

Usable Trust Anchor Management*

Massimiliano Pala
Department of Computer Science
Dartmouth College, Hanover, NH
pala@cs.dartmouth.edu

Scott A. Rea
Institute for Security, Technology, and Society
Dartmouth College, Hanover, NH
Scott.A.Rea@dartmouth.edu

ABSTRACT

Security in browsers is based upon users trusting a set of root Certificate Authorities (called Trust Anchors) which they may know little or nothing about. Browser vendors face a difficult challenge to provide an appropriate interface for users. Providing usable Trust Anchor Management (TAM) for users, applications and PKI deployers is a complex task. The PKIX working group at Internet Engineering Task Force (IETF) is working on a new protocol, the Trust Anchor Management Protocol (TAMP), which will provide a standardized method to automatically manage trust anchors in applications and devices. Although promising, this protocol does not go far enough to allow users to gather information about previously unknown trust anchors in an automatic fashion. We have proposed the PKI Resource Query Protocol (PRQP)—which is currently an Internet Draft on Experimental Track with IETF—to provide applications with an automatic discovery system for PKI management. In this paper we describe the basic architecture and capabilities of PRQP that allow Browsers to provide a more complete set of trust anchor management services. We also provide the design of a PRQP enabled infrastructure that uses a trust association mechanism to provide an easy solution for managing Trust Anchors for Virtual Organizations.

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection—*authentication*

General Terms

Security

*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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IDtrust '09 April 14-16, 2009, Gaithersburg, MD
Copyright 2009 ACM 978-1-60558-474-4 ...\$5.00.

Keywords

Trust Anchor, PRQP, Discovery System, Digital Certificate, PKI

1. INTRODUCTION AND MOTIVATIONS

Browser—based trust decisions are facilitated by a set of Trust Anchors (TA) that come preloaded in the browser or via an operating system based Trust Anchor Store (TAS) that the browser relies upon. These TAS contain many trust anchors that the average user has no idea in respect to their purpose or community of applicability, yet the browser makes trust decisions on behalf of those users based on the presence and configuration of these elements.

At the NIST PKI Workshop in Gaithersburg, MD in 2007 [10], Kelvin Yiu from Microsoft made the revelation that browsers were struggling with the growing size of the TAS—he indicated that Microsoft's browsers were originally designed with a TAS of approximately 100 elements in mind, with an upper limit of 200. As PKI communities continue to proliferate, it was plainly apparent that the current processes of managing the TAS primarily at bootstrap and relying upon users to tweak thereafter was not going to be sufficient [19]. As of today, a simple analysis of the default TA stores in Operating Systems (OSes) and browsers reveals 132 TA in Firefox, 152 TA in OSX, 286 in XP, and (obviously heading in the right direction) just 30 TA in Vista.

A simple task like path construction and validation in hierarchical PKIs still raise many issues today that are not completely solved thus impacting on the operation and management of large-scale PKIs. Most end-user applications today still rely exclusively on trust lists. For example, some of the most widely deployed network applications such as browsers or Mail User Agents (MUAs) use Trust-Lists as a basis for trust. As described in [15], trust lists build trust by embedding digital certificates locally into applications. This approach leads users to completely rely on the application vendor for management of her own Trust Anchors without being actively involved in deciding what is trustworthy and for what purpose. On the other end, the application vendor is forced to manage a quite large set of trust anchors— i.e. the ones that are embedded into the shipped application.

Another important aspect to consider is the low level of users' awareness about trust management [5]: in general users do not understand protection technologies and poor user interfaces exacerbate the problem. The authors practi-

cal experiences participating on the EuroPKI [1, 14] project as well as being reported in other realities [16] show that users often require specialized help and support in order to correctly add certificates to the ones trusted by default in a given application. Therefore using pre-cooked TA stores encourages people to accept by “faith” whatever is hardwired into applications.

There are also environments, such as Grid Computing communities [2–4], that are very sensitive about TAM. In these communities interoperability is extremely important and PKI administrators and application developers struggle with problems related to the lack of flexibility in trust anchor management.

Our work addresses trust management issues from a practical point of view by providing a model which both helps in removing the reliance upon pre-installed trust lists, and provides a local trust management system by using cross-certification. The purpose of our work is to let a single organization (be it a company or an aggregate such as a research network or a Virtual Organization) to unilaterally create trust for its users of selected foreign PKIs or CAs in a way that is easily supported by current applications.

The rest of the paper is organized as follows. Section 2 presents the related work and current limitations. Section 3 introduces an overview of the possibilities offered by a dynamic TA management system. Section 4 describes our new hybrid trust model. Section 5 explains how to integrate our hybrid trust model with PRQP in order to provide a dynamic TA management system. Section 6 contains our conclusions and future work.

2. RELATED WORK

Since the 1976 paper by Diffie and Hellman [22] and the introduction of a public key cryptosystem [18], the presence of some sort of trusted directory for public key access has been a steady reality in PKIs. The very nature of the trusted directory strictly depends on the trust model used in the infrastructure. In fact the organization of a PKI reflects the trust model required by its constituency. In X.509-based PKIs (or PKIX), there are three different “pure” trust models: hierarchical, cross-certification (e.g. bridge CA), and trust lists. In addition to individual “pure” models, combinations of these models may be adopted, with varying capacities and implications for interoperability. In all cases models require that participants obtain trusted knowledge of one or more public keys to enable them to discover and validate certification paths. This information may be obtained by using special configurations, by out-of-band management and/or by explicit acceptance of keys offered during network exchanges. In PKIX (X.509-based PKIs), trust anchors are usually provided in the form of self-signed certificates. Although this approach is a convenient implementation strategy and allows integrity checking, it does not add any fundamental trust enhancement relative to other representations.

In the web environment, where browsers are preloaded with a high number of trusted root keys, the inclusion of root certificates have become commercially valuable assets. Trust

lists are commonly used in major off-the-shelf applications and provide a very simple solution to the trust management problem for the average user, but they are criticized because the criteria for insertion of a CA into the list are often based more on a commercial rather than a security analysis. Also, the use of the browser as the interface to many internet and local applications means that the TAs trusted for one context do not necessarily mean they should be trusted for a different context (yet this is implied by the way the TAs are used). Therefore we deduce a compelling need for a usable approach to provide Trust Anchor Management services that would allow organizations to dynamically manage TAs for a large number of users.

