AutoPKI: a PKI Resources Discovery System[⋆]

Massimiliano Pala and Sean W. Smith

Dartmouth College, Computer Science Department,
6211 Sudikoff, Hanover, NH 03755, US
{pala,sws}@cs.dartmouth.edu
http://www.cs.dartmouth.edu

Abstract. The central goal of *Public Key Infrastructure (PKI)* is to enable trust judgments between distributed users. Although *certificates* play a central role in making such judgments, a PKI's users need more than just knowledge of certificates. Minimally, a relying party must able to locate critical parameters such the certificate repositories and certificate validation servers relevant to the trust path under consideration. Users in other scenarios may require other resources and services.
Surprisingly, locating these resources and services remains a largely unsolved problem in real-world X.509 PKI deployment. In this paper, we present the design and prototype of a new and flexible solution for automatic discovery of the services and data repositories are available from a *Certificate Service Provider* (CSP). This contribution will take real-world PKI one step closer to achieving its goal.

Key words: PKI, Service Discovery, Certification Authority, Digital Certificates

1 Introduction

The central goal of *Public Key Infrastructure (PKI)* is to enable trust judgments between distributed users. At is core, 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.

Unfortunately configuring these tools to carry out these tasks is painful for both server administrators and end users, thanks to badly written User Interfaces

[⋆] The authors would like to thank Stephen Kent, Frank Pooth, Ashad Noor, Sravan and all the PKIX WG for several discussions and comments. This work was supported in part by the NSF (under grant CNS-0448499), the U.S. Department of Homeland Security (under Grant Award Number 2006-CS-001-000001), and Sun. 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.

(UI) and overly detailed configurations. Certification Authorities barely publish access details on their official websites; even data as basic as the URLs for provided services and repositories are usually omitted. As a result, if a CA provides a new service (e.g. OCSP [1]) or a new data repository (e.g. LDAP [2]), users and administrators have difficulty learning of these changes. Furthermore, certificates already issued could not carry any sign of the new services. It is unlikely that users (and applications) will be easily aware of the new services if not directly contacted. This problem impacts even more on users from enterprises other than the issuing organization, as they have very limited knowledge about CA's practices and service locations.

In this paper, we present a new approach to provide a flexible way to automatically discover which services and data repositories are available from a CA. This flexibility would also facilitate interoperability across different infrastructures. Section 2 presents the core aspect of our solution: the design and the implementation of a new (and simple) *PKI Resource Query Protocol* (PRQP) easing PKI management both for administrators and final users. Section 3 presents our prototypes. Section 4 evaluates the performance of our prototypes and the effectiveness of our solutions. Section 5 reviews other approaches to solving the problem. Section 6 concludes with some directions for future work.

2 The PKI Resource Query Protocol

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

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 into—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.

Fig. 1 shows an example of this protocol: an SSL web server needs to retrieve the revocation status of a user's certificate. (Here, the Web server is the PQRP requesting client.) At first (step 1), the web server receives the user's certificate from the browser. The web server looks at the issuer identifier in the certificate and builds up a PRQP request asking the RQA for the location of the OCSP server of the issuing CA (step 2). The RQA provides (step 3) the web server with the URL of the requested service, as configured on the RQA. In this particular example only the OCSP URL is requested, and therefore only the locator for such service is put in the response. The web server, then, continues with the

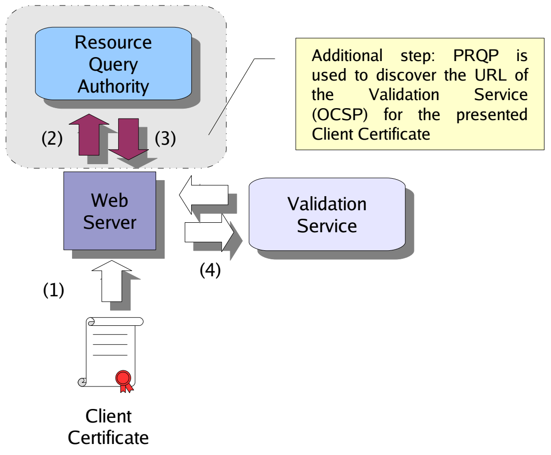

Resource Query Authority
(2)
(3)
Additional step: PRQP is used to discover the URL of the Validation Service (OCSP) for the presented Client Certificate
Web Server
Validation Service
(4)
(1)
Client Certificate

