Journal of Computer Security 18 (2010) 397–420
DOI 10.3233/JCS-2010-0401
IOS Press

Finding the PKI needles in the Internet haystack

Massimiliano Pala [*] and Sean W. Smith
Computer Science Department, Dartmouth College, Sudikoff, Hanover, NH, USA
E-mails: {pala, sws}@cs.dartmouth.edu
URL: http://www.cs.dartmouth.edu

Public key cryptography can uniquely enable trust within distributed settings. Employing it usually requires deploying a set of tools and services collectively known as a *Public Key Infrastructure* (PKI). PKIs have become a central asset for many organizations, due to distributed IT and users. Even though the usage of PKIs in closed and controlled environments is quite common, interoperability and usability problems arise when shifting to a broader, open environment. To make an effective trust judgment about a public key *certificate*, a PKI user needs more than just knowledge of that certificate: she also needs to be able to locate critical parameters such as the certificate repositories and certificate validation servers relevant to that certificate – and all the others the trust path she builds for it. Surprisingly, locating these resources and services remains a largely unsolved problem in real-world X.509 PKI deployment. This issue impacts especially on the usability of this technology and the interoperability of PKIs in open environments such as the Internet.

In this paper, we present the design and prototype of a new and flexible solution for automatic discovery of the services and data repositories made available by a *Certificate Service Provider* (CSP). This contribution will take real-world PKI one step closer to enhancing usability of digital certificates and interoperability between PKIs.

1. Introduction

Public key cryptography enables secure communication and authentication between parties who do not need to share secrets. The benefits coming from the deployment of PKIs have convinced companies and universities to enable the use of digital certificates as an efficient way to manage identities and authenticate their own users. As reported by OASIS, benefits from deploying PKI-enabled applications significantly outweigh the costs of PKI implementation [3].

Unfortunately, these costs are still high. Setting up new PKI-enabled services is equally painful for administrators as well as final users – all of whom are often misguided by badly written User Interfaces (UI) and overly detailed configurations. Every application needs to be properly configured by filling in complex configuration options whose meaning is mostly unknown to the average user (and often to the advanced one as well). Enabling applications to automatically find PKI services (such as OCSP servers, timestamping, etc.) and repositories (e.g., CRLs, certificate

[*]Corresponding author: Massimiliano Pala, 6211 Sudikoff–PKI/Trust Labs, Hanover, NH 03755, USA. Tel.: +1 603 646 8734; E-mail: pala@cs.dartmouth.edu.

pointers, etc.) would relieve users and administrators from this burden, and hence lower these barriers to interoperability and deployment.

Regrettably, Certification Authorities barely publish access details on their official websites; even data as basic as the URLs for the services and repositories they provide are usually omitted. As a result, if a CA provides a new service (e.g., OCSP [27]) or a new data repository (e.g., LDAP [41]), users and administrators have difficulty learning of these changes. Nor would there be any sign of the new service on the certificates that have been already issued. Hence, it is unlikely that users (and applications) will be easily aware of the new services unless someone directly advises them.

This problem has even bigger impact on users from organizations other than the one that issued a certificate; these "foreign" users will usually have very limited knowledge about the specific CA's practices and service locations.

Thus, a new approach is needed to provide a flexible way for PKI users to automatically discover which services and data repositories are available from a CA. This flexibility would also facilitate interoperability across different infrastructures in an open environment (e.g., Web, E-mail, Wireless, etc.). An OASIS survey [18] on issues in PKI deployment survey points out two important aspects (Fig. 1):

- Support for PKI is often missing from applications and operating systems; when present, it is always inconsistent.
- Current PKI standards are inadequate; they are often too complicated, and implementations from different vendors rarely interoperate.

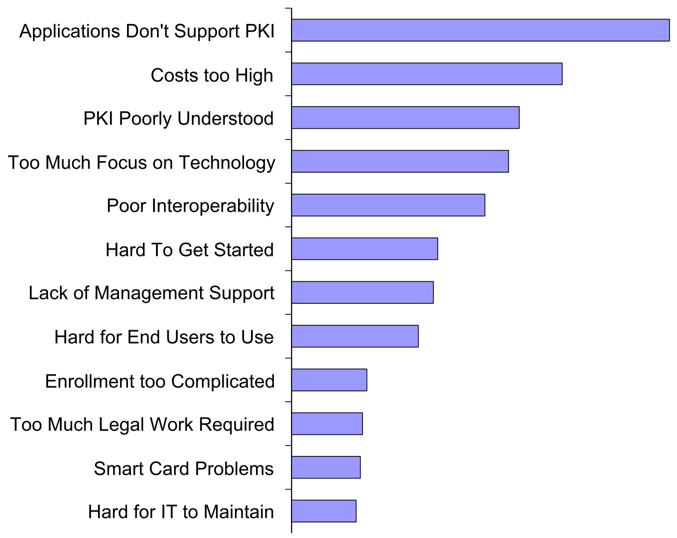


Fig. 1. Our summary of the OASIS survey results – interestingly, seven deployment obstacles out of the first ten are related to PKI usability and interoperability.

It is interesting to notice that seven out of ten reported problems in the list are related to PKI usability and interoperability.

We think that some of these problems can be efficiently tackled by facilitating the location of PKI resources. In this paper, we present a new approach that provides a flexible way to automatically discover which services and data repositories are related to a specific CA. The core of our solution is based on the definition and implementation of a new (and simple) *PKI Resource Query Protocol* (PRQP) which is aimed to ease PKI management both for administrators and final users. The flexibility introduced by PRQP would also facilitate interoperability across different infrastructures. By providing such a mechanism, support for PKI basic operations (e.g., certificates validation) could be easily implemented also at the operating system level.

The outline of this paper is as follows. Section 2 provides an overview of the issue. Section 3 reviews other approaches to solving the problem. Section 4 presents the core aspect of our solution: the design and the implementation of the *PKI Resource Query Protocol* (PRQP). Section 5 presents our prototype, *AutoPKI*. Section 6 evaluates the performance of our prototype and the effectiveness of our solution. Section 7 provides two example scenarios. Section 8 concludes with some directions for future work.

2. Problem description

The central goal of *Public Key Infrastructure* (PKI) is to enable trust judgments between distributed users. At its core, a PKI depends on certificates: signed bindings of public keys to keyholder properties. Effective use of PKI requires use of these certificates; however, effective use of certificates requires many additional services, such as OCSP servers, CRL repositories, timestamping services, etc. As a consequence, client-side PKI tools need to be able to discover and use these services; server-side PKI tools need to be able to provide these services and enable client tools to discover them.

In order to better understand the limitations of the current practices and to validate the need for PRQP, we analyzed the profile of a population of widely deployed CA certificates.

Our analysis focused on two different set of certificates:

(a) the ones embedded into popular browsers (i.e., Firefox, IE and Konqueror) and Mail User Agents (i.e., Thunderbird, Outlook and K-mail) and
(b) the ones used in the main web pages of universities in USA and Europe.