2.1 The Trust Anchor Management Protocol

In response to this growing demand for bulging TAS, the IETF PKIX Working Group set about defining a new protocol (TAMP) that would help to manage them in an automated standardized way. TAMP is a transport independent protocol that allows an entity designated as a Trust Anchor Manager to query for status and update TAS with TA elements. As TAMP is making its way through the standardization process, it has become apparent through comments on the discussion lists, that the protocol is not broad enough to cover all use cases for TA management. In fact TAMP defines only three types of TA:

- (a.) An Apex TA which is the TA that is superior to every other TA within the TAS and is used to manage the rest of the TAS elements;
- (b.) one or more Management TA that is used to manage cryptographic modules within the TAS;
- (c.) one or more Identity TA that are the traditional X.509 anchors used to validate certificate paths for typical PKI

Obviously there are many scenarios where the choice of Apex TA leads to issues. In particular allowing a user to control their own Apex TA may undermine the possibility for all the browser (and OS) vendors to adopt TAMP in the preferred way to manage their respective TAS (however, the later may simply be an effort by browser vendors to bind users to them by holding onto the responsibility of managing trust for the users, but could also simply be a function of the TAMP standard not being completed and accepted by the community yet).

Moreover the current document which specifies the requirements for the TAM protocol [17] restricts the design to the “push” model. In this model, a centralized service would directly manage the application(s) TA store by pushing the content directly to the application. This approach is specifically adopted because the current version of TAMP is thought to be efficient for enterprises where the control over the clients can be very strict. Future versions of the protocol, however, could extend this approach to a “pull” model which would allow for more interoperable TAM services across an organization’s boundaries.

One potential issue with TAMP is that it specifies that there should be one and only one Apex TA for each TAS. While this makes perfect sense from a management protocol perspective, it has implications for the browser trust model. A

Table 1: Comparison between Trust Models.

	Hierarchies	Cross-Cert	Bridge CA	Trust Lists
Trust Anchor	Hierarchy Root	Local CA	Local CA	Listed CAs
Inter Domain Support	Poor	Good	Very Good	Good
Path Construction	Very Easy	Hard	Easy	Very Easy
Repository Dependency	Low	High	High	Low

single root that controls the trust settings of all other trust anchors in a browser TAS should only be allowed where the Apex TA is under direct control of the user. If the Apex TA was controlled by a single vendor, then that vendor could potentially lock out any other vendors from being accepted for that user, and enforce a restriction of trade. In some sense, this is the situation today with the current browser trust model—the browser vendors determine who is in the trust list i.e. the vendors act as an Apex TA—however, out of band updates by users are permitted in the current browser model, and it is not clear that the equivalent functionality is supported in TAMP. Also, TAMP being an “automated” management protocol, means that updates occur without notifying the user once the initial subscription has been entered into.

2.1.1 Current Limitations

Putting the responsibility for managing their TAS into the hands of an uneducated or inexperienced user also has severe trust implications—which is why the vendors have generally undertaken the current process to make a best effort determination at who should be trusted by default and who should not, and allow the users to manage it from there. The problem with this approach is that almost all average users have no idea how to make a trust decision about a given TA. Sure, there are standards and processes that can be utilized to facilitate the trust decision, and that is what most browser vendors currently do for the TAS default set, but a user can not be expected to take on that responsibility as an individual, when even experts in the field have issues agreeing on trust implications on a regular basis. For instance, just because an organization spends \$120K to get a Web Trust audit, it does not necessarily mean that it runs a benevolent Certification Authority (CA)—nor does getting a Web Trust approval necessarily mean that it is running a secure and trustworthy PKI! Yet often this is the yard stick applied to vendor gated TAS.

Putting aside the question of individuals managing their TAS for the moment, a business or enterprise must also make decisions about what TA they intend to trust and for what purpose. This is based upon some assessment (usually risk-benefit based), and they advise their constituents via policy or other method as to which TA’s should make up a TAS that is acceptable to their enterprise. Individuals within the enterprise then adjust their TAS based on the advice or policy from the enterprise or community to which they belong. TAMP is a fantastic protocol to facilitate this process as long as the individual has some way of establishing trust with the enterprise or community. Since individuals are most often not restricted to having trust relationships with a single enterprise or community, an individual adopting a single Apex

TA from a given enterprise is not a sufficient solution, since this may lead to conflicts for an individual as they interact with multiple organizations, by allowing the TA policy (and Apex TA) of one organization to control the trust settings of another.

Therefore we propose that browsers should support a single Apex TA, instantiated by a user, that they can utilize to subscribe to multiple TAS management services (which may be organization based), and that the browsers support multiple TAS instances that are scoped for a given purpose or application, for a given enterprise or community of interest. That way, a user of enterprise X services could subscribe to the enterprise X TAS Management Service (TMS), which would automatically update a partition of their browser of choice TAS, with those TA’s that enterprise X has determined as trustworthy. The user’s browser would then rely upon those trust settings in that partition of their TAS, for all domains that the user indicates they are applicable to.

This approach would make it possible, for instance, for a user to subscribe to a company TMS, and have a partition of their browser TAS (it could be the entire TAS, or a separated store altogether) be automatically set to have the trust settings that the company recommends. The user could then apply those trust settings globally if they wished to all transactions they undertake, or could restrict it to only those dealing with services from the company domain. This would mean that browser vendors could be relieved from managing users initial TAS (thus reducing costs and effort) or offer it as a separated service. Consequently, users would have a simplified trust decision to make rather than having to evaluate each TA individually, that is “do I want to adopt the recommended TA’s from organization X ?”.

Now in order to make the above feasible, it should be possible to discover what PKIs are trusted by what organization, whether an organization has a TMS, and how to access those services. Until recently, there was no mechanism to discover these details.

The PKI Resource Query Protocol (PRQP) [11,12]—recently adopted as a working item by the PKIX Working Group at IETF—would facilitate users and application vendors by being able to discover appropriate PKI and TMS services.

2.2 The PKI Resource Query Protocol in a Nutshell

Integrating all current protocols and standards to provide an efficient way to discover PKI related services is an open challenge. The PRQP protocol provides a simple approach that changes the current paradigm by allowing for more dy-

dynamic management and configuration of applications and their TAS.

PRQP assumes the presence of a Resource Query Authority (RQA) that would provide applications with the locators of services associated to a particular Certification Authority (CA). In PRQP, the client and a RQA (the server) exchange a single round of messages where the client requests a resource token by sending a request to the server and the server replies back by sending a response to the requesting entity. 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 *PRQP Trusted Authority* (PTA). In this case, the RQA may provide responses about multiple CAs without the need to have been directly certified by them. 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.

3. DYNAMIC TRUST ANCHOR MANAGEMENT: THE NEXT CHALLENGE

PKIs enable trust judgments between distributed users without the need of a direct interaction between them. However, being able to securely identify a user is not sufficient. Indeed in order to make trust judgments, a relying party needs more information than the user's bare certificate. For example, a Web Server or a SSO application must be able to locate critical parameters such as the certificate repositories and certificate validation servers relevant to the trust path under consideration.

