PorKI: Portable PKI Credentials via Proxy Certificates[*]

Massimiliano Pala, Sara Sinclair, and Sean W. Smith

Computer Science Department
PKI/Trust Lab, Dartmouth College
6211 Sudikoff, Hanover, NH 03755, US

`{pala,sinclair,sws}@cs.dartmouth.edu`

Abstract. Authenticating human users using public key cryptography provides a number of useful security properties, such as being able to authenticate to remote party without giving away a secret. However, in many scenarios, users need to authenticate from a number of client machines, of varying degrees of trustworthiness. In previous work, we proposed an approach to solving this problem by giving users portable devices which wirelessly issue temporary, limited-use proxy certificates to the clients. In this paper, we describe our complete prototype, enabling the use of proxy credentials issued from a mobile device to securely authenticate users to remote servers via a shared (or otherwise not trusted) device. In particular, our PorKI implementation combines out-of-band authentication (via 2D barcode images), standard Proxy Certificates, and platform attestation to provide usable and secure temporary credentials for web-based applications.

Key words: Mobile Authentication, Usable Security, Website Authentication, Proxy Certificates, PKI

1 Introduction

Usability is critical to the success of a secure computer system [19]. In particular, the user's experience in performing common actions—such as authenticating to the system—has a strong impact on their overall engagement with the system [15]. Authentication schemes based on public key cryptography offer more rigorous security guarantees than passwords, but the overhead costs of obtaining credentials and configuring them for use on commodity machines makes PKI essentially unusable in many domains. Moreover, there is no clear-cut way to use a private key on a potentially untrustworthy workstation without exposing that key to compromise.

[*] This work was supported in part by Intel Corporation and by the NSF, under grant CNS-0448499. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

At the same time that these challenges prevent PKI from gaining wide adoption among end users, we are becoming increasingly dependent on remote systems to store and process sensitive data (e.g., in the cloud). As this dependence grows, so too does our awareness that password-based authentication is glaringly insufficient.

This work. We therefore propose a novel approach to user authentication from a client machine. Our solution—named *PorKI* because it enables long-term PKI credentials to be portable across machines—leverages the user's smart phone as a credential repository, and generates limited-use proxy certificates for short-lived key-pairs for use on potentially untrustworthy workstations. This paper focuses on an implementation of PorKI targeted at authentication to web-based resources; we also present software for the workstation that allows the user to authenticate to websites using the proxy certificate she generates on her smart phone.

Paper Organization. In Section 2 we summarize related work and briefly survey existing mechanisms for user authentication on the web. We describe the design of the PorKI system in Section 3, and discuss details of our implementation in Section 4. How to enable PorKI authentication in Web Applications is discussed in Section 5. Finally, Section 6 provides our conclusions proposals for future work.

2 Related Work

This is a brief overview of related work; for a more complete treatment, see the earlier paper on the design of PorKI [14].

The predominant mechanism for authenticating users on the web is the classic secret password, despite its vulnerability to phishing and other attacks. RSA's SecureID [12] adds an additional factor of authentication, although deploying tokens may not be appropriate for all organizations. Smartcards and USB PKI tokens face similar cost and logistical challenges; moreover, a compromised workstation can take advantage of the access to the user's keypair, even with user- and system-level controls in place [7]. Instead of reducing the size of the Trusted Computing Base (TCB) to fit a secure co-processing unit as in the SHEMP project [6], our work uses an external device (i.e. a smart phone) as a personal TCB.

Work by Wu et al [9] leverages a cell phone and a trusted third party to protect credentials from the workstation. Different from our solution, their solution requires the user to input data to authenticate a specific session instead of automatically generate short-lived strong proxy credentials while authenticating to a remote server. Sharp et al. [13] augment the workstation interface with a trusted browser on the user's mobile device, which can then be used to securely enter passwords or other sensitive information. Garriss et al. [5] present a system with which a user can use a mobile device to verify the identity and integrity of software on the workstation; after the verification process is complete, the user may be more confident about sharing his credentials with the machine.

PorKI's system for establishing a trusted channel of communication between devices—via the mobile device's camera and a 2D barcode displayed on the workstation—is similar to that of the Grey System [8]. In their work, the authors use the discovery functionality of Bluetooth and barcode images to securely transfer ephemeral keys or self-signed certificates. In our work, we leverage their technique to allow pairing of the devices over IP network (i.e., without requiring the Bluetooth discovery properties)—but we differ by providing support for standard proxy credentials (proxy certificates) for web-based applications.

2.1 Proxy Certificates

The problem of exposing a user's credentials when accessing services on public or shared computers is well-known. As the sharing of strategic resources grows beyond a single campus or organization (e.g. in computing grids), the need grows to provide simple ways to use short-lived or ad-hoc credentials demands for usable solutions.

We think that the usage of a mobile device and the possibility to easily transfer proxy-credentials to the shared device provide a better security paradigm and a more usable solution.