Figure 2 shows the results coming from the study of the certificate profiles from the first set. Most of the certificates in (a) do not provide any pointers, thus making it really difficult for applications to correctly reach PKI related resources. For instance, in the Firefox/Thunderbird certificate store 66% of certificates have no pointers to any service or data repository (not even to CRLs), while for IE7/Outlook this percentage

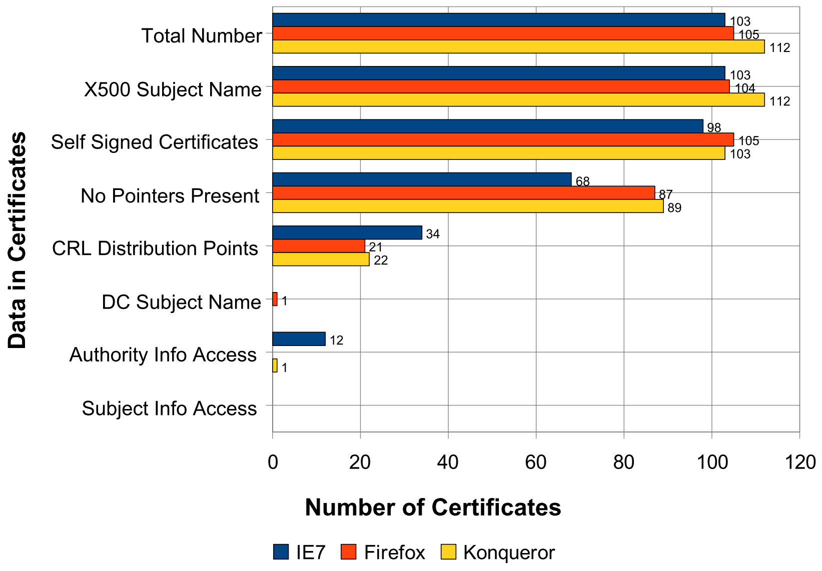


Fig. 2. Profile analysis for certificates embedded in major applications.

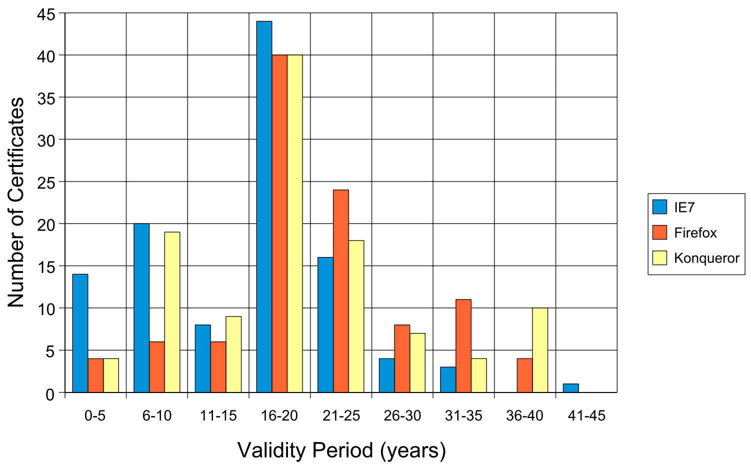


Fig. 3. Validity periods of CA certificates present in today's browsers.

goes up to 82%. This problem is even worse when taking into account the lifetime of the certificates. Figure 3 shows that the majority of the analyzed certificates present a validity period that spans over twenty or more years. Indeed, most certificates have lifetimes far longer than a typical URL – making it risky to solve the resource discovery problem by simply listing the URL in a certificate. The combination of the two analyses suggests that updating the contents of embedded CA certificates could be really difficult. PQRP would solve this problem.

To understand if these results were biased by the requirements imposed by the application policies, we turned our attention to the second set, university certificates. The profiles of the collected certificates are depicted in Fig. 4. By contacting all the university websites [40,42] with the HTTPS protocol – where supported – we were able to dump the list of certificates from the servers. After having retrieved all the certificates, we analyzed the results. From a pool of 2013 US universities, 1016 support HTTPS. The retrieved certificates were primarily issued by organizations external to the university (91.4%). In this scenario, many certificates were pointing to the same providers; only 35 different CAs provide certificates for 929 different universities. Most of the certificates were providing pointers to CRLs and OCSP servers which were, most of the time, the same across different organizations. We think that the university usage of certificates from commercial vendors, even when the university has an internal CA, is due to the lack of real solutions to achieve interoperability between PKIs. (That is, it's easier for relying parties to find your root if it's baked into the browser.)

Results for European universities were quite different. In this case out of 2541 universities, 745 support HTTPS. Differently from the US case, the number of internally[1] issued certificates exceeds the number of certificates from external vendors. We were able to count 414 different providers of which 332 were "internal". In this environment, where there are many different vendors, we discovered that more than 54% of certificates did not provided any pointer to any PKI resources. From this result, it is therefore evident that also for *End Entities* (EE) certificates, such as the

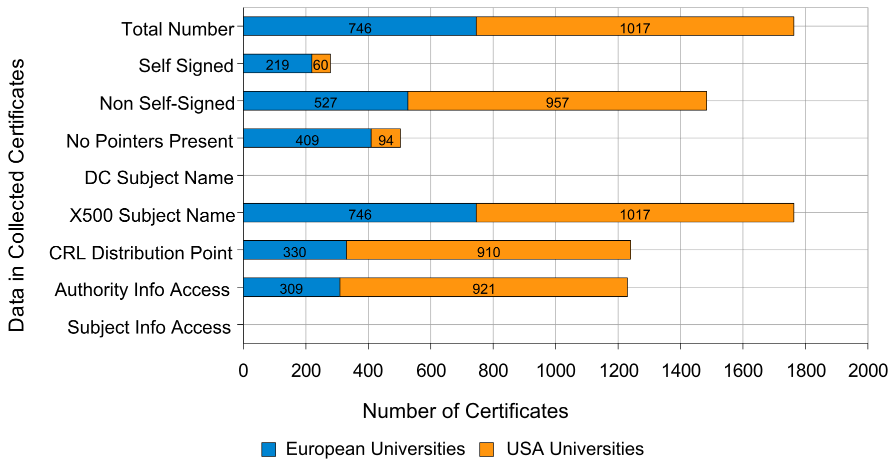


Fig. 4. Profile analysis for certificates of European and USA universities.

[1] Issued by the university's internal CA.

ones from university websites, solving the resource discovery problem by simply listing URLs in the certificate does not provide a working solution. Also in this case PRQP would solve the problem.

3. Related work

Our work focuses on the PKI resources look-up problem. This problem involves not only certificate retieval, but also discovery of new services whenever they are made available by service providers. In the literature, we see three primary methods for clients to obtain pointers to PKI data: adopting specific certificate extensions; looking at easily accessible repositories (e.g., DNS, local database, etc.); and adapting existing protocols (e.g., Web services).

3.1. Certificate extensions

To provide pointers to published data, a CA could use the *Authority Information Access* (AIA) and *Subject Information Access* (SIA) extensions (within X.509 certificates), as detailed in RFC 3280 [19]. The former can provide information about the issuer of the certificate while the latter carries information (inside CA certificates) about offered services. The *Subject Information Access* extension can carry a URI to point to certificate repositories and timestamping services. Hence this extension allows to access services by several different protocols (e.g., `HTTP`, `LDAP` or `SMTP`).

Although encouraged, usage of the AIA and SIA extensions are still not widely deployed. There are two main reasons for this. The first is the lack of support for such extensions in available clients. The second reason is that extensions are static, i.e. not modifiable. Indeed to modify or add new extensions, in order to have users and applications to be aware of new services or their dismissal, the certificate must be re-issued.