Current PKIs often fail to provide such integration, therefore application vendors and users struggle when trying to enable all the supported protocols. This failure is primarily due to the lack of (a.) interoperability between PKIs and (b.) availability of pointers to services. In [11], the authors point out how current CA certificates do not provide a good source of information when it comes to discovering the services that are associated with them. For example, out of the analyzed browser stores¹, almost no service locators were provided in the certificates (besides a few OCSP pointers). This lack of flexibility and service availability is also present in current TA management. Indeed each party involved in TA management deals with it independently without considering all the different points of view of the other parties.

When considering Trust Management issues, one of the most common errors is to restrict the view to a single perspective only. We think that the problem should be analyzed from, at least, three different points of view: the user perspective, the application vendor perspective, and the PKI vendor perspective. In this section we focus on the benefits of adopting a dynamic approach to TAM and how PRQP is a fundamental building block of a viable TAM infrastructure.

The User Perspective. *As far as most users are con-*

¹Firefox, IE, and Konqueror

cerned, the notion of Digital Certificates or Public Key Infrastructure is a mystery. Many papers [5,20] have discussed the issue of the lack of understanding about security and the underlying technologies. Therefore we need a new approach that will allow Users to make informed Trust decisions.

A Recent paper [7] has shown that users understand trust through the concept of Institutions and their reliability more than Digital Certificates and Certification Authorities. In particular the study shows how users tend to rely on well known institutions and their level of reliability more than other users' suggestions (also if well known) or technical details. It is therefore clear how the concept of trust should be presented to the users in terms of what they can really understand: "who already trusts this organization (eg., Certification Authority) ?"

By integrating PRQP with current applications and enabling the automatic discovery of PKI-related resources, the TA management problem can be actively put into the hand of the users. One major hurdle to overcome however, is how the interface for managing these services are presented to, and interacted with by the user. A sample User Interface (UI) from a common application is reported in Figure 1. It is evident that such interface is too complicated for the majority of users. In order to be able to provide an easier UI, the application could dynamically discover all the services provided by the PKI and the trust information about the CA provided by other organizations, and present this additional information to the user in a meaningful way. The PRQP protocol would allow applications to provide such an interface for Trust Management.

The Application Vendor Perspective. *As far as the average developer is concerned, the notion of Digital Certificates or Public Key Infrastructure is also a mystery.* Moreover the complexity of current PKI standards makes it difficult for developers who are not PKI experts to correctly process the information within certificates. Also, because of the number of different services and repositories to access in order to build a simple certificate chain, the average application either does not implement correctly and fully the standards or has dozens of options to configure those services and they pass those decisions onto the user who has no idea how to set them up.

To solve this problem PRQP provides a simple and efficient way to enable applications to easily discover PKI services. We estimate that it would be possible to get rid of most of the current configuration options faced by vendors in preparing UIs, thus allowing for easier UIs and faster development times in applications.

In particular, for TAS Management, PRQP could be used to discover TAMP data sources. Multiple sources could be dynamically discovered for specific domains and a hierarchy of stores can be built according to the user's preferences. The Application Vendor will still be able to dynamically provide its own Trust Anchor Management updates (e.g., for OS management purposes or specific trusted domains) while the user would be able to add its own settings.

The PKI Vendor Perspective. There are many differ-

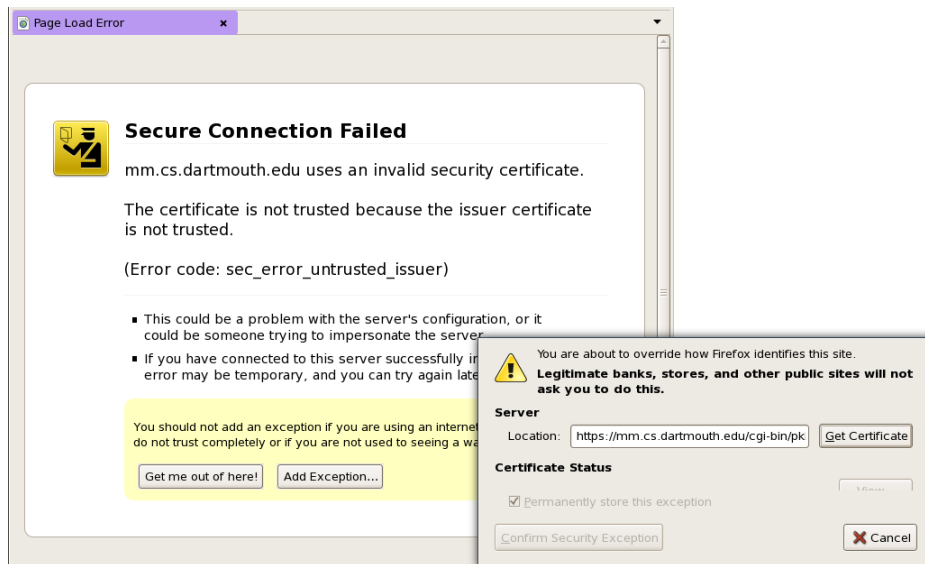


Figure 1: Example UI—Understanding the full consequences of adding a Trust Anchor is beyond the ability of the average users and of many advanced ones. Also, interfaces from other applications suffer from similar problems: either it is misleading and too complex or it provides no useful information to the user.

ent types of PKIs that serve different purposes. Besides SSL vendors, a huge market for PKI related services is today totally ignoring commercial vendors. Environments like Computing Grids where Public Key Certificates have been used for well over a decade are, today, left with little support by application vendors. By enabling organizations to integrate different PKIs and provide dynamic TA management, the occasion for a bigger market is evident. The more usable PKIs are, the more the users will be willing to adopt this technology. There are potentially real world-wide client-side PKIs everywhere you look—e.g, cellular phones, mobile devices, wireless network authentication, home devices, e-commerce application, etc...

In order to provide more flexible support to the users, PKI vendors need the capability to deploy more flexible PKIs. In particular the adoption of PRQP—and eventually its Peer-2-Peer extension [13]—can enable vendors to activate, move, dismiss, and enhance services easily. This provides PKI deployers with the possibility of molding the infrastructure as the needs or profile of the users change thus being able to offer a more usable experience of the offered products. An example of the benefits of adopting PRQP would be the seamless dismissal of an old (or potentially broken) service—like CRLs—in favor of a new one—e.g. OCSP—without the need of re-issuing every certificate because of the usage of static extensions.

As we wait for trust anchor management protocols to wind their way through the standardization process, we still need to manage trust anchors in environments today. In the next section we present a hybrid trust infrastructure that could be deployed in current systems to help manage TAs until TAMP matures.