To allow for simple authentication delegation, the IETF standardized Proxy Certificates allows a user to issue herself a certificate which enables only a subset of the permissions present in her long-lived identity certificate. Proxy Certificates use the X.509 standard with the prescriptions described in RFC 3820 [18]. Proxy Certificates bind a public key to a subject name, as normal identity certificates do. The use of the same format as X.509 public key certificates allows Proxy Certificates to be compatible with existing protocols and libraries which significantly eases implementation. However, unlike a public key certificate, the issuer (and signer) of a Proxy Certificate is an End Entity (EE) or another Proxy Certificate rather than a Certification Authority (CA). Therefore, Proxy Certificates can be created dynamically without requiring the normally heavy-weight vetting process associated with obtaining EE certificates from a CA. Intuitively, this approach is safe because it only permits the EE to delegate a subset of its own identity and privileges to this new, temporary keypair.

As the standard mandates, the subject name of a Proxy Certificate is scoped by the subject name of its issuer to achieve uniqueness. This is accomplished by appending a CommonName relative distinguished name component (RDN) to the issuers subject name. This means that the *subject* of the proxy certificate (or *Distinguished Name*) has to include the user's original one plus an additional *CommonName* (CN) field whose value is, usually, set to "Proxy". Moreover, the user can include only a subset of her own identity certificate permissions into her proxy credentials. That is, let ϕ be the set of original permissions in the user's certificate χ and ϕ' the set of permissions present in the proxy certificate χ'. Then the set of allowed permissions ψ is a subset of the one present in the user's certificate χ as restricted by the proxy certificate's permissions, i.e. $\psi = \phi \cap \phi'$. In particular, the user can limit the validity period, the Key/Extended Key usage, and embed a usage policy.

By using proxy certificates, temporary strong short-lived credentials can be easily issued (matching temporary, short-lived key-pairs). This approach limits the exposure of the user's credentials to a very short temporal window, thus allowing their usage also on shared computing devices as their expiration guarantees that they can no longer be used beyond the intended period. Although this approach has been deployed within some communities, such as scientific grid computing, its main drawback is that revocation of these type of certificates is very inconvenient. Therefore, such approaches(e.g., [2]) completely avoid publishing revocation information. Moreover, although requesting a new certificate is a quite frequent operation (because of the limited validity period), the process can be either quite cumbersome on the user part or rely on very lightweight authentication procedures.

Although already standardized, proxy certificates are not yet widely adopted outside specific communities of users, such as grid computing.

3 System Design

Suppose Alice wants to authenticate to a remote relying party Bob through a client workstation W that neither Alice nor Bob trust completely. Alice could use a PKI-equipped USB token—but this requires that W contain drivers for this token, and (what's worse) can expose Alice's key to malicious use by W [7].

PorKI enables Alice to do this without exposing her identity private key—but without having to do any complicated work or assuming anything special hardware properties of W. In particular, our work combines the usability and ubiquity of mobile devices, the security of out-of-band communication for secure key management, and the compatibility of legacy X.509-based authentication with proxy-credentials delegation.

In PorKI, the main idea is to generate and transfer ad-hoc proxy credentials (i.e., the proxy certificate and the corresponding private key) from a mobile device (such as a smart phone) to W (e.g., a shared lab computer or an internet-cafe terminal). These newly generated credentials can be used to securely authenticate Alice to Bob via standard protocols (e.g., SSL, TLS, or DTLS) applications already use. By using the proxy keypair, the Alice's long-term one is kept safe on the mobile device and never transferred to the shared device W.

In our work, we identified three main components, i.e. (1) the mobile device application, (2) the shared device extension that manages PorKI's operations, and (3) Proxy Certificates enabled services and applications. In this section we focus on the description of the first two components. We provide considerations on leveraging the information embedded in the user's proxy credentials in section 4.

PorKI in a Nutshell. The user activities required to issue proxy-credentials via PorKI is depicted in Fig. 1. The mobile device is paired with the shared one