This would not be feasible for EE certificates, except during periodic reissuing. This would also not be really feasible for the CA certificate itself. The CA could retain the same public key and name and just add new values to the AIA extension in the new certificate. If users fetch the CA certificate regularly, rather than caching it, this would enable them to become aware of the new services. Unfortunately, almost every client available does not look for CA certificates if they are already stored in the client's local database.

In any case, since URLs tend to change quite often while certificates persist for longer time frames, experience suggests that these extensions will invariably point to URLs that no longer function. Moreover, considering the fact that the entity that issues the certificates and the one who runs the services may not be the same, it is infeasible that the issuing CA will reissue all of its certificate in case a server URL's changes. Therefore, it is not wise to depend on the usage of AIA or SIA extensions for discovery of available services and repositories.

Table 1

Analysis of AIA statistics, for each browser we report the number of certificates that carry a particular type of Authority Information Access pointer

AIA Datatype	Firefox	IE7	Konqueror
OCSP	12	0	1
caIssuers	0	0	0
timeStamping	0	0	0
DVCS	0	0	0

In Table 1 we report the contents of the AIA extensions in some representative applications. As expected, only OCSP pointers are present in a very small number of certificates (i.e., 11% for Firefox/Thunderbird, 0% for IE7/Outlook and Konqueror/K-mail), whilst no pointer to other services are provided.

3.2. DNS service records

The SRV record or *Service record* technique is thought to provide pointers to servers directly in the DNS [25]. As defined in RFC 2782 [14], the introduction of this type of record allows administrators to perform operations rather similar to the ones needed to solve the problem we are addressing in this paper, i.e. an easily configurable PKI discovery service.

The basic idea is to have the client query the DNS for a specific SRV record. For example if an SRV-aware LDAP client wants to discover an LDAP server for a certain domain, it performs a DNS look up for *_ldap._tcp.example.com* (the "_tcp" means the client requesting a TCP enabled LDAP server). The returned record contains information on the priority, the weight, the port and the target for the service in that domain. The priority indicates the order in which the client should contact the available servers. The weight field specifies a relative weight for entries with the same priority and it is used for server selection mechanism as specified in [14].

The problem in the adoption of this mechanism is that in PKIs (unlike DNS) there is usually no fixed requirement for the name space used. Most of the time, there is no correspondence between DNS structure and data contained in the certificates. The only exception is when the *Domain Component* (DC) attributes are used in the certificate's subject.

The DC attributes are used to specify domain components of a DNS name, for example the domain name "example.com" could be represented by using the

$$\texttt{dc} = \texttt{com}, \qquad \texttt{dc} = \texttt{example}.$$

If the CA's subject field would make use of such a format, the content of issued certificates (e.g., the *Issuer* field) would allow client applications to perform DNS lookups for the provided domain where the information about repositories and services could be stored.

However, currently, the practice is very different. In fact it is extremely difficult for a client to map digital certificates to DNS records because the DC format is not widely adopted by existing CAs. As shown by our analysis, only one certificate[2] (out of the 105 in the IE7/Outlook store) uses the domain components to provide a mapping between the certificate and an Internet Domain.

Recently a new proposal has been presented by the IETF PKIX Working Group [2] to standardize the usage of DNS records to locate PKI repositories. It emerged from discussion that, although a client has been implemented that is capable of locating an LDAP service for a specific e-mail address, the authors of the proposal were not able to find anyone who announces their directory service in the DNS according to the specification.

Another example of the infeasibility of this solution is presented in Fig. 5. The figure depicts a very common scenario where an organization "A" buys a certificate for its web server from a CA ran by organization "B". Neither the contents of the distinguished name nor the contents of other fields in the certificate (e.g., `subjectAltName`) provide a pointer to the right domain where the query for RR records should be made.

Moreover, the issuing organization may not even have control over the DNS records in case they need to be updated. In our example, if RR records are put in the DNS under the domain identified in the *Common Name* (CN) attribute of the web server's certificate, i.e. "`my.server`", the management of such records is not under control of the issuing organization ("B").

3.3. Web services

Web services [9] is a new technology using three different components to allow applications to exchange data: *SOAP (Simple Object Access Protocol)* [24], *WSDL (Web Services Description Language)* [5,6] and *UDDI (Universal Description Discovery and Integration)* [7].

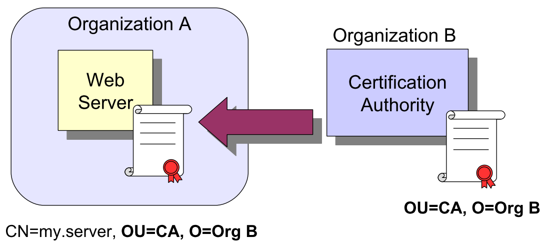


Fig. 5. The Certificate Authority from Organization "B" issues a certificate to the web server from Organization "A".

[2] `/DC=com/DC=microsoft/CN=Microsoft Root Certificate Authority`.

By using UDDI, applications discover available Web Services (described by using the WSDL language) and interact with them by using SOAP to exchange data. Although Web Services provide a good tradeoff between flexibility and complexity (e.g., CORBA [8] offers much more possibilities but CORBA-oriented applications are difficult to implement), the format of exchanged messages is still complex. In fact, communication is handled by using XML [43] which is quite slow when compared to other binary formats like DER [20]. These aspects are to be considered with special care when it comes to mobile devices. XML-formatted messages require a large amount of computational power to be correctly processed and large bandwidth (messages are usually bigger in size). From our experience a message encoded by using the DER format is less than the 30% in size when compared to the corresponding XML format.

Another important aspect to be considered here is the *ease of integration* into existing applications. Every application dealing with digital certificates already have its own implementation for DER, while it is not true that XML is widely supported as well.

3.4. Local network-oriented solutions

Another approach to provide reliable information is to use existing protocols for service location such as *Jini* [1,11], *Universal Plug and Play* protocol (UPnP) [23, 39] or *Service Location Protocol* (SLP) [15–17].

Jini is used to locate and interact with Java-based services. The main disadvantage of Jini is that it is tied to a specific programming language and it requires a lot of Java-specific mechanisms (e.g., object serialization, RMI [22] and code downloading) in order to function properly. In addition it provides many communication services which are quite complex and not really needed in our environment.

Like Jini, UPnP provides a mechanism to locate and to interact with services over a network. UPnP is also very complex as it involves the usage of different techniques like XML (SOAP) over HTTP. The protocol is peer-to-peer and it is aimed for home environments. There exists a service-discovery subset of UPnP, the *Simple Service Discovery Protocol (SSDP)* [13], which operates on HTTP over UDP. As UPnP, the SSDP is thought to be operated in small environments and it is possible that administrators block UPnP from leaving the LAN or disable it for security reasons, in the same way they currently block/disable NetBIOS from leaving local networks.

The IETF defined the SLP to provide a service location mechanism that is language and technology independent. Some issues, however, make it not the right choice to solve our problem. First of all, the protocol is very complex to implement, although a freely available reference implementation [32] exists. Moreover, there is little deployment of SLP and there is little knowledge of its existence.

4. The PKI Resource Query Protocol (PRQP)

After considering the available options, we concluded that the definition of a specific and simple protocol for PKI resources location is needed to ease its integration into existing and future applications, especially for mobile devices which have limited computational power and communication bandwidth.