4. TA MANAGEMENT FOR CURRENT APPLICATIONS

While waiting for standard protocols to be finalized and adopted by applications, the need for a solution is compelling. The trust model we present in this section allows organizations (or Virtual Organizations such as accreditation bodies) to provide users with an easy-to-use solution which is compatible with deployed software and supports the way users make trust decisions.

When already deployed infrastructures—established by different organizations—want to develop a common trust relationship between them, some form of cross-certification must be applied to link them. Table 1 reports a summary of the characteristic of different trust models. It could be desirable to have a model which provides the flexibility typical of cross-certification while not introducing high complexity for certification path building. Our work is based on the integration of two different components: a deployed PRQP infrastructure and a hybrid trust model. Our solution is capable to address trust management issues by providing:

- a method to avoid multiple trust points hardwired into applications (i.e. to avoid embedding large trust stores); our approach requires a single trust anchor (e.g. a Root CA which could, in future, act as an Apex TA under TAMP) for inter-hierarchies trust support
- a simple trust management system for applications based on cross-certificates
- the possibility for PKI operators to provide their users with a set of revocable endorsed TAs

Cross-Certification has the interesting characteristic that it can preserve the organization's ability to enforce constraints within their hierarchies. Annex G of the ISO document [9] discusses specifically this case by presenting three different types of Cross-Certificates:

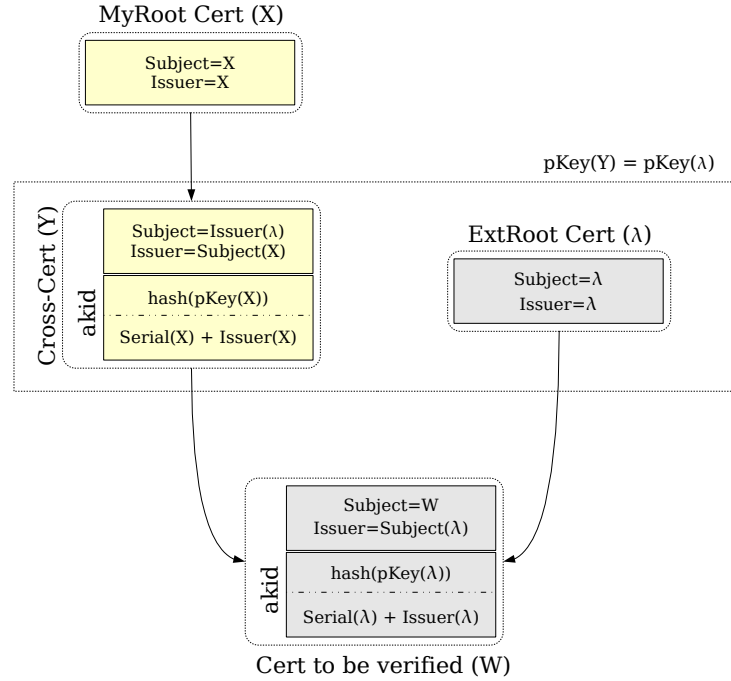


Figure 2: Cross-Certificate of an external Root CA.

- *Hierarchical Cross-Certificates* extend path construction from leaf CAs upwards Root CA, thus allowing relying parties to use their local CA as trust anchor
- *General Cross-Certificates* to interconnect CAs. This can be done either at root level or at sub-CA level
- *Special Cross-Certificates* are intended to allow selective establishment of certification paths that may not conform to the restrictions imposed hierarchically

In this work we combine the hierarchical and cross-certification models, by using “special” cross-certificates to allow for locally managed trust path building for applications. In order to better understand how our model is particularly efficient and why it is supported out of the box by existing applications, the next subsection provides a brief description of the certification path building process in PKIX.

4.1 Certification Path Building

Assuming we have a set of trusted certificates and we need to build a trust path from certificate x up to one trust anchor, we have to identify the issuer of certificate x and check if it is trusted or not. In case it is not, we need to verify if its issuer is trusted. The process continues until a trusted certificate is identified in the chain or no suitable issuer is found among the available certificates or, finally, a self-signed certificate is reached. Therefore the identification of the issuer of a certificate is a crucial aspect of the path building process. Assuming we want to check if the subject of certificate y is the issuer of certificate x , the needed steps are:

- check the *Issuer* field of certificate x to be equal to the *Subject* field of certificate y
- If *authorityKeyIdentifier* extension exists in certificate x , then check it matches the data of certificate y

- Check that the *keyUsage* in certificate y supports certificate signing
- Check that the policy and name constraints requirements are fulfilled

To continue the chain building process, repeat the steps above till one trust anchor or a self-signed certificate is reached. Special attention should be made during step (b). In fact the presence and the contents of the *authorityKeyIdentifier* can vary depending on the certificate to be verified. Four different certificate profiles are hereby reported which summarize the possible contents of the *authorityKeyIdentifier* (*akid*) extension:

- α the extension is not present in the certificate. The chain is actually built by using the subject and issuer fields of the certificate
- β the extension is present in the certificate and it carries the *keyIdentifier* which, usually, contains the hash of the public key of the issuer
- γ the extension is present in the certificate and it carries the *authorityCertIssuer* and the *authorityCertSerialNumber*.
- δ the extension is present in the certificate and it carries both the *keyIdentifier* and the *authorityCertIssuer* and *authorityCertSerialNumber* couple.

The first two profiles do not require special care when dealing with cross-certification while, as we will discuss in detail in the next parts of this section, the other two do. In fact, within the last two profiles, the certificate’s issuer is identified both by the *Issuer* field contents and by its issuer and its serial number in the *akid* extension. For instance if we want to identify the certificate x we use the couple:

$$issuer(x) + serialNum(x) \quad (1)$$

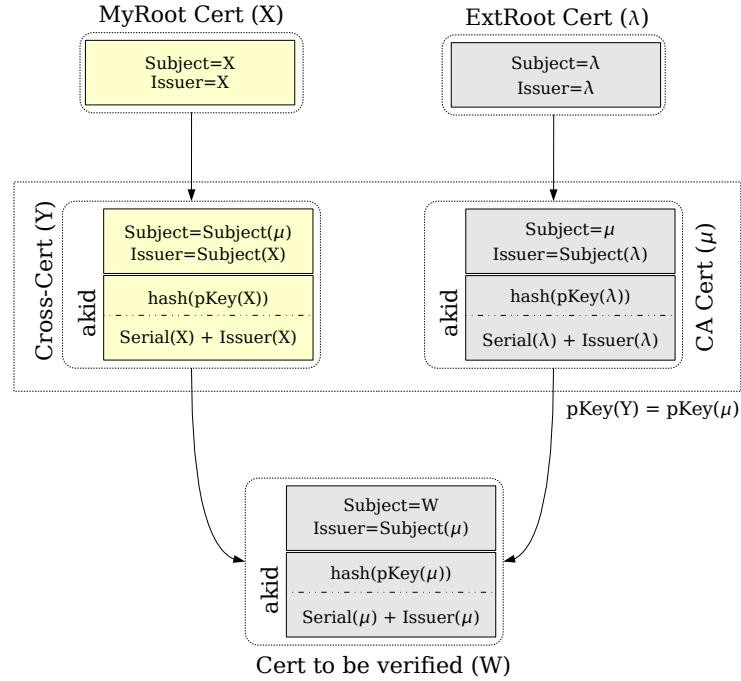


Figure 3: Cross-Certificate (Y) for an external Sub-CA (μ).

in the same way, if we want to identify the issuer of certificate x we use the couple:

$$issuer(issuer(x)) + serialNum(issuer(x)) \quad (2)$$

where the $issuer(x)$ is the *Subject* of the issuer of x . This explanation is needed to better understand differences in our trust model when trusting Root CAs or subCAs as explained in Sections 4.3 and 4.4.