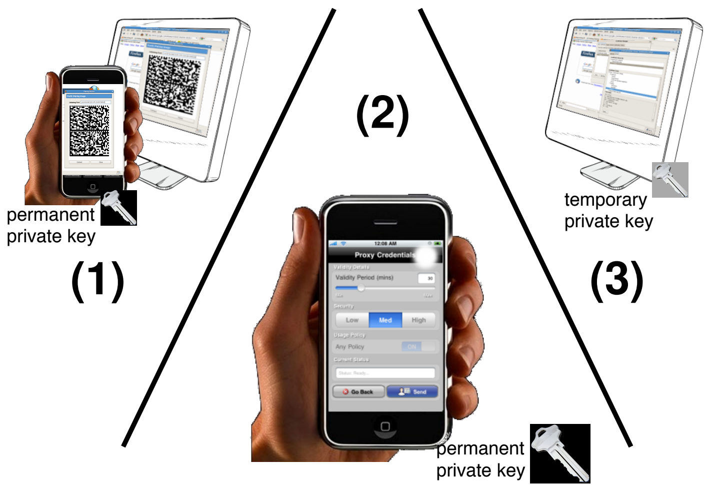


Fig. 1. User interactions with PorKI: first (1) the user pairs the mobile device and the shared workstation using the former's camera to read the barcode displayed on the latter. Next (2) she has the option to modify the parameters according to which the proxy certificate with be generated; when she taps the *send* (highlighted) button, the credentials are transferred. The user can now (3) use her short-term credentials in the browser on the shared workstation to authenticate to web resources.

(e.g., Internet station) via barcode images generated on the latter that carry the channel authentication information needed to secure the communication between the devices. Once a secure communication channel has been established between the two devices (via this pairing), trust measurements about the shared one (eg., TPM attestation) are transferred to the mobile device and embedded into the proxy credentials. Then, the new certificate and keypair are transferred back to the paired device together with the full chain of certificates up to the trust root.

As described later in 3.2, we extended the Firefox browser to act as the PorKI-enabled application on the shared device. Once PorKI is activated and the proxy-credentials are transferred to the browser on the shared device, the user can authenticate herself to web applications by using her new proxy-credentials.

3.1 Smart Phone Application

PorKI leverages the possibility to securely store the user's long credentials into a mobile device that the user is acquainted with. In particular, as phones and other smart devices like iPods or PDAs are very popular, we leverage the user's familiarity with these devices in order to provide usable proxy credentials.

Assumptions. To securely pair the mobile device with the shared one (W), we assume that:

– The shared resource W has a display capable of displaying images of 400 by 400 pixels. Since we use black and white images, the display does not need color capabilities.
– The mobile device is capable of taking VGA resolution pictures. Although it is possible to use lower resolutions for detecting the barcode displayed on the shared resource, the usage of VGA resolution reduces the detection error rates and increases the usability of the mobile application. We notice that today even low-cost mobile phones carry camera with higher resolution.
– The mobile resource is capable of contacting the shared resource via IP (e.g., over wifi, 3G, or EDGE networks).

As the third assumption is quite important, it requires further discussion. To transfer the shared device's trust measurement and the proxy credentials back to the shared device, PorKI needs a secured communication channel between the two devices. In our previous work [14], we focused on the usage of bluetooth [1] as the out-of-band channel for transferring the user's proxy credentials, but this proved to be too restrictive as most of the shared resources (eg., desktop computers) do not come with bluetooth capabilities. Therefore, we divided the pairing process into two separate phases. During the first one, we transfer only the information needed to establish the secure communication channel (a symmetric 128-bit key) by using barcode images as the out-of-band channel. During the second phase, we establish the secure channel and transfer the information back and forth the two devices.

In other words, we use the out-of-band communication only for transferring the shared resource's (W) network address and the encryption/integrity key to securely pair the devices (i.e., setup the encrypted communication channel). Subsequent data transfers happen via IP communication. This assumption is reasonable as the two devices are physically in the same location and, in many cases, share the same LAN segment. Although this approach cannot be applied to every possible scenario (e.g., because of the presence of firewalls), we think it is the least restrictive assumption possible.

The User Interface. In PorKI, we designed the user interface in order to minimize the steps required by the user to issue and transfer the proxy credentials among devices. The application workflow is as follow:
1. The User starts the application and take a picture of the barcode image displayed on the shared resource display
2. The application automatically decodes the information from the image and establishes the secure communication channel (via TLS)
3. The User selects one of the stored credentials on the mobile device and sets the validity period of the proxy credentials via a simple bar selector
4. The mobile application generates the new keypair and issues the new proxy certificate.
5. The mobile device sends the proxy credentials in an encrypted PKCS#12 file to shared device over the authenticated communication channel

It is to be noted that the required level of interaction with the mobile device application is sensibly low (steps 1 and 3). In fact, compared to other solutions that

```
id-porki        OBJECT IDENTIFIER ::= { id-pkix 50 }
     -- Object Identifier for the porkiDeviceInfo extension

porkiDeviceInfo ::= SEQUENCE {
     retrievedAt                GeneralizedTime,
        -- time when the Info has been collected
     retrievedBy               INTEGER,
        -- identifier for the component which gathered
        -- the device information
     data                      SEQUENCE OF OCTET STRING }
```

Fig. 2. ASN.1 notation for the `porkiDeviceInfo` extension that is used in the proxy certificate to store the shared device information.

require pairing the two devices via bluetooth, our approach based on barcode images proved to be quite effective.