To solve this problem, we define the *PKI Resource Query Protocol* (PRQP) for finding any available PKI resource related to a particular CA. In PQRP, the client and a *Resource Query Authority* (RQA) exchange a single round of messages (Fig. 6):

(1) the client requests a resource token by sending a request to the server;
(2) the server replies back by sending a response to the requesting entity.

The client embeds zero or more resource identifiers (OIDs) – when specifying exactly the data the client is interested in – in the request token, in order to specify which subset of CA resources she wants. If the client does not specify any services by providing an empty list of OIDs in the request, all of the available data for a particular CA should be returned by the server in the response. The resources might be items that are (occasionally) embedded in certificates today – such as URLs for CRLs or OCSP or SCVP – as well as items such as addresses of the CA homepage address, the subscription service, or the revocation request.

4.1. Resource Query Authority (RQA)

In our protocol, an RQA can play two roles. First, a CA can directly name a given RQA as the party who can answer queries about its certificates, by issuing

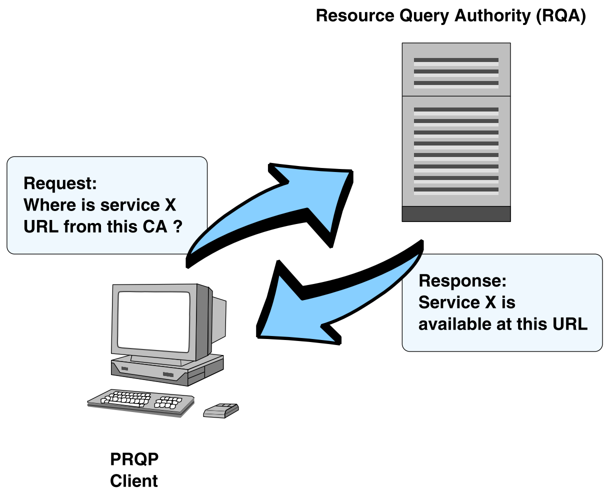

Resource Query Authority (RQA)
Request: Where is service X URL from this CA ?
Response: Service X is available at this URL
PRQP Client

Fig. 6. PRQP processing.

a certificate to the RQA with a unique value set in the *extendedKeyUsage* (i.e., `prqpSigning`). The RQA will provide authoritative responses for requests regarding the CA that issued the RQA certificate. Alternatively, an RQA can act as *Query Trusted Authority* (QTA) ("trusted" in the sense that a client simply chooses to trust the RQA's judgment). In this case, the RQA may provide responses about multiple CAs without the need to have been directly certified by them. In this case, provided responses are referred to as *non-authoritative*, meaning that no explicit trust relationship exists between the RQA and the CA. To operate as a QTA, a specific extension (*prqpTrustedAuthority*) should be present in the RQA's certificate and its value should be set to `TRUE`. In this configuration the RQA may be configured to respond for different CAs which may or may not belong to the same PKI as the RQA's one.

4.2. The message format

A PRQP request contains several elements. The *protocol version* is used to identify whether the client or the RQA is capable of handling the request format. (Currently, `v1` is the only allowable value.) The `NONCE` (optional) is a random number long enough to assure that the client will produce it only once. The `ResourceRequestToken` identifies the resource (e.g., the CA and the service itself). The `MaxResponse` identifier tells the RQA the maximum number of `ResourceResponseToken` that may be present in the response.

The `ResourceRequestToken` contains a CA's target certificate identifier and optionally one or more `ResourceIdentifier` fields. If one or more are provided in the request, the RQA should report back the location for each of the requested services. If no `ResourceIdentifier` is present in the request, the response should carry all the available service locations for the specified CA (with respect to the `MaxResponse` constrain). Extensions can be used for future protocol enhancement.

The *PRQP response* also contains several elements. Again, the *protocol version* identifies the response's version. The `NONCE`, if present, binds the response to a specific request. The usage of the `NONCE` is meaningful only in signed responses and its value must be copied directly from the corresponding request. The status data structure (`PKIStatusInfo`) carries the response status and, in case of error, a description of the cause. The `ResourceResponseToken` is used to provide the pointers to the requested resources (one for each requested service). Optional Extensions may be added if requested.

4.3. Considerations about PRQP

When designing the protocol, we paid special attention to several aspects: simplicity, security, message complexity, and RQA address distribution.

An important target of the protocol design was simplicity. By keeping the protocol very simple, its adoption would not add a big additional burden to PKI management, nor to applications and developers.

Security was another major concern. The PRQP provides URLs to PKI resources, therefore it only provides locators to data and services, not the real data. It still remains the client's job to access the provided URLs to gather the needed data, and validate the data (e.g., via signatures or SSL). Because of this consideration, both the `NONCE` and the signature are optional in order to provide flexibility in how requests and responses are generated. Also, it is then possible to provide pre-computed responses in case the `NONCE` is not provided by the client. If an authenticated secure channel is used at the transport level between the client and the RQA (e.g., HTTPS or SFTP) signatures in requests and responses can be safely omitted. As the RQA would become a very important service of a PKI, special care should be taken in protecting the service against malicious attackers. In particular when considering Denial of service (DoS) attacks, caching servers or IP filtering could be used to mitigate their effectiveness.

We also analyzed the level of complexity of messages. Some type of services, e.g., delta CRLs, can be directly detected upon data downloading. However, if a client is looking for a specific version of a protocol or data type, a fine-grained query system can reduce server load by only permitting data download when the requesting client actually supports that version.

We considered two different candidates for the PRQP message format: *eXtensible Markup Language* (XML) and *Distinguished Encoding Rules* (DER). The adoption of the *Abstract Syntax Notation (ASN.1)* to describe the data structures would let the software developer provide either DER or XML-based implementations of the protocol. However we think that a DER-based implementation of PRQP is the best choice because of compatibility considerations with existing applications and APIs. Moreover, signed DER encoded messages are smaller in size and easier to process then XML encoded ones [26] and almost all PKI aware applications already support it.

Last but not least considered issue was the distribution of the RQA's address. We envisage two different approaches. A first option would be to use the AIA and SIA extensions to provide pointers to RQAs. Although this approach seems to be in contrast with considerations provided in Section 3.1, we believe that by using only one extension to locate the RQA would provide an easy way to distribute the RQA's URL. The size of issued certificates would be smaller, thus providing a more space efficient solution. A second option is applicable mostly in LANs and consists in providing the RQA's address by means of DHCP. This method would be mostly used when a trusted RQA is locally available. These two techniques can then be combined together.

4.4. Extending querying capabilities

PRQP was designed in order to enhance interoperability between PKIs. Thanks to the extensibility of the PRQP format, the definition of a common set of OIDs is what was required to provide grid specific pointers. Table 2 summarizes the results.