Fig. 1. A web server uses PRQP to help verify a user certificate.

normal validation procedures (step 4) by using the provided URL to directly access the OCSP server.

2.1 Resource Query Authority (RQA)

In our protocol, an RQA can play two roles. First, a CA can directly delegate an RQA as the party who can answer queries about its certificates, by issuing 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 *Trusted Authority* (TA) ("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 TA, 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.

2.2 The Message Format

A PRQP request contains several elements. The *protocol version* is used to identify whether the client or the RQA is capable to handle 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.

Discussion. 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 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.

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 to 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

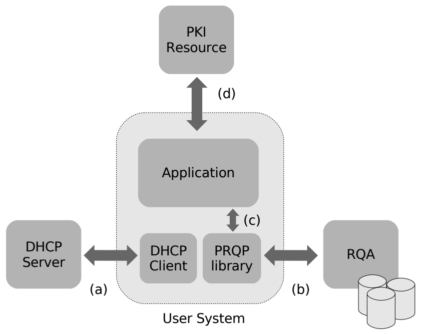


Fig. 2. AutoPKI General Design.

applications and APIs. Moreover DER encoded messages are smaller in size then XML encoded ones 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 5.1, 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.

3 Prototype

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 provide clients with addresses of PKI resources and to ease administrators and users from PKI configuration issues.

Our system differs from previously presented work in that it is aimed to provide an easy to use location service without providing 3^{rd} party validation or proxying services (e.g. does not provide services as SCVP [3]).

Our AutoPKI prototypes makes use of the three principal components (Fig. 2): an Extended DHCP client and server, a Resource Query Authority server, and a PRQP library.

3.1 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

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

Fig. 3. Example configuration file originated by the extended DHCP client (dhclient).

pointer to the RQA directly in digital certificates. This approach works 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, today we could not rely on the presence of RQA pointers in certificates, yet. Therefore we needed a way 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. Fig. 3 reports an example configuration file for PRQP[1]. 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.

3.2 PRQP Library

Our *PRQP library* can be invoked by applications in order to discover the address of a repository or a service. The implemented library 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. The library makes use of OpenSSL [4] for cryptographic operations such as signature generation and verification.

The library uses the configuration file generated by the DHCP client in order to 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.

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

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

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; (8) prints out the response details in text format;
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. As the address of the RQA is directly taken by a global pki configuration file located in `/etc/pki.conf` which is generated by the extended DHCP client, no specific knowledge about the PRQP protocol is required out of the application.

3.3 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 [5] OCSPD [6] 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 PRQP library: ASN.1 functions capable to load, parse and save PRQP data structures by using the I/O abstraction layer of OpenSSL (i.e. the `BIO` interface); request and response processing functions; network communication functions to manage the simple HTTP POST method used between the client and the RQA.

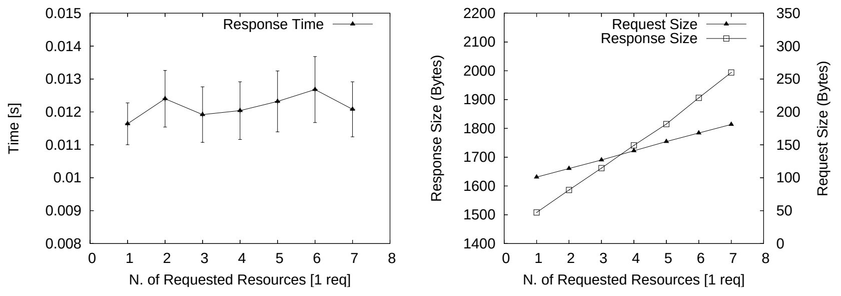
Because of the simple design of OpenCA's OCSP responder, we could reuse much part of the original code in order to build PRQP responses instead of OCSP ones. Currently we support PRQP over HTTP only. We also defined the "`application/prqp−request`" and "`application/prqp−response`" HTTP content types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

Our server is capable to act 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.