4.2 The Model Basics

In our model we assume that at least one CA certificate is installed into the application certificate store and it is trusted, e.g. this may be the Apex TA in future as required by TAMP. To better introduce our solution, a practical example could be represented by a National Research Network (NREN)–named OrganizationA— which already established a PKI and wants to provide a way to automatically verify all the certificates issued by other n NRENs. The root CA from OrganizationA ($orgA_rootCA$) cross-certifies the other NRENs root-CAs (i.e. $extCA_1, extCA_2, \dots, extCA_n$) by issuing certificates carrying the same details as the original ones (e.g. Subject, Public Key and Extensions). The newly issued certificates will only differ from the original ones in that they are issued by $orgA_rootCA$ within the OrganizationA’s infrastructure. The cross-certificates will be referred as $extCrossCA_{1\dots n}$.

If a user from OrganizationA’s hierarchy wants to verify a certificate issued by one of the $extCA_{1\dots n}$ hierarchies (e.g. in order to verify a signed S/MIME message) then a path from that certificate ($extUserCert$) up to $orgA_rootCA$ has to be established.

By providing the $extCrossCA_{1\dots n}$ certificates to the client, it is possible to build a trusted chain of certificates up to the OrganizationA’s root-CA. Indeed by comparing the *authorityKeyIdentifier* in the $extUserCert$ with the locally stored $extCrossCA_{1\dots n}$ certificates’ contents, the path construction can proceed up to $orgA_rootCA$, thus establishing a trusted chain from $extUserCert$ up to the only trust anchor needed in the application.

Although the basic principles are very simple, further analysis is needed to correctly allow the path building process to take place. In the next subsections we provide the details of our hybrid trust model. In particular we will use the following terminology:

- $issuer(x)$ identifies the Issuer field in certificate x
- $subject(x)$ identifies the Subject field in certificate x
- $serialNum(x)$ identifies the serialNumber field in certificate x

4.3 Trusting Root CAs

Extending the trust to a whole external PKI is done by issuing a cross certificate for the external root-CA. Figure 2 reports the scenario where the certificate W of the external PKI is linked to the organizational PKI throughout Y . If the *akid* extension is not present (case α and β of Section 4.1), the cross-certificate Y will use the same public key and *Subject* of the original root-CA whilst having a different *Issuer* ($MyRoot$). If the *akid* extension is used (case γ and δ in Section 4.1) it is important to issue the cross-certificate in such a way it will be referred by the *akid* of certificate W . In this particular case of cross-certifying a root-CA, the *Issuer* and the *Subject* contents of the certificate are the

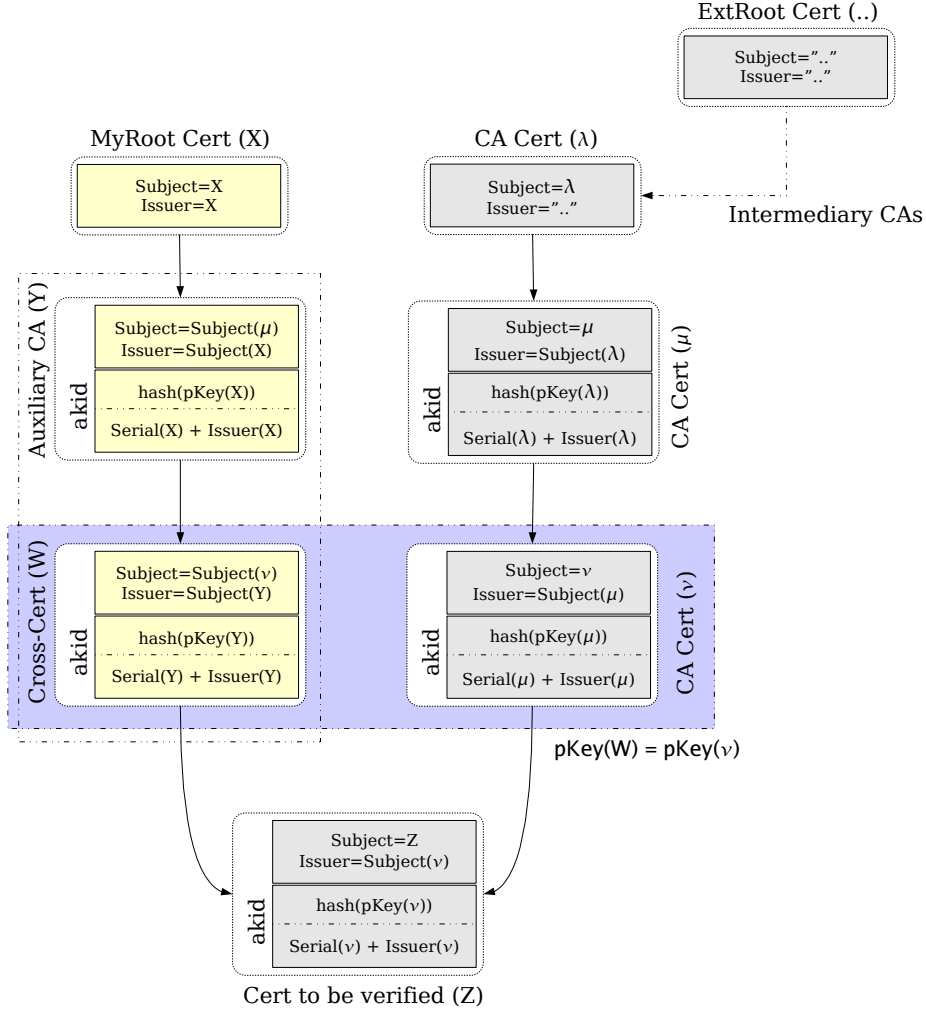


Figure 4: Our Hybrid Trust Model with Auxiliary CA (Y).