Table 2

Newly identified OIDs for PKI operations

	OID	Text	Description
PKIX[3]	id-ad 1	ocsp	OCSP service
	id-ad 2	caIssuers	CA information
	id-ad 3	timeStamping	Time stamping service
	id-ad 10	dvcs	DVCS service
	id-ad 11	scvp	SCVP service
New Defined OIDs	id-ad 50	certPolicy	Certificate Policy (CP) URL
	id-ad 51	certPracticesStatement	Certification Practices Statement (CPS) URL
	id-ad 60	httpRevokeCertificate	HTTP based (browsers) certificate revocation service
	id-ad 61	httpRequestCertificate	HTTP based (browsers) certificate request service
	id-ad 61	httpRenewCertificate	HTTP based (browsers) certificate renewal service
	id-ad 62	httpSuspendCertificate	Certificate suspension service
	id-ad 40	cmsGateway	CMS gateway
	id-ad 41	scepGateway	SCEP gateway
	id-ad 42	xkmsGateway	XKMS gateway
	eng-ltd 3344810 10 2	webdavCert	Webdav certificate validation service
	eng-ltd 3344810 10 3	webdavRev	Webdav certificate revocation service

Some of the newly proposed pointers (OIDs) deserve further description. Today several standardized protocols can be used to interact with a CA. In order to allow users and applications to discover these *Points of Interactions*, we provide a set of pointers that specify these "Communication Gateways". In particular we identified three protocols that are mature enough to be deployed in applications. The first one is the Certificate Management over CMS (CMC) Transport Protocol [37] by IETF. The second one, the Simple Certificate Status Protocol (SCEP) [28] was designed by CISCO. Although this protocol is not officially a IETF standard,[3] it is one of the most widely supported protocols both by commercial and open source software. We also defined a pointer for the XML Key Management System (XKMS) [36], which was defined by W3C. Although the XKMS specifications have been available for quite a while now, not many software implementations support it yet.

We also identified two pointers that may be used for Certificate Policy (certPolicy) and Certification Practices Statement (certPracticesStatement) specification. CP and CPS provide the relying parties with the information upon which the trust in the certificate is built. By providing the pointers to this information in a standardized way, automatic auditing tools could be used to download this information and verify it is made publicly available.

As PKIs change and new protocols are defined, PRQP can provide the CA management with a dynamic model capable of providing information about new services

[3]During the 70th IETF Meeting the PKIX Working Group decided to dismiss the proposal to move SCEP on standard track.

or, if needed, to allow clients to switch to newer and more efficient ones. In the IETF PKIX WG, a new use of the WebDav protocol has been proposed in order to provide an efficient certificate publication and revocation [4] mechanism. To support this proposal we identified the WebDav Certificate Validation Service (webdavCert) and the WebDav Certificate Revocation Service (webdavRev) pointers. A simple change in the configuration of the RQA enables a CA to specify the URLs where access to the new services (e.g., the WebDav revocation information) is located.

5. Prototype design and implementation

To bring our solution into practice, we built *AutoPKI*, a prototype implementation of the libraries and software support to carry out PQRP in real PKI applications. The basic idea behind AutoPKI is to automatically provide clients with addresses of PKI resources and to ease administrators and users from PKI configuration issues.

Our system differs from previous work in that it is aimed to provide an easy-to-use and simple-to-deploy location service *without* the complexity of providing third party validation or proxying services (e.g., does not provide services as SCVP [12]). The availability of PKI resource addresses enables applications to automatically look for different type of data:

- *Certification Services* – where the client could automatically submit a certification request to get a new certificate or where to submit a revocation request for an already issued certificate.
- *Data Repositories* – where to look for certificates and revocation data (e.g., an LDAP server or an FTP repository).
- *Extra Services* – URLs of additional services, this would include all services currently available from a CA, e.g., OCSP, SCVP, TimeStamping, etc.

The AutoPKI system is mostly built on the integration of the PRQP into PKI applications and it makes use of the three principal components (Fig. 7): an Extended DHCP client and server, a Resource Query Authority server, and a PRQP library. We describe each of them separately.

5.1. The extended DHCP client and server

To bring PRQP into the real world, we need to distribute the addresses of available RQAs. A naive option is to include the AIA and SIA extensions to carry the pointer to the RQA directly in digital certificates. This approach would actually work only if the CA of the target certificate provides an RQA. The extension contents would point to the available RQA and the client could directly discover services provided by the CA by querying the RQA.

However, we cannot rely on the presence of RQA pointers in certificates, since they are not generally present in today's infrastructure. Therefore we needed a way

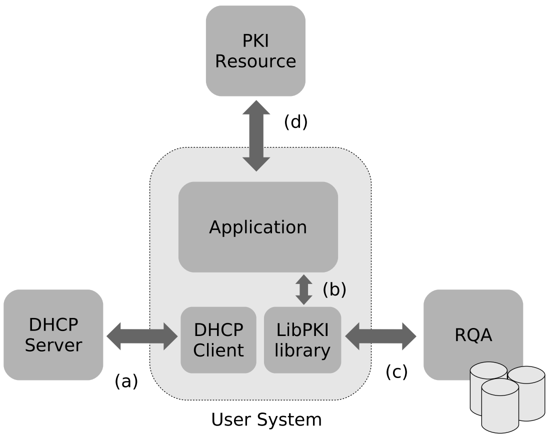


Fig. 7. General Design of AutoPKI, our PRQP prototype. The address of the RQA server is retrieved via (a) an enhanced DHCP server/client when the user system is started. When a PKI resource is to be found, the application communicates to the (b) RQA server by using the functions provided by (c) libPKI. The application can then (d) retrieve the needed data by contacting the discovered PKI resource URL.

```
# generated by /sbin/dhclient-script
queryauthority 130.192.1.23
queryauthority 130.192.1.59
```

Fig. 8. Example configuration file originated by the extended DHCP client (dhclient).

to provide clients with a pointer to a *local RQA* to query for resources provided by CAs that do not have RQAs. In fact if no RQA address is present in the certificate, a client application could use a default configured one.

The DHCP protocol provides sufficient flexibility for this purpose. In particular it allows the client to request the server to send specific information if needed. By modifying the configuration (to add specific options both to the client and the server) it is possible to store the provided addresses in a system-wide configuration file where applications could retrieve the local RQA address. Figure 8 reports an example configuration file for PRQP.[4] In case no DHCP server is available, configuration can be provided by using a simple user interface, also common practice for DNS configuration on many systems.

5.2. Integrating PRQP into applications

To easily provide support for applications that want to make use of PRQP, we implemented a *PRQP library*. It can be invoked by applications in order to discover

[4]Our implementation stores the file as `/etc/pki.conf`.

the address of a repository or a service. The implemented interface provides applications with easy-to-use functions that handle both the generation and parsing of requests/responses as well as communication with the designated RQA. To further facilitate interoperability, we decided to integrate the developed code into LibPKI [29], which enables developers with the possibility to implement complex cryptographic and certificate management operations with a few simple function calls by implementing an high-level cryptographic API. At its lower layer, LibPKI makes use of OpenSSL [33] and can support different cryptographic providers, e.g., KMF [38].

Our PRQP library uses the configuration file generated by the DHCP client in order to automatically retrieve the address of the RQA. Besides the low-level functionality needed to manage the PRQP data structures, we also implemented several high-level ones that help developers to integrate PRQP in their applications. In particular we provided the following functions described in Table 3.

Along with the library, we built a *command line tool* that accepts an X.509 certificate and configuration options (e.g., names of requested resources) as input, and outputs the response both in PEM/DER and in a human-readable format. The output could then be parsed by any calling application in order to use the response's data.

When the command line tool is executed, it performs the following steps:

(1) verifies the user input and load the certificate(s) whose services and/or data are requested;
(2) builds up the PRQP request;
(3) parses the global PKI configuration file;
(4) connects to the configured RQA server via TCP sockets;
(5) sends the request to the server by using the HTTP protocol (in particular we use the POST method to upload the request to the server);
(6) retrieves and parse the RQA response;
(7) eventually saves the request and the response in separate files; and
(8) prints out the response details in text format.