Proxy Credentials. By being able to directly issue proxy credentials (as opposed to have to request them from a third party), it is possible to embed the shared device's authentication information in the proxy certificate. For this purpose, we identified a new extension (i.e., porkiDeviceInfo) that allows for unstructured content to be embedded. In our application, we include a simple XACML assertion that carries information about the shared device that have been gathered by the shared device's PorKI application (i.e., the Firefox extension). The ASN.1 notation for the extension syntax is reported in Figure 2. This extension carries several fields. The `retrievedAt` is the time at which the measurement is performed. The value of this field can be used to evaluate the freshness of the information. In particular, it is possible to use cached values if the measurement is sufficiently recent, thus allowing for shorter communication between the paired devices if a recent pairing has recently took place. In `retrievedBy` we store the identifier of the component that performed the measurement. We identify with `0` the mobile device—the core of trust for PorKI. The browser extension is identified by `1`, while any external component is identified by `2`. Currently, the trust measurements are performed directly on the shared device and then transferred to the mobile device. In future versions of PorKI, we envisage gathering more reliable information about the shared device by performing a remote attestation of the platform [17]. If a remote attestation is performed by the mobile device, a higher level of trust for the performed measurements can be adopted by the web application. In this case, the presence of the `retrievedBy` field allows the relying party (e.g., the web application) to raise or lower its confidence in the porkiDeviceInfo contents. Ultimately, if the remote attestation cannot be performed (e.g., because of the lack of the TSS stack on the shared device), the current mechanism based on self-attestation on the shared device can be used as fallback option.

```
certIssuingLocation ::= SEQUENCE {
      latitude               UTF8String,
         -- latitude information
      longitude              UTF8String,
         -- longitude information
      elevation              INTEGER,
         -- elevation (in meters) information
      levelOfAccuracy        INTEGER
         -- level of accuracy (in meters)
}
```

Fig. 3. ASN.1 notation for the `certIssuingLocation` extension. The information carried in this extension can be used for authentication, authorization and auditing purposes.

Location-Aware Certificates. In PorKI, we enabled the possibility to embed location information in the proxy-credentials. In fact, if the mobile device provides support for GPS or GSM positioning, the information is embedded in the proxy certificate in the form of another extension. For this, we identified the `certIssuingLocation` extension as defined in Figure 3. Because the level of accuracy of GPS/GSM positioning is usually very low inside buildings, the location information includes the `levelOfAccuracy` field which has to be taken in consideration when the location information is consumed. Also in this case, because the mobile device is the source of trust in PorKI, the location information embedded in the proxy credentials can be considered trustworthy under the assumption that the device is not subject to a GPS spoofing attack. Although we notice that this attack is easy to carry out [16] and that there is no protection against it in current GPS receivers, it is possible to detect GPS spoofing attack by combining and analyzing location information gathered through both the GSM and the GPS networks. On mobile devices that do not provide the possibility to access a second source of location information (e.g., PDAs), several proposals have been made on how to fix this problem [4, 10]. We hope that the next generation of GPS receivers will provide also authentication information to allow applications to recognize the level of trustworthiness of the signal (e.g., by using information about the time, noise level and strength's of the GPS signal, reported accelerations' sanity checks).

In PorKI, both authentication and authorization engines of web applications could use the location information to grant or restrict access to specific resources depending on the physical location of the user. The information collected by the mobile device, could be of value for several scenarios. Besides using the user's position for authorization purposes, this information could be used later on for auditing purposes.

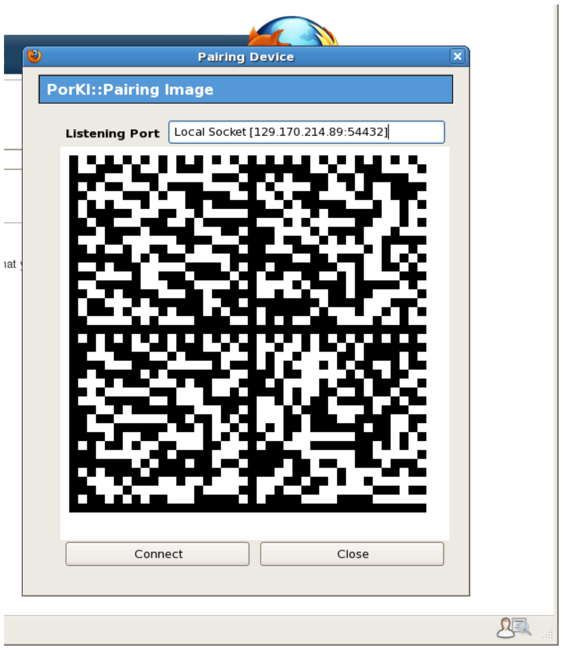


Fig. 4. Barcode image generated by the PorKI application. The image encodes the shared device IP address and the shared session key.

3.2 The Browser Extension

In our work, a central component is the PorKI application that enables the shared device to transfer and install the temporary proxy credentials.