same. This means that, according to (2), the issuer of the root-CA is identified in the *akid* by:

$$issuer(issuer(rootCA)) + serialNum(issuer(rootCA)) \quad (3)$$

where:

$$issuer(rootCA) = subject(rootCA) \quad (4)$$

$$serialNum(issuer(rootCA)) = serialNum(rootCA) \quad (5)$$

therefore (3) becomes:

$$issuer(subject(rootCA)) + serialNum(rootCA) = subject(rootCA) + serialNum(rootCA)$$

hence if the certificate W (with the *akid* extension carrying the $serialNum(\lambda) + Issuer(\lambda)$ identifier) points correctly to Y if and only if:

$$subject(Y) = Issuer(\lambda) = subject(\lambda) \quad (6)$$

and

$$serialNum(Y) = serialNum(\lambda) \quad (7)$$

The condition (7) is the most hard to match. In fact typically rootCA certificates have the *serialNum* set to zero and, as the serial number must be unique within a CA, it is difficult that the serial number required for Y is available under X . As we will discuss in detail in the next sections, the *serialNum* constraint can be fulfilled by introducing into the model an auxiliary CA for each external CA/PKI we want to establish a trust with.

4.4 Extending the model: trusting subCAs

Figure 3 represents the scenario where we want to issue a cross certificate to include only a sub-CA, not a whole external PKI. In this case we want to be able to verify certificate W by building the chain:

$$W \longrightarrow Y \longrightarrow X \quad (8)$$

to do this, the cross certificate Y will have:

$$subject(Y) = subject(\mu)$$

if no *akid* extension is present in W or if it contains only the *keyIdentifier*, the path building process does not present

particular issues. On the contrary if we are in case γ or δ (Section 4.1), then the path building process will fail because it is impossible to issue Y obeying the *akid* constraints. In fact the *akid* of W identifies the *issuer*(W):

$$\begin{aligned} authorityCertIssuer(W) &= serialNum(\mu) + issuer(\mu) \\ &= serialNum(\mu) + subject(\lambda) \end{aligned}$$

and to fulfill its requirements (besides the *serialNum* problem) it should be:

$$issuer(Y) = issuer(\mu) = subject(\lambda) \quad (9)$$

unfortunately with the proposed infrastructure, this is not possible because:

$$issuer(Y) = subject(X) \neq subject(\lambda) \quad (10)$$

To overcome this problem, an addition to the model is needed, as detailed in the next section.

4.5 Introducing auxiliary CAs

To provide a generally applicable model we introduce an auxiliary CA (Figure 4) into our schema. The purpose of this CA, which is identified by the certificate Y , is to provide a more general solution that is capable of addressing the potential limitations imposed in the path building process by the *akid* extension.

Indeed, to have the *akid* to correctly point to the cross-certificate W , we have to set *subject* of the auxiliary CA Y to be equal to the *issuer* of the sub CA μ . By adding Y to the infrastructure, we have:

$$subject(W) = subject(\nu) \quad (11)$$

$$issuer(W) = subject(Y) = subject(\mu) = issuer(\nu) \quad (12)$$

$$serialNum(W) = serialNum(\nu) \quad (13)$$

therefore the *akid* of Z points correctly to W because the serial number of W is equal to *serialNum*(ν) as in (13), its issuer is equal to *Issuer*(ν) as in (12) and its public key is equal to *pKey*(ν) as W is the cross certificate for ν .

Moreover the *Subject/Issuer* content requirements are also satisfied because:

$$issuer(Z) = subject(\nu) = subject(W) \quad (14)$$

It is therefore possible to extend trust to external PKIs/CAs by introducing only one auxiliary CA that issues one cross-certificate for the CA to be trusted.

4.6 Revoking Trust Anchors

One interesting aspect of the proposed model is the possibility of revoking TAs by using standard PKIX mechanisms. In fact, because trust is built by issuing standard certificates, it is possible to revoke them by simply using revocation services that are already in place (e.g., CRLs or OCSP).

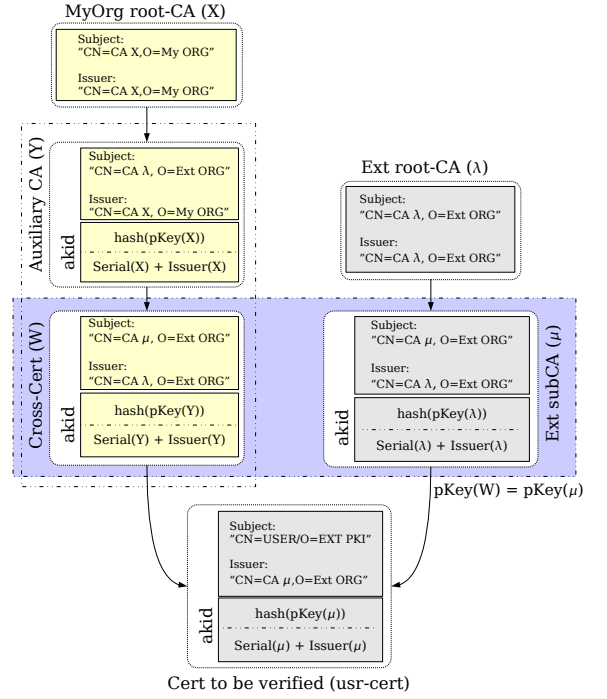


Figure 5: Test bed environment.

4.7 Application Support

One of the main advantages of this solution is that it is supported out-of-the-box by several widely diffused operating systems and applications. This section focuses its attention on performed tests and code analysis (whenever possible) to describe how the hybrid trust model is actually supported. Figure 5 depicts the used test environment. The certificate to be verified in the example is the “usr-cert” which is issued by subCA- Y from the external organization we want to establish a trust link with. In order to do that, we issue a certificate for the subCA- Y which acts as the auxiliary CA in our hybrid trust model. Then subCA- Y issues the cross-certificate subCA- W which, in our model, “replaces” the original certificate from the external organization in the path building process. The purpose of the performed tests was to establish if the proposed trust model was supported by current applications.

4.7.1 OpenSSL libraries

In the Unix world (e.g. Linux, BSD, Solaris, etc...), the OpenSSL suite provides the most used cryptographic libraries. By analyzing the OpenSSL code, it has been possible to discover that our model is actually supported by OpenSSL. Anyway further considerations are needed for special covered cases.