Table 3
Implemented PRQP API in LibPKI

Function	Description
`PRQP_REQ *PRQP_REQ_new_url(...)`	Generates a new PRQP request
`int PRQP_REQ_add_service_stack(...)`	Adds the stack of requested services to the request
`PRQP_RESP *PKI_get_PRQP_RESP_url(...)`	Sends the request and retrieves the response to/from the server
`int PRQP_REQ_sign_tk(...)`	Signs the PRQP request
`int PRQP_RESP_sign_tk(...)`	Signs the PRQP response
`int PRQP_REQ_verify(...)`	Verifies the signature on the PRQP request
`int PRQP_RESP_verify(...)`	Verifies the signature on the PRQP response

By using the client library, steps from two throughout six can be performed automatically. For this purpose we provided the library with the `getpkiresources` function which handles all the PRQP details and returns a stack of `URL` structures back to the calling application. The address of the RQA is directly extracted from a global PKI configuration file located in `/etc/pki.conf` which is generated by the extended DHCP client. The application is not required, at any point, to have any specific knowledge about the PRQP protocol as all the messaging with the RQA server is handled within our PRQP library.

5.3. RQA server

Because of many similarities between PRQP and OCSP in the basic design we decided to implement our PRQP responder by using the OpenCA [31] OCSPD [30] package. This software uses OpenSSL and implements an OCSP responder over HTTP. To implement PRQP, we modified the software by leveraging the functionality provided by the LibPKI library: ASN.1 functions capable of loading, parse and save PRQP data structures by using the I/O abstraction layer; request and response processing functions; network communication functions to manage the simple HTTP POST method used between the client and the RQA.

Because of the simple design of OpenCA's OCSP responder, we could reuse much part of the original code in order to build PRQP responses instead of OCSP ones. Currently we support PRQP over HTTP only. We also defined the "`application/prqp−request`" and "`application/prqp−response`" HTTP content types for PRQP requests and responses, respectively.

Our server is capable of acting also as a TA by supporting multiple CAs by setting the appropriate configuration options. Each configured CA and its provided services have been grouped together in separated sections of the configuration file, thus being very easy to add new CAs to the server.

6. Evaluation

To test the system, we set up a testbed consisting of two computers connected over a switched Fast Ethernet LAN. On the first machine (Intel Core Duo @ 2.13 GHz, 4 GB RAM) we installed the PRQP library and the PRQP server, while on the second one (Intel Pentium M @ 600 MHz, 512 MB RAM) we installed the PRQP library and the command line tool. Both systems were running Linux 2.6.18.3 Kernels on a Fedora Core 6 distribution. On the RQA server, we configured the pointers to services provided by our CA. Each response was digitally signed by using the RSA algorithm and 1024 bit keys; no crypto hardware was used.

On the client, we ran several tests that made use of the command line application to query the RQA server; in particular, we queried the server with an increasing number of requested pointers and repeated the experiment fifty times each. Although

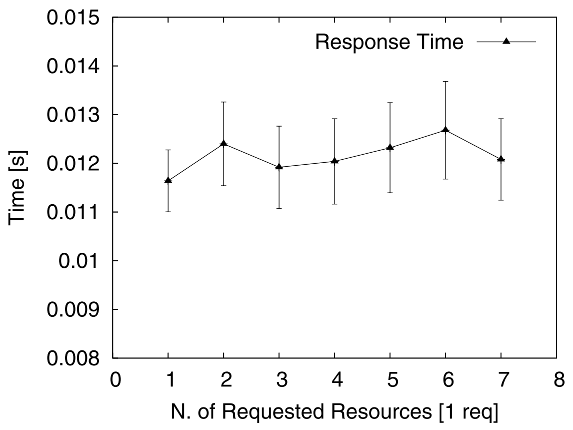


Fig. 9. PRQP response times.

the PRQP enables for caching of responses during their validity period, no caching of responses has been used during our tests. Response times are reported in Fig. 9.

The results show that the overhead introduced by our system is small – and is almost negligible for the majority of today's applications. Moreover, no increase in response time has been noticed with the number of requested locators.

We also analyzed the size of PRQP requests and responses. Collected data are shown in Fig. 10. Generated requests are considerably smaller in size in respect to responses. The main reason for this is that we decided not to sign requests and not to include any certificate in the request (because we envisage this would be the most common scenario), while the server is signing and including its own certificate into generated responses.

We also noticed that the size of the responses grows more rapidly than the size of the requests. This is easily explained by the fact that in the request a single OID is used to identify the service, whilst in the response a more complex data structure is used that comprises the actual locators and the validity period for that information.

Enhancing PRQP response caching. In the first version of our PRQP protocol proposal [34], only PRQP requests carry an identifier for the CA. This identifier is used by the RQA to identify the CA whose pointers are requested by the client. Although efficient, the client would not be able to identify the CA that the response refers to by simply looking at the response, thus caching across different applications on the same computer would be infeasible.

In order to allow for client caching of responses, an additional field is required in the response. We added a CA identifier in the PRQP response message. This identifier allows the client to tie the information received from an RQA to a CA without the need to cache the sent request as well. The new CA identifier structure definition is shown in Fig. 11.

By adding the new data structure, we introduced a small overhead in terms of response size (from 160 to 330 bytes); nevertheless this simple change simplifies

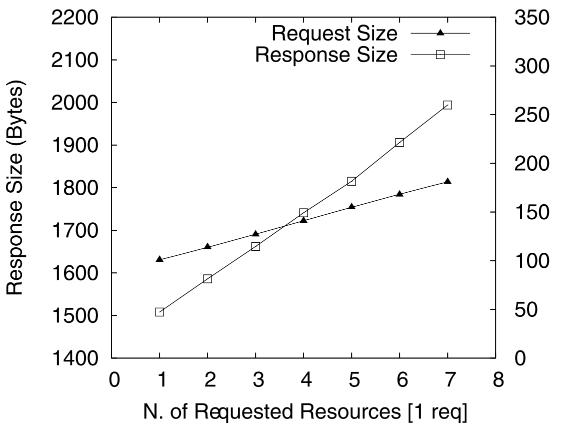


Fig. 10. PRQP message sizes.

```
BasicCertIndentifier ::= SEQUENCE {
    subjectNameHash    OCTET STRING,
    issuerNameHash     [0] OCTET STRING   OPTIONAL,
    serialNumber       [1] SerialNumber   OPTIONAL  }
```

Fig. 11. ASN1 description of the CA identifier added to the response message in our modified version of PRQP.

response caching on PRQP enabled clients. Moreover, because the CA identifier does not change, its contents can be pre-computed, thus it does not add any significant computational burden on the server.

7. Solving the problem: Two examples

In this section we provide two different examples. The first one presents a scenario in which authentication software wants to discover where the OCSP server (if it exists) for a particular CA is located. The second example describes a possible usage of PRQP in order to discover the address of a revocation service from the certificate's own CA.

7.1. Discovering OCSP

Consider a scenario where many large organizations of a particular type have deployed their own PKIs within their own user populations, but wish to provide services to users from peer organizations. (This is already happening: in US universities, in pharmaceutical corporations, in branches of the US Federal Government, for a few examples.) In such scenarios, the authentication layer of the authentication framework of an organization needs to discover services provided by external CAs in order to get the latest information about the revocation status of a given supplicant certificate.