Because our implementation is aimed at providing support for web-based applications, we chose to develop the PorKI application as an extension for the Firefox browser. This choice has several side-benefits. First, the installation of extensions for the browser is quite an easy process and users are already used to it. Moreover, the extension code can be digitally signed thus allowing for code verification and automatic updates.

Although our choices were driven by the specific example scenario of Alice authenticating to a Web site operated by Bob, the general design of the PorKI application can easily be applied to other contexts. For example, operating systems could provide an update/extension process for their certificates store that would allow the installation/creation of a PorKI-enabled store. It is out of the scope of this work to discuss all the possibilities offered by the different OSes or software distribution technologies.

Out-of-Band Data. The browser extension is responsible to generate the barcode image (Figure 4) that is used to setup the secure communication channel between devices. As the first step, the shared device generates a session key that is used to establish the encrypted and integrity-checked channel with the mobile device. The same key is also used to decrypt the PKCS#12 file that contains the proxy credentials. Together with the generated symmetric key (3DES), the network IP address of the shared device is embedded in the 2D barcode that is displayed to the user when she requires to transfer/generate a new proxy credential. Once the secure communication is established between the two devices, the PorKI application sends the details about the shared device to the mobile one. This data can be in any format and is meant to be consumed by the web-application. It is to be noted that, besides initiating the process, the user is not asked to interact with the PorKI application on the shared device.

4 Implementation Details

During the development of PorKI, we identified two main challenges: providing support for Proxy Certificates across operating systems and devices, and enabling the usage of Proxy Certificates for SSL/TLS authentication in Firefox.

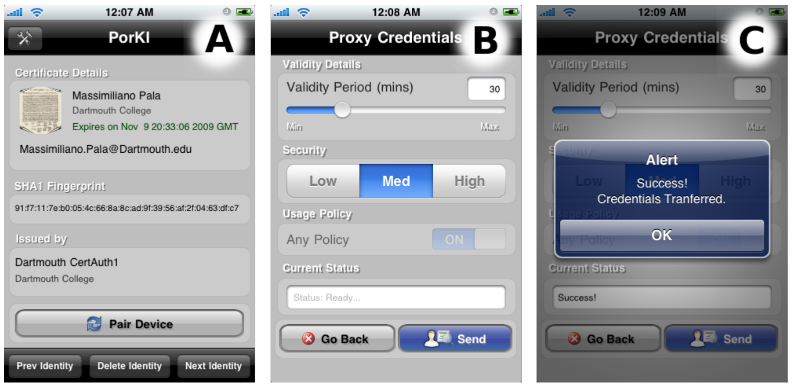
To address the first issue, we used and enhanced LibPKI [11] to support proxy certificates. In particular, we added support for `PKCS#12` tokens manipulation and proxy certificates profiles management.

The second issue proved to be difficult to solve as much legacy software does not allow easily extension of the crypto APIs in order to support proxy certificates. The main problem in adding support for proxy certificates is the path building process. In fact, according to RFC 5280 [3], an End Entity (EE) cannot sign certificates. Therefore, the path construction and validation for a proxy certificate fails in crypto libraries that do not explicitly support them. Thus, solving this problem can potentially require to directly patch large parts of the browser software. Fortunately, this was not required in our implementation. As we describe in 4.2, we leverage the properties of Firefox's certificates store to alter the normal behavior of the browser and convince it to use the imported proxy credentials as normal identity certificates.

4.1 iPhone and Proxy Certificates via LibPKI

In order to develop the PorKI mobile application, we needed a platform that was able to provide the required functionality out of the box or that would allow us to port existing software to the device in an easy fashion. Fortunately, the iPhoneOS and its development environment allow for UNIX programs to be ported by using open-source tools like GCC. In particular, to address our needs for cryptographic functionality, we were able to compile LibPKI by using the iPhone SDK (v2.0, v3.0, and v3.1.3). The possibility of using LibPKI instead of the native cryptographic functionality built into the iPhoneOS helped us in:

– reducing the size and complexity of the iPhone application
– support proxy-certificate creation easily
– building secure connection channel with the shared device via the provided `URL` interface



Certificate Details
Massimiliano Pala
Dartmouth College
Expires on Nov 9 20:33:06 2009 GMT
Massimiliano.Pala@Dartmouth.edu

SHA1 Fingerprint
91:f7:11:7e:b0:05:4c:66:8a:8c:ad:9f:39:56:af:2f:04:63:df:c7

Issued by
Dartmouth CertAuth1
Dartmouth College

Validity Details
Validity Period (mins) 30

Security
Low [✓ Med] High

Usage Policy
Any Policy ON

Current Status
Status: Ready...

Alert
Success!
Credentials Tranferred.

Current Status
Success!

Fig. 5. PorKI's user interface on the iPhone: (a) the identity certificate display view, (b) the proxy-credential issuing view, and (c) communication box for the successful transfer of the proxy credentials.