The OpenSSL cryptographic library provides the functions needed to build the chain of certificates and to verify them. To better understand how the library verifies certificates, it is useful to describe the main involved data structures. OpenSSL uses different objects during the verification process:

- the `X509_STORE` object is actually used to represent a collection of certificates and eventually certificates revocation lists (CRLs)
- the `X509_STORE_CTX` object holds the data used during an actual verification

After loading all the needed data in the `X509_STORE`, OpenSSL uses this datastructure to initialize the `X509_STORE_CTX` by calling the following function:

```
int X509_STORE_CTX *X509_STORE_CTX_init(...)
```

If the initialization function completes successfully a pointer to a `X509_STORE_CTX` data structure is returned (`ctx`). The following function is then used to perform the verification of the chain of certificates:

```
int X509_verify_cert(X509_STORE_CTX *ctx);
```

The verify function builds the chain up to the trust anchor by looking in the `ctx` for a suitable issuer of the current certificate. This process is then repeated until no issuer for the certificate is found in the `ctx` either because of an error or because a trust anchor has been reached. The checks performed to see if certificate *B* has been issued by certificate *A* are:

- Check the **Subject** field of *A* to be equal to the **Issuer** field of *B*
- If *authorityKeyIdentifier* exists in *B*, then check it matches details of certificate *A*
- If *keyUsage* extension is present in *A*, then check it supports certificate signing
- returns 0 on success, or a positive for the reason for mismatch

Thanks to the addition of the auxiliary CA, the OpenSSL verification function returns successfully if the calling application provides the chain up to the trust anchor. Tests have been carried out by using applications which use these libraries (i.e. KMail and Konqueror) and, as we expected, results were positive.

4.7.2 Mozilla Suite

The explained hybrid trust model provides a method for trust path building that follows RFC-5280 [6], therefore all applications should be able to support the model. From the performed tests, we found out that our model is fully supported when one of the following conditions is matched:

- the CA to be trusted is a rootCA
- the CA is a subCA and the client does not have the original the chain of certificates up to the external rootCA stored in the local repository.

Regrettably, Mozilla based applications (i.e. Firefox and Thunderbird) do not fully support our solution in the remaining cases because of the way certificates are stored in its local repository. In fact if the original chain of certificate is already present in the certificate store an error about the presence of two certificates issued by the same authority with the same serial number is displayed to the user when importing the cross certificate for the subCA. This is obviously an error in the certificate store as it reports wrong information to the user, i.e. the cross-certificate has the same

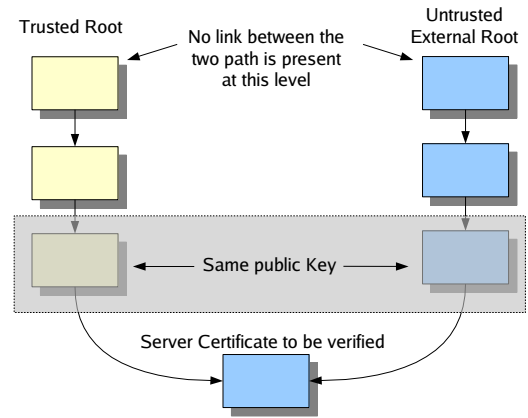


Figure 6: SSL/TLS connections and our hybrid trust model.

serial number but its issuer is actually different. Hopefully this behavior will be fixed in future versions of the software. As detailed in Section 4.7.4 issues are also present when SSL/TLS connections are considered.

4.7.3 Windows CryptoAPI

It has been possible to successfully test support for our model on Windows based systems, in particular it was possible to correctly verify certificates in:

- Windows 2000
- Windows XP
- Windows 2003
- Windows Vista

Unfortunately, it was not possible to verify how exactly the path building is done by Windows CryptoAPI because of the lack of the libraries source code. Anyway every application that relies on the Windows system libraries should automatically support this model. Performed tests show that support for the proposed model is available in Internet Explorer as well as in Outlook.

4.7.4 SSL and TLS connections

Besides applications based on Microsoft CryptoAPI that fully supports the proposed trust model, our tests show that some of the tested software presents issues when setting up SSL/TLS channels. In particular the problem we found is present only when extending trust to *non-root CAs* (Section 4.4). In OpenSSL, for example, the function used to build and verify the path up to a trusted anchor for secure channel setup is different than the one used for statically verify a chain of certificates. This is due to the fact that the RFC-2246 [8] standard allows the server to push the chain of the certificates to the client. Figure 6 depicts the scenario.

When setting up an SSL/TLS connection, the application makes use of the `SSL_CTX` family of functions. These functions use the chain of certificates that is pushed through the channel instead of the certificates in the local store up to the TA (which is verified against the local store). Therefore when the last certificate in the pushed chain is to be verified, there might be no way for the application to link the

trusted anchor (in the local store) to the external root (in the original trust path). A possible way to avoid this situation would be to check, at each step of the path building process, if either:

- a locally stored certificate is a suitable issuer of the certificate to be verified. If such a certificate exists, use it as the issuer instead of proceeding in the path building process by searching in the pushed certificate chain from the server
- a locally stored certificate has the same public key of the current certificate to be verified. If such a certificate exists, use its issuer from the local repository as the next step in the path building process instead of searching for the issuer in the pushed certificate chain from the server.

In other words, if path checking fails, a second attempt should be made by letting the locally stored certificates to take precedence over those pushed by the peer. By using these simple changes in the path building process of OpenSSL and Mozilla based applications our model is fully supported also for SSL/TLS channel setup.

5. INTEGRATING PRQP AND OUR HYBRID TRUST MODEL

The first interesting feature provided by this trust model is that *trust is locally managed by simply issuing or revoking “special” cross-certificates*. This approach saves users from having to perform security checks each time a new certificate is to be added to the application’s repository. By using the proposed model the verification of public-keys (and extCAs details) can be performed by CA managers (or experts in PKI policy verification and auditing) when issuing the cross-certificate in a reasonably and reliable secure way (e.g. all needed steps to verify details about the CA certificate to be cross-certificated will be checked).

The second attractive feature of this solution is that no extCA_{1...n} original certificate is needed on the application to verify the certificates chain. Thus it is possible to provide trust into applications by locally storing only certificates from one trusted organization which provides the needed TA in the application’s local store. This hybrid trust model could productively be used together with other Trust-Related projects.

An attempt to decouple the trust list from the applications is the TACAR (TERENA Academic CA Repository) project [21], started at the end of 2003. It aims to provide a trusted repository to hold the appropriate root CA certificates needed by applications. The collected certificates are those directly managed by the member National Research Networks (NRENs), or belonging to a national academic PKI, or to non-profit research projects. As in our solution, this project aims to solve the cross-domain usage nightmare of PKI. In particular, the TACAR project provides a certified process for gathering and verifying root-CA certificates, and publishes them in one easily downloadable and importable trusted file. Therefore, as detailed in the next subsection of this paper, by integrating our hybrid trust model with the TACAR repository as a trusted source of root-CA certificates, it could be possible to enable inter-domain verification of all TACAR’s provided certificates.