At first, the authentication server receives the client's certificate from the application – e.g., a web server or a custom application. In case the needed data is not already cached by the authentication software, the authentication server can build a PRQP request for the information by processing the certificate data (i.e., by looking at the issuer identifier in the certificate). In this example, the authentication layer asks the RQA for the location of the OCSP server for the CA that issued the client's certificate. This step might also be useful when the OCSP pointer is *not* provided in the certificate (e.g., the OCSP service was activated after the particular certificate was issued). This provides the server with the freshest information in case the CA started, moved or dismissed the OCSP service.

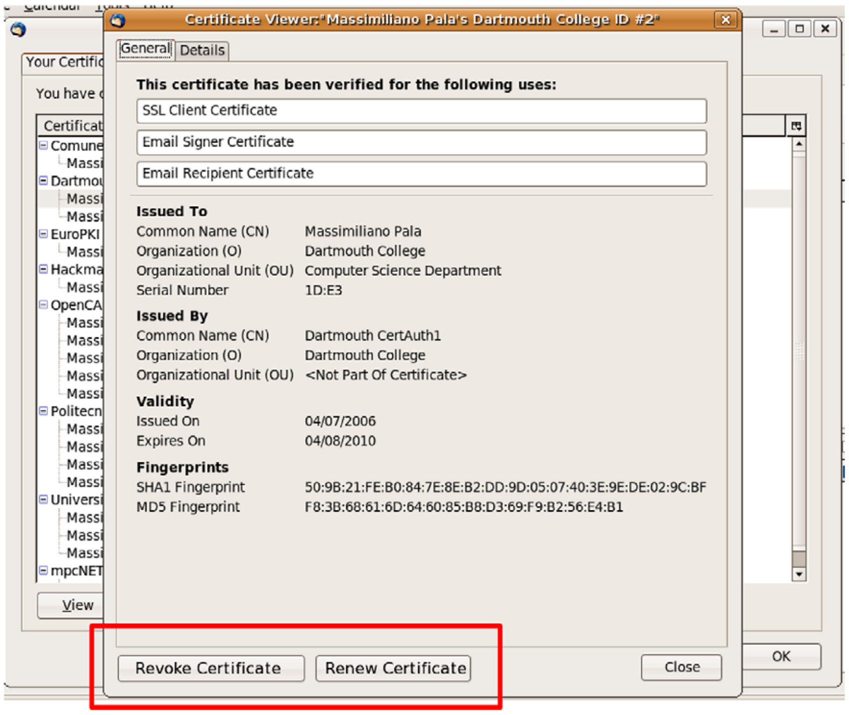
The RQA provides the authentication server with the URL of the requested service which has been configured on the RQA. In this particular example only the OCSP URL is requested, and therefore only the locator for this service is put in the response. When the authentication server has all the needed data, it continues with the normal validation procedures by using the provided URL to directly access the OCSP server.

7.2. Asking for revocation

Accidents happen. As PKI permeates broader user populations, the more likely it will be that some users will encounter scenarios – such as discovering that an external adversary has "owned" the user machine – which require revoking the user's certificate.

However, requesting certificate revocation is typically not an easy task. In many cases, the URL to request the revocation of a certificate is only distributed to the user by means of an email sent at the time the certificate was initially issued; that e-mail may not be easily accessible when the subscriber has the greatest need for it. By using PRQP, an application could present a simple UI where an "Ask for Revocation" button is presented. An example is shown in Fig. 12. By requesting the URL of the revocation service from the RQA, the application could automatically access the service and ask for the revocation of the certificate. Such an automated process would only be possible if the subscriber still had control of their private key, as it is anticipated that any revocation request not signed by the subject's private key would need to be handled via an out of band process involving an RA of the particular PKI. This approach potentially eases key management issues for the subscriber, thus enhancing the certificate usage experience for the user.

Another scenario where PRQP could enhance the usability of digital certificates is the renewal process. A renewal URL could be provided so that the process of renewing a certificate could be completely automated. Also in this case the application could provide the user with a simple UI where an "Ask for Renewal" button is presented. By using PRQP, the application could automatically contact the CA (or


Certificate Viewer:"Massimiliano Pala's Dartmouth College ID #2"
General
Details

This certificate has been verified for the following uses:

SSL Client Certificate
Email Signer Certificate
Email Recipient Certificate

Issued To	
Common Name (CN)	Massimiliano Pala
Organization (O)	Dartmouth College
Organizational Unit (OU)	Computer Science Department
Serial Number	1D:E3
Issued By	
Common Name (CN)	Dartmouth CertAuth1
Organization (O)	Dartmouth College
Organizational Unit (OU)	<Not Part Of Certificate>
Validity	
Issued On	04/07/2006
Expires On	04/08/2010
Fingerprints	
SHA1 Fingerprint	50:9B:21:FE:B0:84:7E:8E:B2:DD:9D:05:07:40:3E:9E:DE:02:9C:BF
MD5 Fingerprint	F8:3B:68:61:6D:64:60:85:B8:D3:69:F9:B2:56:E4:B1

Fig. 12. Example of a simple User Interface where the user is presented with simple "Revoke Certificate" and "Renew Certificate" buttons.

Registration Authority) application and send the renewal request without the user to be required to remember the list of URLs for all of his certificates.

It is to be noted that the more the CAs will support PRQP, the more the user can be relieved from difficulties in using digital certificates. We think that this aspect could enable the user to actually use many certificates, each one for a different purpose, as the management costs (in terms of time) required would be negligible.

In this way PRQP facilitates greater flexibility for PKI-enabled applications.

8. Conclusions

The lack of interoperability among closed PKI islands is a very urgent problem and demands a solution. The current infrastructure does not solve it. By providing a PKI-specific protocol for resource discovery, our PRQP protocol can provide this solution. We offer a design and prototype – and are working on wider deployment via standards efforts and integration with Open Source tools.

One example of an environment where our system could provide measurable improvements is the Grid community [35], which already makes heavy use of X.509.

One of the most sensitive technical issue to be solved is related to the availability of revocation data and validation services in big Grids. The *Grid Security Infrastructure* (GSI) uses proxy certificates to allow an entity to temporary delegate its rights to remote processes or resources on the Internet. Such a certificate is derived from, and signed by, a normal EE certificate. Therefore an easy way to find validation services and CRLs for EE certificates is needed in order to verify their validity. Administrators decide a set of CAs, and therefore users, to be trusted for accessing the shared resources.

PRQP could help automatic configuration of validation services by providing updated URLs to OCSP, CRLs repositories, or other services (e.g., SCVP). This would increase data availability and possibility to securely use existing PKIs for Grid Computing. Moreover, a party like the International Grid Trust Federation (IGTF) [21], established in October 2005, could run a centralized RQA to provide URLs about federated CAs to all users and resource managers.

Wireless is another very interesting scenario for the deployment of PRQP. Usage of digital certificates in open environments (e.g., university and enterprise WLANs) is strongly limited by interoperability issues. Access Points (or RADIUS servers) could leverage the use of PRQP to discover services and, then, retrieve PKI data needed for validation of client certificates. For example support for visiting students or professors to access the University's network could be easily managed without requiring complex authentication infrastructures (e.g., EduRoam [10]) and without delegating credentials validation to third parties.