For encoding and decoding barcode images, we used libdmtx[1]. We managed to port this library to the iPhoneOS as well as MacOS X and Linux operating systems. Because all the other libraries we rely on are already available across these systems by default (e.g., libxml2, pthreads, and OpenSSL), we decided to use the same set of tools to develop both the iPhone application and the browser's extension.

Touch User Interface. The PorKI UI on the iPhone provides a usable and clean interface that minimizes the required user interaction. Figure 5 provides some screenshots of the interface. At first, the PorKI application allows the user to select the identity to be used to generate the proxy credentials in the main screen (A). After that, the user can use the "Pair Device" button to initiate the pairing of the device. The standard iPhone picture-taking interface is then displayed and, after the picture is taken, the proxy-credential issuing interface (B) is displayed. In the current version, the user can select the validity period (in minutes) of the proxy credentials and the key length—in the form of a simple "weak", "medium", and "strong" selector—only. In future versions we will explore how to provide the user with meaningful information required to decide about which policy information to be embedded in the proxy certificate. After the validity period is chosen, a tap on the "Send" button will send the newly issued credentials to the shared device and imported into the browser automatically. A simple confirmation popup (C) is displayed to the user upon successful transfer. In case of an error, an appropriate message is displayed instead.

The* `PKCS#12` *Contents. Because the `PKCS#12` format is meant to provide a container for identity certificates, it does not explicitly support proxy-certificates.

[1] The DMTX library is available at http://www.libdmtx.org/

This means that there is no specific "bag" where the proxy-certificate should be stored in. We decided to put the proxy-certificate in the identity-certificate bag and push the identity-certificate in the CA's one. As specified in more details in 4.2, this allowed us to identify the proxy-certificate as a `CERT_TYPE_PROXY_USER` while the user's identity one as a `CERT_TYPE_PROXY_CA`.

The current version of the iPhone application does not allow a user to import her own credentials from an external source. In the prototype we simply embedded several identity certificates in the form of encrypted PKCS#12 files. In our future work, we plan to leverage the `URI` interface in LibPKI to transfer the user's long term credentials from an external repository to the mobile device via different transport protocols (e.g., ldap, https, mysql, postgresq, ssh). By using this approach, a simple URI can be used to configure the application. Both the Firefox Extension and the iPhone source code are available from the project's repository[2].

4.2 The PorKI Firefox Extension

The PorKI extension for Firefox is responsible (a) to generate the pairing images, (b) to import the proxy-credentials, and (c) to enable the usage of the newly issued proxy credentials and setting the appropriate trust configuration in the application store.

Our Firefox extension has two main parts. First, there's the lower-level functionality developed in C++. These functionality are then wrapped by using the XPCOMM interface and exposed to the top layer via JavaScript calls. The usage of LibPKI as our cryptographic provider sensibly contributed to reduce the size of the C++ code. In particular, the compiled part of the Firefox extension is only $\tilde{1}160$ lines of code (the JavaScript part is $\tilde{5}00$ lines of code).

Lower Level Functions. The PorKI lower level API exposes a very restricted number of functions to the upper level. In particular, we expose the following:

- `GetLocalAddr()` provides the IP address of the shared device
- `GenHexKey()` generates the shared key
- `GenBarCode()` generates the barcode image and stores it in a local directory on shared device
- `GetUserProxy()` opens the communication channel with the mobile device and stores the retrieved `PKCS#12` file
- `ImportProxyCertDB()` — imports the proxy-credentials into Firefox's certificate store and sets the appropriate trust configuration

among these, the most interesting function is the latter. In fact, during the design of PorKI we planned to extend Firefox in order to:

- correctly import a proxy certificate
- enable the proxy-certificate to be used as a normal user certificate

Because Firefox provides the possibility to interact with its certificate store and, most importantly, to extend the SSL/TLS callbacks, in our original design

Certificate Type	Trust Flags
`CERT_TYPE_CA`	`CERTDB_TRUSTED_CA, CERTDB_VALID_CA, CERTDB_TRUSTED`
`CERT_TYPE_EMAIL`	`CERTDB_VALID_PEER`
`CERT_TYPE_SERVER`	`CERTDB_VALID_PEER`
`CERT_TYPE_PROXY_CA`	`CERTDB_TRUSTED_CA, CERTDB_TRUSTED_CLIENT_CA` `CERTDB_VALID_CA, CERTDB_SEND_WARN, CERTDB_TRUSTED`
`CERT_TYPE_PROXY_USER`	`CERTDB_VALID_PEER, CERTDB_SEND_WARN` `CERTDB_TRUSTED, CERTDB_USER`

Table 1. Trust Settings for PorKI's Imported Certificates in Firefox Trust Store.