4 Evaluation

4.1 Performance

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, 4GB Ram) we installed the PRQP library and the PRQP server, while on the second one (Intel Pentium M @ 600MHz, 512MB 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



(a) *PRQP Response times.* **(b)** *PRQP Message sizes.*

Fig. 4. PRQP Performance Stats

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 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. 4/a.

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. 4/b. 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 was 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.

4.2 Solving the Problem

To demonstrate that PRQP solves the resource discovery problem, we analyzed the profile of a population of widely deployed CA certificates.

Our analysis has been focused on two different set of certificates, the ones embedded into popular browsers (i.e. Firefox, IE and Konqueror) and Mail User Agents (i.e., Thunderbird, Outlook and KMail) and the ones used in the main

	Firefox and Thunderbird	**IE7 and Outlook**	**Konqueror and KMail**
Total Certs	103	105	112
Self Signed Certs	98	105	103
Non Self Signed	5	0	9
Certs Without Pointers	68	87	89

Table 1. Profile analysis for certificates embedded in major applications.

webpages of universities in USA and Europe. Table 1 shows the results coming from the study of the certificate profiles from the first set. Most of the certificates 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 has no pointers to any service or data repository (not even to CRLs), while for IE7/Outlook this percentage goes up to 82%. This problem is even worse when taking into account the lifetime of the certificates. Fig. 5 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 analysis 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 of certificates. By contacting all the universities websites [7,8] by using 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 where, most of the time, the same across different organizations. We think that the usage of certificates from commercial vendors, even when an internal CA exists, is due to the lack of real solutions to achieve interoperability between PKIs.

Results for European universities were quite different. In fact out of 2541 universities, only 745 support HTTPS. However, differently from the US case, the number of internally[2] 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 that 54% of certificates did not provided any pointer to PKI resources. From this results it is therefore evident that also for EE certificates,

[2] Issued by the university's internal CA

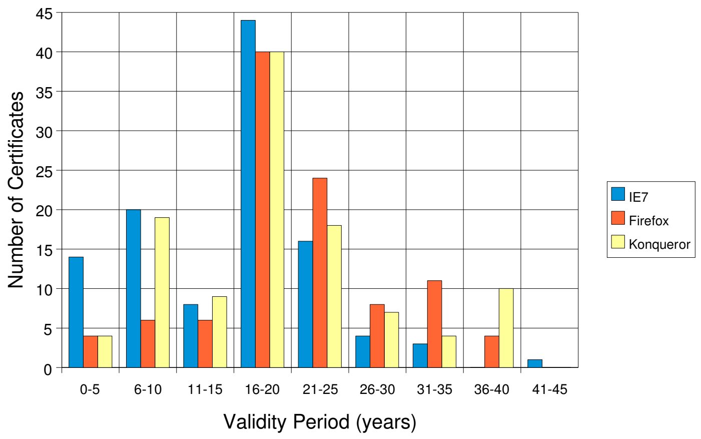


Fig. 5. Validity periods of CA certificates present in today browsers.

such as the ones from university websites, solving the resource discovery problem by simply listing URLs in the certificate does not provide a working solution.

OASIS conducted a survey [9] about PKI deployment. This survey found that support for PKI is often missing from applications and operating systems and, when present, it is always inconsistent in the sense that it differs widely in what is supported. This survey also found that current PKI standards are inadequate as they are often too complicated and implementations from different vendors rarely interoperate. It is interesting to notice that seven out of ten reported problems in the list are related to PKI usability and interoperability.