The PRQP protocol also offers a starting point for the development of a PKI Resource Discovery Architecture where different RQAs cooperate to access data which is not locally available. Our research will next proceed by evaluating the usage of an authenticated *Peer-To-Peer* (P2P) network for distribution of URLs of available services between RQAs. These authorities would share data about configured services with other peers in the P2P network. In this scenario, each client would use one of the configured RQAs as an entry point where all its requests will be sent to. Thus the P2P network would map network addresses to services mostly like the DNS maps logical names to IP addresses. Current research is focused both on the study and the implementation of such a network.

Acknowledgments

The authors would like to thank Stephen Kent, Frank Pooth, Ashad Noor, Sravan Maringanti, the anonymous reviewers and all the PKIX WG for several discussions and comments. This work was supported in part by the NSF (under Grant CNS-0448499), the US Department of Homeland Security (under Grant Award No. 2006-CS-001-000001) and Sun Microsystems. 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.

References

[1] K. Arnold (ed.), *The Jini Specification*, 2nd edn, Addison-Wesley, Boston, MA, 2000.

[2] S. Boeyen and P. Hallam-Baker, Internet X.509 public key infrastructure repository locator service, IETF Experimental, September 2005, available at: http://tools.ietf.org/wg/pkix/draft-ietf-pkix-pkixrep/draft-ietf-pkix-pkixrep-04.txt.

[3] D. Brink, PKI and financial return on investment, OASIS White Paper, August 2002, available at: http://www.oasis-pki.org/pdfs/Financial_Return_on_Investment.pdf.

[4] D.W. Chadwick, Use of WebDAV for certificate publishing and revocation, Internet Draft, June 2007, available at: http://www.ietf.org/internet-drafts/draft-chadwick-webdav-00.txt.

[5] R. Chinnici, M. Gudgin, J.-J. Moreau and S. Weerawarana, Web Services Description Language (WSDL), Version 2.0, part 1: Core Language, W3C working, May 2005, available at: http://www.w3.org/TR/wsdl20.

[6] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, PWeb Services Description Language (WSDL) 1.1, W3C Note, March 2001, available at: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[7] L. Clement, A. Hately, C. von Riegen and T. Rogers, UDDI version 3.0.2, October 2004, available at: http://uddi.org/pubs/uddi_v3.htm.

[8] Common object request broker architecture: Core specification, March 2004, available at: http://www.omg.org/technology/documents/corba_spec_catalog.htm.

[9] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing* **6**(2) (2002), 86–93, available at: http://dx.doi.org/10.1109/4236.991449.

[10] Education roaming (Eduroam) homepage, available at: http://www.eduroam.org/.

[11] W.K. Edwards, *Core Jini*, 2nd edn, Prentice-Hall, 2000.

[12] T. Freeman, R. Housley, A. Malpani, D. Cooper and W. Polk, Server-based Certificate Validation Protocol (SCVP), Internet Draft, January 2007, available at: http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-31.txt.

[13] Y. Goland, T. Cai, P. Leach, Y. Gu and S. Albright, Simple service discovery protocol, Internet Draft, October 1999, available at: http://www.ietf.org/internet-drafts/draft-cai-ssdp-v1-03.txt.

[14] A. Gulbrandsen, P. Vixie and L. Esibov, A DNS RR for Specifying the Location of Services (DNS SRV), Internet Engineering Task Force: RFC 2782, February 2000.

[15] E. Guttman, Service location protocol: Automatic discovery of IP network services, *IEEE Internet Computing* **3**(4) (1999), 71–80.

[16] E. Guttman, C. Perkins and J. Kempf, Service templates and schemes, Internet Engineering Task Force: RFC 2609, June 1999.

[17] E. Guttman, C. Perkins, J. Veizades and M. Day, Service location protocol, Version 2, Internet Engineering Task Force: RFC 2608, June 1999.

[18] S. Hanna, Follow-up survey on obstacles to PKI deployment and usage, October 2003, available at: http://www.oasis-open.org/committees/pki/pkiobstaclesaugust2003surveyreport.pdf.

[19] R. Housley, W. Polk, W. Ford and D. Solo, Certificate and Certificate Revocation List (CRL) profile, Internet Engineering Task Force: RFC 3280, 2002.

[20] Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T recommendation X.690 – ISO/Uniform Resource Locators (URL) IEC 8825-1:1995, 1994.

[21] International Grid Trust Federation, available at: http://www.gridpma.org.

[22] Java RMI specification, 2003, available at: http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html.

[23] M. Jeronimo and J. Weast, *UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play*, Intel Press, Hillsboro, OR, 2003.

[24] G. Martin, H. Marc, M. Noah, M. Jean-Jacques and N. Henrik Frystyk, SOAP version 1.2, W3C recommendation, June 2003, available at: http://www.w3.org/TR/.

[25] P. Mockapetris, Domain names – implementation and specification, Internet Engineering Task Force: RFC 1035, Request for comments, November 1987.

[26] D. Mundy and D. Chadwick, An XML alternative for performance and security: ASN.1, *IEEE IT Professional* **6**(1) (2004), 30–36, available at: http://www.cs.kent.ac.uk/pubs/2004/2102.

[27] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, Online certificate status protocol – OCSP, Internet Engineering Task Force: RFC 2560, June 1999.

[28] A. Nourse, C. Madson, D. McGrew and X. Liu, CISCO systems Simple Certificate Enrollment Protocol (SCEP), IETF Draft, December 2007, available at: http://www.ietf.org/internet-drafts/draft-nourse-scep-16.txt.

[29] OpenCA LibPKI, available at: https://www.openca.org/projects/libpki/.

[30] OpenCA OCSPD, available at: https://www.openca.org/projects/ocspd/.

[31] OpenCA project homepage, available at: https://www.openca.org/.

[32] OpenSLP project, available at: http://www.openspl.org.

[33] OpenSSL homepage, available at: http://www.openssl.org/.

[34] M. Pala, The PKI resource query protocol (PRQP), Internet Draft, December 2008, available at: http://www.ietf.org/internet-drafts/draft-ietf-pkix-prqp-02.txt.

[35] M. Pala, S. Cholia, S.A. Rea and S.W. Smith, Extending PKI interoperability in computational grids, in: *CCGRID'08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, IEEE Computer Society, Washington, DC, USA, 2008, pp. 645–650.

[36] G.A. Rey, S. Farrell, J. Kahan, B. Lautenbach, T. Lindberg, R. Lockhart, V. Motukuru, S. Mysore, R. Salz and Y. Zhang, XML key management specification (XKMS 2.0), W3C recommendation, June 2005, available at: http://www.w3.org/TR/2001/NOTE-xkms-20010330/.

[37] J. Schaad and M. Myers, Certificate management over CMS (CMC) transport protocols, Internet Draft, November 2007, available at: http://www.ietf.org/internet-drafts/draft-ietf-pkix-cmc-trans-06.txt.

[38] Sun's key management framework, available at: https://www.opensolaris.org/os/project/kmf/.

[39] Universal plug and play specifications, available at: http://www.upnp.org/resources/specifications.asp.

[40] Universities worldwide, available at: http://univ.cc/.

[41] M. Wahl, T. Howes and S. Kille, Lightweight directory access protocol (v3), Internet Engineering Task Force: RFC 2251, December 1997.

[42] World list of universities, available at: http://www.unesco.org/iau/.

[43] F. Yergeau, J. Cowan, T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler, Extensible Markup Language (XML) 1.1, W3C recommendation, February 2004, available at: http://www.omg.org/technology/documents/corba_spec_catalog.htm.