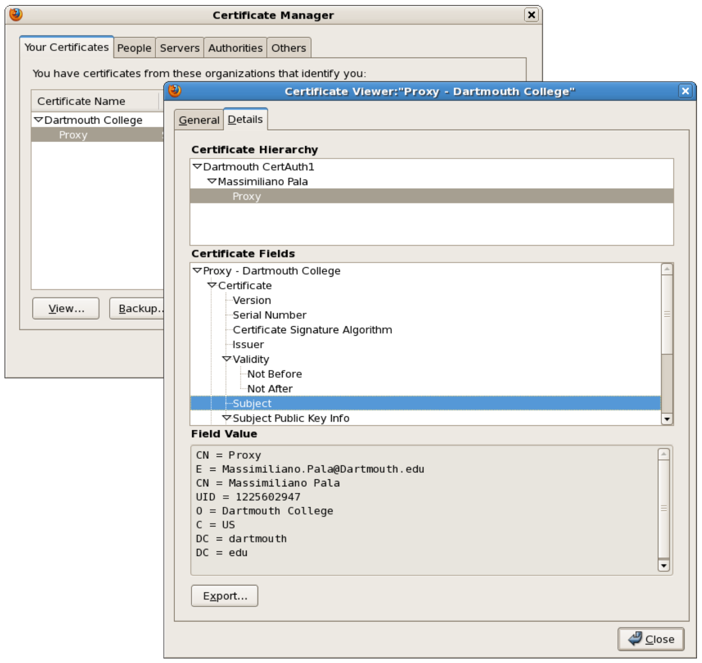
we planned to use both of these features to enable the usage of proxy-credentials for normal browsing operations.

Because of the lack of documentation on the internals of Firefox, the best implementation strategy was not clear. After studying the NSS library (the security library that provides the cryptographic functionality to Firefox) internals, we realized that Firefox heavily relies on the library to store, retrieve, and verify certificates. The proxy-credentials import process has been divided into two functions: `ImportNSSKeypair()`, and `ImportNSSCertificate()`. The first function takes care of importing the private key and set the usage flags in the store. The second one, takes care of importing a certificate in the appropriate store. In PorKI we use the following types of certificates during our import process:

– `CERT_TYPE_PROXY_USER`, which identifies the proxy-certificate issued by the mobile device
– `CERT_TYPE_PROXY_CA`, which identifies the user's identity certificate (the certificate used to sign the proxy credentials)
– `CERT_TYPE_CA`, which identifies any other CA certificate in the chain of certificates up to the user's root CA

When parsing the contents of the `PKCS#12` file, the certificates are imported into the Firefox store according to their type. In particular, after retrieving a reference to the default certificate store, our `ImportNSSCertificate()` function sets the trust flags in the NSS object. Table 1 provides a list of all the trust settings for the different type of certificates and their trust settings. Because of the lack of documentation about the meaning and the effects of each trust setting, we performed some tests to discover the right set of trust settings. Specifically, after importing the certificate with the `CERT_ImportCerts()` function, by setting the appropriate trust settings via the `CERT_ChangeCertTrust()` NSS function, PorKI is able to correctly import the proxy credentials and have Firefox to correctly verify the full chain of certificates. Surprisingly, this automatically enables the usage of the

[2] http://mm.cs.dartmouth.edu/porki/



```
CN = Proxy
E = Massimiliano.Pala@Dartmouth.edu
CN = Massimiliano Pala
UID = 1225602947
O = Dartmouth College
C = US
DC = dartmouth
DC = edu
```

Fig. 6. The Proxy Certificate installed on Firefox as displayed by the standard certificate selector.

proxy credentials in Firefox also for browsing activities. We note that enabling proxy-certificates usage in the NSS library was easier than expected.

Higher Level Functionality. The high-level functionality are provided via the JavaScript portion of the Firefox extension. In particular, we use an Icon on the status bar that the user can interact with. By left-clicking on the Icon, the user activates the generation of the barcode image (`openBarCode()`), while a right-click on the PorKI icon simply opens up the Certificate Dialog where the user can verify her own certificates and identity settings. Figure 6 shows the default Certificate Dialog correctly showing the details of a proxy certificate generated with PorKI.

5 Enabling PorKI Authentication in Web Applications

In order to evaluate the usability of the PorKI system, we evaluated the impact on configuring some PKI-enabled web applications to accept our proxy-credentials instead of the user's identity certificate. In particular, we evaluated a widely adopted systems: GridWiki.

This software is used by many communities of users or researchers to exchange information about their interests or work. We choose this specific software because it supports user authentication via X.509 certificate out of the box.

Our purpose was to determine how difficult would it be to enable the usage of PorKI's proxy credentials (standard X.509 proxy certificates) in GridWiki, and how usable would the user's experience be. Specifically, we installed GridWiki on a Linux machine running Apache v2.2.6.