PRQP could help the deployment of PKIs and PKI-aware applications by providing a flexible way to automatically discover which services and data repositories are available from a CA. By providing such a mechanism, support for PKI basic operations (e.g. certificates validation) could be easily implemented also at the operating system level.

5 Related Work

Our work focuses on the PKI resources look up problem. This problem does not only involves the certificate retieval, but also the discovery of new services whenever they are made available by service providers. In prior work, 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).

5.1 Certificate Extensions

To provide pointers to published data, a CA could use the *Authority Information Access* (AIA) and *Subject Information Access* (SIA) extensions as detailed in

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

Table 2. Analysis of AIA statistics.

RFC 3280 [10]. 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 extension is 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 *End Entities* (EE) certificates, except during periodic reissuing, but it would be 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 cert regularly, rather than caching it, this would enable them to become aware of the new services. Although this is possible, almost every available clients do not look for CAs certificates if they are already stored in clients' local database.

In any case, since URLs tend to change quite often while certificates persist for longer time frames, experience suggests that these extensions invariably point to URLs that no longer exist. 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 available services and repositories look up.

In Table 2 we report the contents of the AIA extensions in most diffused 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/KMail), whilst no pointer to other services are provided.

5.2 DNS Service Records

The SRV record or *Service record* technique is thought to provide pointers to servers directly in the DNS [11]. As defined in RFC 2782 [12], the introduction of this type of record allows administrators to perform operations rather similar

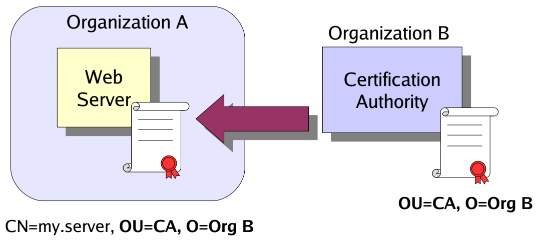


Fig. 6. The Certificate Authority from Organization "B" issues a certificate to the web server from Organization "A".

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 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 `dc=com, dc=example` format. If the CA's subject field would make use of such a format, 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[3] from 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 [13] 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 to locate an LDAP service for a specific e-mail address, the authors were not able to find anyone who announces their directory service in the DNS according to the specification.

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

Another example of the infeasibility of this solution is presented in Fig. 6. 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”).

5.3 Web Services

Web Services [14] is a new technology using three different components to allow applications to exchange data: *SOAP (Simple Object Access Protocol)* [15], *WSDL (Web Services Description Language)* [16, 17] and *UDDI (Universal Description Discovery and Integration)* [18].

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 [19] 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 [20] which is quite complex when compared to other binary formats like DER [21, 22]. 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.

5.4 Local Network Oriented Solutions

Another approach to provide reliable information is to use existing protocols for service location such as *Jini* [23,24], *Universal Plug and Play* protocol (UPnP) [25, 26] or *Service Location Protocol (SLP)* [27–29].

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 [30] 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)* [31], 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.

Indeed, we believe 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.

6 Conclusions

The lack of interoperability among closed PKI islands is a very urgent problem and demands a solution.

One example of an environment where our system could provide measurable improvements is the Grid community, which already make 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) [33], 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 [34]) and without delegating credentials validation to third parties.

The PRQP protocol provides a PKI-specific protocol for resource discovery, and 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.

