join the space means issuing them a new certificate signed by the space's root public key. The new member's public key is authenticated using location-limited channels (see the previous case study). Two members of a space authenticate each other by proving that their respective certificates stem from the same root. Again, all this happens under the covers—users do not see the setup or use of certificates.

The user experience is similar to the previous case study. In order to invite a new member, *some* sort of action is necessary on behalf of the group owner. In Casca, this happens to be a gesture that aligns the infrared ports of the group owner's computer and the new member's computer. As a result of the gesture, the group owner sees a new member alongside the group's canvas. The new member sees the group's canvas and immediately has access to all objects made available through the shared space. In addition, this also serves as an example of Yee's guideline:

> "Enable the user to express safe security policies in terms that fit the user's task."[17]

In Casca, users assemble groups of colleagues with whom they wish to collaborate. This is a very task-centric activity (Casca is a *collaboration* tool). At the same time, this group of

colleagues also represents a security policy in that it specifies that these and only these collaborators have access to the group's converspace.

Case Study: Usable Access Control for the World Wide Web

Instant PKIs do not always have to use location-limited channels to enroll new clients. Balfanz describes a system where web servers themselves take on the role of certification authorities, issuing certificates for potential clients of the web sites they're serving.[18] The system is designed for small-time web publishers such as hobby photographers or keepers of online diaries who want to restrict access to their online content to a select group of people. After they create new content, Balfanz suggests that a natural part of the "workflow" for these small-time web publishers is to inform their clients of the new content by announcing to them (and only to them) the creation of the new content. This intention of access control (which is expressed, say, by sending an email announcement to a select group of people) is enforced by a small, dedicated iPKI: when a recipient of such an announcement visits the web site for the first time, he automatically receives a certificate issued by the site (as opposed to some general-purpose certification authority). This certificate is then used to authenticate to the site, during both the first and subsequent visits.

Why does the site issue a certificate to any (!) client that connects, only to turn around and test for the possession of such certificates as proof of access? The idea is that the legitimate recipient of the announcement email will visit the site before any attacker, thus obtaining the certificate. The web site honors only the first certificate request from each user; subsequent requests are rejected. If, for some reason, an attacker manages to visit a web site before the legitimate recipient of the certificate does, that legitimate user will encounter an error message asking him to contact the web site's publisher. Upon such contact, the publisher can easily revoke the existing certificate (assuming that it was issued to an illegitimate user), and instead issue a new certificate to the legitimate user.

This threat model is similar to the widely used tool *ssh*, where there is a small chance of a man-in-the-middle attack on the first connection to a remote host. If and when such a first connection is successful, subsequent connections are secured by the full strength of the underlying protocol (in this case, SSL or TLS with client authentication).

The result is a system that, while protected with an iPKI, is to the users almost indistinguishable from a system that does not employ any security mechanisms at all—that is, the additional security does not place an additional burden on the user. As in an insecure system, web publishers publish information through a web server and announce this fact in an email message to a select group of people. And as in an insecure system, recipients of the announcement click on a link in that message and have immediate

17 Yee.

18 D. Balfanz, "Usable Access Control for the World Wide Web," *Proceedings of the Annual Computer Security Applications Conference*, IEEE Computer Society (2003), 406–415.

access to the published content. The only difference in the implemented system is that during their first visit, clients encounter a few "setup" screens they have to walk through. The users (both publishers and clients) never notice that they are managing, and partaking in, a public key infrastructure. What's more, this is actually an example of Yee's guideline:

> "Match the most comfortable way to do tasks with the least granting of authority."[19]

Because the secure version is (almost) indistinguishable from the insecure system, it represents a mechanism that users are already very comfortable with. At the same time, however, authority is granted to no one but the individuals addressed in the announcement.

Instant PKIs

The examples in the previous sections show that, with careful design, public key technology can be made not only useful but also *usable*. A set of five principles for secure and usable system design can be drawn from these and other examples.[20] Together they suggest a general new approach to usable PKI design:

You can't retrofit usable security
: Just as security cannot be "bolted on" to an existing system late in its design, existing unusable systems cannot be fixed simply by adding a better GUI. Systems must be designed to be both secure and usable from the ground up.

Tools aren't solutions
: Components like SSL/TLS, public key cryptography, and digital certificates are important resources in building secure systems. However, by themselves, they are not solutions to a user's problem—they must be applied effectively in a well-designed application context.