Because Apache uses the OpenSSL library as its cryptographic provider, in order to enable the support for proxy-certificates for SSL/TLS, we just needed to set the `OPENSSL_ALLOW_PROXY_CERTS` environment variable in the server's startup script. This simple change allowed existing users (users that already registered themselves on the wiki page) to login into the Wiki by using their proxy credentials instead of their regular certificates. Also, no problems were reported when new users registered on the wiki by using their proxy credentials directly.

As reported earlier in this paper, it is interesting to notice how simple the whole setup process was, both from the sysadmin and the user perspectives.

6 Conclusions and Future Work

This work represents a new approach to end-user key management. The PorKI system allows average users to authenticate to remote web resources with their PKI credentials; moreover, it allows those users to authenticate from workstations of varying trustworthiness without exposing their long-term keypair to attack. The tool is flexible, usable, and based on widely-accepted PKI standards; because it also leverages a popular mobile platform and a lightweight browser extension, it can be easily deployed in a variety of settings.

As described in Section 3, we envision using the mobile device to perform remote attestation of the workstation (similar to work by Garriss et al. [5]) during the negotiation of the short-term credentials. Having more trustworthy information about the workstation would allow the remote party stronger confidence in providing the user with access to the sensitive data. As in the SHEMP project [6], we can also envision crafting custom policies to govern the types of access that may be performed using a given set of short-term credentials; we anticipate that corporate organizations who allow their employees to access company systems from personal workstations would have a particular interest in this sort of fine-grained access management.

References

1. Bluetooth SIG: Specification of the Bluetooth System, Core Version 1.2 (2003), `http:/www.bluetooth.org/`
2. Cholia, S., Genovese, T., Skow, D.: Profile for SLCS X.509 Public Key Certification Authorities with Secured Infrastructure (2009), `http://www.tagpma.org/files/SLCS-2.1b.pdf`
3. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (May 2008), `http://www.ietf.org/rfc/rfc5280.txt`

4. El-Bakry, H.M., Mastorakis, N.: Design of Anti-GPS for Reasons of Security. In: CIS'09: Proceedings of the International Conference on Computational and Information Science 2009. pp. 480–500. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2009)
5. Garriss, S., Cáceres, R., Berger, S., Sailer, R., van Doorn, L., Zhang, X.: Trustworthy and Personalized Computing on Public Kiosks. In: MobiSys '08: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services. pp. 199–210. ACM, New York, NY, USA (2008)
6. Marchesini, J.: Shemp: Secure Hardware Enhanced MyProxy. Ph.D. thesis, Dartmouth College, Hanover, NH, USA (2005)
7. Marchesini, J., Smith, S.W., Zhao, M.: Keyjacking: The Surprising Insecurity of Client-Side SSL. Computers & Security 24(2), 109–123 (2005)
8. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In: SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy. pp. 110–124. IEEE Computer Society, Washington, DC, USA (2005)
9. Min Wu and Simson Garfinkel and Rob Miller: Secure Web Authentication with Mobile Phones. In: MIT Project Oxygen: Student Oxygen Workshop (2003)
10. Mundt, T.: Two Methods of Authenticated Positioning. In: Q2SWinet '06: Proceedings of the 2nd ACM International Workshop on Quality of service & Security for Wireless and Mobile Networks. pp. 25–32. ACM, New York, NY, USA (2006)
11. Pala, M.: The LibPKI project. Project Homepage, `https://www.openca.org/projects/libpki/`
12. RSA: RSA SecurID Two-Factor Authentication. RSA Solution Brief (2010)
13. Sharp, R., Madhavapeddy, A., Want, R., Pering, T.: Enhancing Web Browsing Security on Public Terminals using Mobile Composition. In: MobiSys '08: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services. pp. 94–105. ACM, New York, NY, USA (2008)
14. Sinclair, S., Smith, S.W.: PorKI: Making User PKI Safe on Machines of Heterogeneous Trustworthiness. Computer Security Applications Conference, Annual 0, 419–430 (2005)
15. Singh, S., Cabraal, A., Demosthenous, C., Astbrink, G., Furlong, M.: Password Sharing: Implications for Security Design Based on Social Practice. In: CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 895–904. ACM, New York, NY, USA (2007)
16. Tippenhauer, N.O., Rasmussen, K.B., Pöpper, C., Čapkun, S.: Attacks on Public WLAN-based Positioning Systems. In: MobiSys '09: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services. pp. 29–40. ACM, New York, NY, USA (2009)
17. Trusted Computing Group: TCG Specification Architecture Overview. Specification, Revision 1.4 (August 2007), `http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf`
18. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard) (June 2004), `http://www.ietf.org/rfc/rfc3820.txt`
19. Whitten, A., Tygar, J.D.: Why Johnny Can't Encrypt: a Usability Evaluation of PGP 5.0. In: Proceedings of the 8th USENIX Security Symposium. pp. 14–14. USENIX Association, Berkeley, CA, USA (1999)