References

1. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "Online Certificate Status Protocol - OCSP," Internet Engineering Task Force: RFC 2560, June 1999.
2. M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3)," Internet Engineering Task Force: RFC 2251, December 1997.
3. T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk, "Server-based Certificate Validation Protocol (SCVP)," IETF Draft, January 2007. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-31.txt
4. "OpenSSL Homepage." [Online]. Available: http://www.openssl.org/
5. "OpenCA Project Homepage." [Online]. Available: http://www.openca.org/
6. "OpenCA OCSPD." [Online]. Available: http://www.openca.org/ocspd/
7. World List of Universities. [Online]. Available: http://www.unesco.org/iau/
8. Universities Worldwide. [Online]. Available: http://univ.cc/
9. S. Hanna, "Follow-up Survey on Obstacles to PKI Deployment and Usage," October 2003. [Online]. Available: {http://www.oasis-open.org/committees/pki/pkiobstaclesaugust2003surveyreport.pdf}
10. R. Housley, W. Polk, W. Ford, and D. Solo, "Certificate and Certificate Revocation List (CRL) Profile," Internet Engineering Task Force: RFC 3280, 2002.
11. P. Mockapetris, "Domain Names - Implementation and Specification," Internet Engineering Task Force: RFC 1035, Request for Comments, November 1987.
12. 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.
13. S. Boeyen and P. Hallam-Baker, "Internet X.509 Public Key Infrastructure Repository Locator Service," IETF Experimental, September 2005. [Online]. Available: http://tools.ietf.org/wg/pkix/draft-ietf-pkix-pkixrep/draft-ietf-pkix-pkixrep-04.txt
14. 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*, vol. 6, no. 2, pp. 86–93, March 2002. [Online]. Available: http://dx.doi.org/10.1109/4236.991449

15. G. Martin, H. Marc, M. Noah, M. Jean-Jacques, and N. Henrik Frystyk. (2003, June) SOAP Version 1.2. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/
16. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "PWeb Services Description Language (WSDL) 1.1," W3C Note, March 2001. [Online]. Available: http://www.w3.org/TR/2001/NOTE-wsdl-20010315
17. 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. [Online]. Available: http://www.w3.org/TR/wsdl20
18. L. Clement, A. Hately, C. von Riegen, and T. Rogers. (2004, October) UDDI Version 3.0.2. [Online]. Available: http://uddi.org/pubs/uddi_v3.htm
19. (2004, March) Common Object Request Broker Architecture: Core Specification. [Online]. Available: http://www.omg.org/technology/documents/corba_spec_catalog.htm
20. F. Yergeau, J. Cowan, T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. (2004, February) Extensible Markup Language (XML) 1.1. W3C Recommendation. [Online]. Available: http://www.omg.org/technology/documents/corba_spec_catalog.htm
21. "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 (1994) — ISO/Uniform Resource Locators (URL)IEC 8825-1:1995, 1994.
22. "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 (1994) — ISO/Uniform Resource Locators (URL)IEC 8825-1:1995, 1994.
23. W. Edwards, "Core Jini (2nd edition)," Prentice-Hall, 2000.
24. K. Arnold, "The Jini Specification (2nd edition)," Addison-Wesley, 2000.
25. Universal Plug and Play Specifications. [Online]. Available: http://www.upnp.org/resources/specifications.asp
26. M. Jenronimo and J. Weast, "UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play," 2003.
27. E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, version 2," Internet Engineering Task Force: RFC 2608, June 1999.
28. E. Guttman, C. Perkins, and J. Kempf, "Service Templates and Schemes," Internet Engineering Task Force: RFC 2609, June 1999.
29. E. Guttman, "Service Location Protocol: Automatic Discovery of IP Network Services," *IEEE Internet Computing*, vol. 3, no. 4, pp. 71–80, 1999.
30. (2003) Java RMI Specification. [Online]. Available: http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html
31. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright, "Simple Service Discovery Protocol," IETF Draft, October 1999. [Online]. Available: http://www.ietf.org/internet-drafts/draft-cai-ssdp-v1-03.txt
32. OpenSLP Project. [Online]. Available: http://www.openspl.org
33. International Grid Trust Federation. [Online]. Available: http://www.gridpma.org
34. "Education Roaming (Eduroam) Homepage." [Online]. Available: http://www.eduroam.org/