Keep your customers satisfied
: The only way to tell if a system is usable is to ask users—usability testing is key to being sure that a system's design is successful.

Think locally, act locally
: Make technology easier to deploy and use by tailoring it to the application at hand, and by avoiding external deployment dependencies.

Mind the upper layers
: Users are trying to accomplish goals expressed in the context of the application at hand. Don't ask them to make security decisions or take actions outside that context—use the context to automate necessary security steps.

19 Yee.

20 Balfanz, "In Search of Usable Security."

While the first three of these are important general principles for the design of any usable, secure system, the last two lead to very specific indications of how we should design a usable PKI. *Think locally, act locally* suggests that we should avoid designs that require establishing large, coordinated trust hierarchies, or certificates that are predesigned to fit every possible future application. This not only simplifies deployment by reducing the need for interorganization coordination, but it also allows greater opportunity for autoconfiguration.

Mind the upper layers suggests that users should be allowed to focus on the task at hand, and that systems should take contextual information about what is needed to accomplish that task into account in order to figure out what security-related actions need to be performed. Users should not be asked to think about certificates directly—either how to obtain one or whether to trust a particular CA. An increasing amount of research and experience finds that users are very confused by such requests.[21,22,23,24] Instead, certificate management should be integrated into the task at hand. For instance, in the Network-in-a-Box system described earlier, users were asked to indicate by a gesture what wireless network they wished to join. The fact that joining that network required them to obtain a certificate was not something they had to find out explicitly.

This approach to simplified PKI deployment can be generalized and reused for a wide variety of applications. When used in combination with intuitive approaches to initial user authentication like the gesture-directed approaches described earlier, the combined system is secure, easy to use, and easy to deploy. As we've mentioned, the term *instant PKI* (iPKI) is used throughout this chapter to describe such systems.

What Makes a PKI Instant?

An iPKI is a lightweight PKI centered around a standalone CA—one that doesn't participate in a larger CA hierarchy. Certificates issued by that CA are usually used for a single, specific application. The goal of an iPKI is to make it so simple and easy to issue certificates and configure them for use (while not sacrificing security) as to make it easier to issue a new certificate than reuse an old one. These small, special-purpose PKIs turn out to be considerably easier to configure and set up than a reusable PKI—most often, PKI setup can be completely automated. And because the certificates issued are closely linked to an application context, it's much easier to make enrollment and certificate use comprehensible and easy for the end user, who prefers not to have to think about certificates at all.

A number of elements are common to iPKIs:

21 *Ibid*.

22 Doyle.

23 Gutmann.

24 Whitten.

CASE STUDY: USABLE PKI DEPLOYMENT FOR AN ENTERPRISE WIRELESS NETWORK

With the principles for instant public key infrastructures in mind, we redesigned the procedure for joining the wireless network at PARC described in the earlier sidebar. We reused many of the design choices from the NiaB system to greatly simplify the user experience.

A user wishing to join a laptop to the wireless network walks to a room that houses an *enrollment station*. The user installs and launches a client application that instructs her to align the laptop's infrared port with that of the enrollment station; she is then told that a certificate request has been received, and she can leave the room. After administrator approval, the user receives an email asking her to rerun the client application, which automatically retrieves the certificate and installs it for use with the wireless network.

This system provides an intuitive trust model. Our enrollment station is locked in a room, so someone who has physical access to the room can request a certificate, while someone who doesn't have physical access cannot. At PARC, users need to present their badges to a system administrator to enter the enrollment room.

Usability studies demonstrate that this approach is simpler and more intuitive for end users than setting up security using traditional methods. It takes users an average of 1 minute and 39 seconds, and a total of four steps, to add a new device to the wireless network. The system also gets much higher marks than the traditional system in user satisfaction and confidence. Even our system administrators, who by this time had become quite familiar with the traditional system, prefer the new system to the point where they exclusively use it to enroll end users' computers.

Application-specific, lightweight CA

An iPKI is used for a particular narrow application context (e.g., to identify all the devices that can access a particular wireless LAN, or all the devices that are allowed to use a shared repository of resources). Instead of reusing existing certificates, in an iPKI it's often easier to get a new one for a different application or different trust condition.