5.1 Distributing hybrid certificates

One issue that we have not yet addressed in this paper is how to provide the applications with the needed special cross-certificates. In fact, applications need to know that a cross-certificate for that particular rootCA or subCA exists in order to correctly build the verification path up to the trusted anchor. Since recently, there was no interoperable solution to dynamically provide applications with references (URLs) to certificate bundles. We considered several possibilities when trying to address this issue. For example some sort of automatic update system could be implemented to gather the special cross-certificates from a trusted source. Such mechanisms are already in place for updating many modern operating systems, therefore a similar solution could be adopted in some environments. Because we want to provide a generic and interoperable solution across operating systems and applications, we decided to adopt a different approach.

In particular, in our solution we used a PRQP server (i.e., an RQA) to distribute locators (URLs) to sets of certificates endorsed by CAs. In order to do that, we needed to extend the current specifications of the PRQP protocol. To identify the packet of endorsed CAs, we specified a new Object Identifier (OIDs) as follows:

```
id-ad-prqp-p7endorsedTA ::= { id-ad-prqp 100 }
```

which we used to identify the locator for a PKCS#7 signed object. This object contains the set of certificates that are endorsed by a particular CA and it should be signed by the CA directly.

In our infrastructure, when an application needs to import the set of TAs endorsed by a specific CA, it queries the RQA that carries the configuration for the specific CA. In the case of a single organization, the location of the RQA can be provided via DHCP or via DNS SRV records as specified in the current PRQP draft. Other configurations based on Peer-to-peer technologies [13] are also possible.

When the `id-ad-prqp-p7endorsedTA` OID is present in the PRQP request, the RQA will provide the client application with one (or more) pointer(s) to the available packages of endorsed CAs. The client application will then proceed by retrieving the PKCS#7 object from the URL received from the RQA. After verifying the signature on the PKCS#7 object, the application can safely import the list of TAs included in the retrieved object. As all the certificates present in the PKCS#7 object are issued by one single CA (the one that signed the data object), only a single trust decision is required from the user. In the case that the issuing CA is already trusted by the application—eg., the user’s organization CA certificate or the application’s vendor certificate—no interaction with the user is actually required. However, we do recommend that a simple dialog be presented to the user the first time the TA package signed by a CA is actually downloaded from a URL.

6. CONCLUSIONS AND FUTURE WORK

In this paper we described how Trust Anchor Management is a central point for PKI usability. We also explained some of the current standardization activities within IETF and in particular we discussed the possibilities offered by combining PRQP and TAMP. We then focused the central part of the paper on how it is possible to enhance the usability of PKIs from a different point of view. In particular, we advocate that by adopting a more dynamic approach to PKI services—in particular within browsers—both PKI and Application vendors can benefit from easier to use services and more flexible infrastructure. In particular we detailed different approaches to link certification structures, differing in their security properties, their scalability, their management requirements, their implications on path construction and validation, and their dependencies on directory services. Tests show that the hybrid model, which combines the flexibility of cross-certification with the ease of deployment typical of the hierarchical trust model, is supported by many available applications. The solution provided in this paper addresses also trust management issues by using the hybrid trust model to reduce the number of trust anchors needed by applications to just one. Still further investigation is required to better understand possibilities provided by the usage of this model. In particular our efforts are directed to integrate our trust management system with existing realities (e.g. TACAR) to provide practical support in real life environments. We are currently in the process of deploying RQA servers for all of the CAs participating in the TACAR project. We believe that operational experience will be important in validating the usability of this approach and its adoption in specific environments (e.g. research networks and grid communities) where simple trust management of certificates and linking of isolated PKIs is required. We also hope that our work will provide valuable feedback for the standardization of TAMP and provide a useful use-case to further promote the standardization of PRQP.

7. REFERENCES

- [1] EuroPKI Infrastructure. EuroPKI website. [Online] <http://www.europki.org>.
- [2] GSI working group of the Global Grid Forum. [Online] http://www.gridforum.org/2_SEC/GSI.htm.
- [3] The European Policy Management Authority for Grid Authentication in e-Science. [Online] <http://www.eugridpma.org/>.
- [4] The International Grid Federation. [Online] <http://www.gridpma.org/>.
- [5] Alma Whitten and J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *8th USENIX Security Symposium*, August 1999.
- [6] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.
- [7] Denise Anthony, James Kitts, Chris Masone, and Sean W. Smith. Technology and Trust. In *Eastern Sociological Society Annual Meetings*, Feb 2008.
- [8] T. Dierks and C. Allen. The TLS Protocol. Internet Engineering Task Force: RFC 2246, January 1999.
- [9] ISO/TC68/SC2. Certificate management for financial services – Part 1: Public key certificates. ISO 15782-1:2003, August 2003.
- [10] Kelvin Yiu. 6th Annual PKI R&D Workshop, “Applications Driven PKI (It’s The Apps, Stupid!)”, April 2007.
- [11] Massimiliano Pala. PKI Resource Query Protocol (PRQP). Internet-Draft, Experimental Track, June 2008.
- [12] Massimiliano Pala and Sean W. Smith. AutoPKI: A PKI Resources Discovery System. In *Public Key Infrastructure, 4th European PKI Workshop: Theory and Practice, EuroPKI 2007*, volume 4582. LLNCS, Springer-Verlag, June 2007.
- [13] Massimiliano Pala and Sean W. Smith. PEACHES and Peers. In *5th European PKI Workshop: Theory and Practice*, volume 5057, pages 223–238. Lecture Notes in Computer Science, Springer Verlag, June EuroPKI 2008.
- [14] Massimiliano Pala, Marius Marian, Natalia Moltchanova, Antonio Liroy. PKI past, present and future. *International Journal on Information Security*, 5:18–29, January 2006.
- [15] M. Pala, A. Liroy, M. Marian, and N. Moltchanova. The EuroPKI Experience. In *Proceedings of the 1st European Workshop on PKI*, volume 3093, pages 14–27, Berlin, Germany, June 2004. Springer-Verlag.
- [16] R. Guida, R. Stahl, T. Bunt, G. Secrest, J. Moorcones. Deploying and Using Public Key Technology: Lessons Learned in Real Life. *IEEE Security and Privacy*, pages 67–71, September 2004.
- [17] R. Reddy, C. Wallace. Trust Anchor Management Requirements. Internet Draft: Informational, October 2008.
- [18] R. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21 (2):120–126, 1978.
- [19] Sean W. Smith. A Funny Thing Happened on the Way to the Marketplace. *IEEE Security and Privacy*, 1 (6):74–78, November/December 2003.
- [20] Simson L. Garfinkel and Robert C. Miller. Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24, 2005.
- [21] TACAR Project. TERENA Academic CA Repository. [Online] <http://www.tacar.org>.
- [22] W. Diffie, M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22 N.6:644–654, November 1976.