Automated PKI and CA setup

To make it possible to deploy many lightweight iPKIs, it must be as simple as possible to set each one up—ideally requiring no user intervention at all.

Simple, intuitive enrollment mechanism

To make it feasible for users to obtain multiple certificates, it must be simple for them to get each one. More importantly, each certificate and enrollment action should be embedded in a task the user knows he wants to perform (e.g., enrolling in a particular wireless LAN or inviting someone to participate in a shared repository of resources). The user should be asked to do nothing more than would be necessary to indicate

what he wants to do, such as pointing out the AP serving the WLAN he wants to join via IR, instead of being asked to do separate, security-specific steps.

A simple, intuitive trust model

It should be simple and clear to the user who will and will not be able to join the iPKI. More importantly, that access model should be expressible in application terms. In the NiaB example described earlier, only people who can get physical access to the NiaB AP (close enough to exchange IR) can get onto its wireless network.

Secure bootstrapping

Making the enrollment mechanism intuitive doesn't mean that it should also then be insecure—giving out certificates without enforcing a well-thought-out trust model for who should get them, and effectively enforcing that model. In the examples presented in this chapter, authentication over location-limited channels provided a strong cryptographic binding between the user's intention (e.g., demonstrating a desire to join a WLAN and physical access to the AP and pointing out who he wants to join his shared space), and the system trust model. This form of ad hoc authentication acts to bootstrap trust in the resulting PKI.

Certificates as capabilities

Although an iPKI may issue standard X.509 identity certificates, these certificates are rarely used for their ability to provide identity information. Instead, they usually play the role of simple, easy-to-verify capabilities—"anyone owning a valid certificate issued by this iPKI is allowed to do X," where X might be participating on a particular network, accessing a particular set of files, etc. (Ownership here is defined in the sense of proving possession of the private key corresponding to the public key in a certificate signed by the iPKI's CA.) This simplifies the design and implementation of applications that will use the PKI.

No need for direct user interactions with certificates

One of the corollaries of automated enrollment is that users don't have to directly manipulate certificates or even know they are in use. This improves the user experience dramatically.

The "instant PKI" approach is not an all-or-nothing one—it's simply a collection of design choices that make it easier to deploy small, usable PKIs. Subsets of these design choices can be combined with traditional PKI techniques to generate all sorts of *hybrid* solutions, which are more usable than traditional PKI enrollment, while still keeping centralized management features that may be necessary in certain organizations or situations. The earlier sidebar, "Case Study: Usable PKI Deployment for an Enterprise Wireless Network," shows an example of such a hybrid.

We close this chapter by remarking that many of the concepts discussed here are applicable above and beyond CA-based infrastructures that use public key certificates. An instant public key infrastructure is really only one example of an *Instant Trust Infrastructure* (ITI). Instant Trust Infrastructures may differ in the security mechanisms they employ: certificates, shared secret keys, more complicated forms of cryptographic

credentials, etc. What they have in common is that they are just the right size, are application specific, and provide an easy bootstrap mechanism and an intuitive security model. Armed with those properties, Instant Trust Infrastructures in general, and iPKIs in particular, show us how to make the impossible easy and combine usability and security in one system.

About the Authors

Dirk Balfanz joined the Palo Alto Research Center (formerly Xerox PARC) in 2001 after receiving his Ph.D. from Princeton University for work on distributed systems security. Dirk is interested in end-to-end security, which means the correct interplay of human factors, systems security, and cryptography in distributed systems such as the Internet. Most recently he has focused on usability issues for securing computer networks. For recent publications, see his web site:

http://www.parc.com/balfanz

Glenn Durfee is a member of the computer and network security research group at the Palo Alto Research Center. His interests range from mathematical cryptography to the development, implementation, and deployment of next-generation security systems. Currently he has been especially interested in security for wireless and mobile devices. Glenn received his Ph.D. in Computer Science from Stanford University in 2002, and has since worked at the Palo Alto Research Center. More information and a list of publications can be found at his web site:

http://www.parc.com/csl/members/gdurfee

Diana Smetters has been a member of the security research group in the Computer Science Laboratory at the Palo Alto Research Center since 1999. Her current research focuses primarily on the usability of security and security for wireless networks and mobile devices, and around problems of key distribution and key management. She obtained her Ph.D. in 1995 from MIT. Additional biographical details and a list of publications can be found at her web site:

http://www.parc.com/smetters