J Internet Serv Appl (2010) 1: 83–93
DOI 10.1007/s13174-010-0009-4

ORIGINAL PAPER

A search engine for the global PKI

Paul Rabinovich

Received: 24 December 2009 / Accepted: 24 June 2010 / Published online: 27 July 2010

Abstract Today the public key technology enjoys wide acceptance and use. Countless network protocols and applications use it to guarantee strong authentication and privacy. Usability and maintainability of this technology remains problematic, however. It is still very cumbersome and time-consuming to set up an enterprise Public Key Infrastructure (PKI) that has relationships with external parties. The emergence of PKI bridges, while solving one set of problems, created a new one: management of distributed trust became much more difficult. Complexity of the global PKI mesh and its decentralized nature created a need for a service with a unified view of the global PKI. In this paper we propose a PKI search engine that can provide such a service. The engine supports facilities for certificate and certificate revocation list (CRL) discovery, testing and troubleshooting of extra-enterprise PKIs, certificate revocation status lookup, certification path construction and validation, all based on the Internet-mined and user-registered information.

Keywords Public Key Infrastructure · PKI mesh · X.509 certificate · Certification authority · Certification path discovery and validation · Certificate discovery

1 Introduction

Adams and Just attribute the beginning of the modern Public Key Infrastructure (PKI) to late 1993–early 1994 [2]. This is when the recently published 1993 version of the ITU-T Recommendation X.509 [39] began to be implemented on a small scale. This version formulated some of the most important approaches to certificates, certification authorities (CAs) and related concepts.

Since then the public key technology proved its viability and dynamism. X.509-based PKI is widely used in numerous network protocols: IPSec [9], 802.1x [13], PKINIT [35], VoIP [30], SSL/TLS [27], S/MIME [25], SAML [6], and many, many others. It is an enabling technology for such important and diverse applications as document signing, Web server security, secure e-mail, Web Services security, virtual private networks, electronic commerce, and single sign-on [10].

X.509-based PKI was conceived as an authentication infrastructure for the global X.500 directory. As the X.500 project lost its momentum, the vision of a single global Public Key Infrastructure gave way to the reality of numerous independent, disjoint hierarchical PKIs [1]. This quickly proved insufficient: for example, the recipient of a signed e-mail message must belong to the same PKI as the sender or must explicitly trust the sender's CA. As business-to-business applications became widespread, organizations started to link their PKIs, first by establishing peer-to-peer relationships between certification authorities, and then by creating PKI bridges that connect multiple parties into a single PKI. This PKI mesh is more complex than a hierarchical PKI: certification path construction is more involved and becomes non-deterministic in a general case [24].[1]

Figure 1 shows a portion of the global PKI containing three bridges in the United States: Federal Bridge operated by the U.S. Government, the pharmaceutical industry

P. Rabinovich (✉)
Exostar LLC, Herndon, VA, USA
e-mail: paul.rabinovich@exostar.com

[1] Identifying all certification paths in the PKI mesh is only possible through exhaustive search which is not practical. Most relying parties use heuristic algorithms to narrow down the search space, and different algorithms may potentially produce different certification paths—unless the search process itself is standardized. See, for example, RFC 4158 [8].

bridge SAFE, and CertiPath, the aerospace and defense industry bridge. Although fairly complex, Fig. 1 provides only a high-level—and, thus, simplified—view of the PKI; many details are not shown:

- The bridges operate more than one certification authority; some of them are cross-certified with the "main" CA.
- Each leaf node represents an intra-enterprise hierarchical PKI with multiple CAs in it.
- Some trust relationships are unidirectional, others are bidirectional.
- Each link is established at one or more assurance levels reflected in the `certificatePolicy` extensions of the corresponding cross-certificates. These have to be taken into account during certification path construction: what appears to be a transitive trust relationship in Fig. 1 may be broken because of assurance level mismatch.
- Additional restrictions such as name and certification path length constraints may further complicate prediction of which certificates are acceptable to a relying party and which are not.

The global PKI is complex and, as more participants join in, it will become even more so. Discovery of certificates suitable for a particular set of security requirements, certification path discovery and validation, design, testing and troubleshooting of externally-facing enterprise PKIs, and other PKI-related management functions are time-consuming and costly. On the other hand, much of the information needed to perform these activities is already available on the Internet. Oftentimes, however, this information is difficult to locate and evaluate.

This paper proposes a service aimed at simplifying the use and management of the global PKI. Just like Google acts a single index for the global hyperlinked document corpus available through the Web, this service can act as a single index for the global PKI. PKI objects, e.g., X.509 certificates and certificate revocation lists, found in various repositories on the Internet are themselves examples of hyperlinked documents (in many cases they contain URLs) providing additional means for resource discovery and analysis.

This paper is organized as follows. Section 2 makes the case for a global PKI search engine. In Sect. 3 we enumerate the services the proposed PKI engine will provide. Section 4 discusses the engine's architecture. Supported PKI objects, PKI object-containing resources on the Internet and protocols for their retrieval are discussed in Sect. 5. Section 6 covers the non-functional aspects of the proposed service: scalability, availability and trust. Finally, Sect. 7 discusses related work, and Sect. 8 summarizes the paper.

2 The case for a PKI search engine

In this section we discuss various PKI-related functions performed by end users and administrators, and show how a PKI search engine can make their job easier.

2.1 End-entity certificate discovery

Public key cryptography provides two primary services to communicating parties: authenticity and integrity, on the one hand, and privacy, on the other. If the sender of a message wants to assure its recipient(s) of its authenticity and/or integrity, she signs the message often attaching her public key (in the form of a digital certificate) to the message itself. Message privacy, however, requires an out-of-band process of discovery of the recipients' public keys. The S/MIME standard provides one means for such discovery: the potential recipient sends a signed e-mail message to the potential sender attaching his X.509 encryption certificate to it [25]. This mechanism works if the sender can construct a certification path from the recipient's certificate to its trust anchor but this is not assured by any means. For example, the sender may send the first encrypted message long after receiving the certificate, and the certificate may expire or be revoked by that time. The sender may not even trust the certificate because it is not issued by a trusted CA or because additional constraints imposed by the sender's enterprise security policies invalidate the certification path. The recipient may have multiple certificates; sending the correct one would have solved the problem but how to decide which one to send? In addition, the S/MIME mechanism is application-specific (e-mail) and will not work with other applications such as instant messaging, Web conferencing, etc.

Some mail agents such as Microsoft Outlook can be configured to look up recipients' certificates on an LDAP server [4]. The Open Group S/MIME Secure Messaging Architecture [46] goes a step further and defines an LDAP-based certificate lookup proxy that, given a recipient's e-mail address, locates the correct certificate repository (in a list of known repositories) and fetches an encryption certificate from it if one exists. These solutions are sufficient when the set of all possible certificate repositories of interest is relatively small and static but do not scale as the list of potential e-mail recipients grows.

In contrast, PGP uses a centralized model of certificate distribution: one or more worldwide repositories store all end-user certificates where they can be easily found [45].[2] Because the X.509-based PKI is inherently distributed, this approach is not directly applicable to it. But a notion of an "all-knowing" centralized entity, possibly virtual, is nevertheless attractive.

[2]Note that PGP also provides a key distribution mechanism similar to that of S/MIME: encryption keys can be attached to e-mail messages for future use [11].

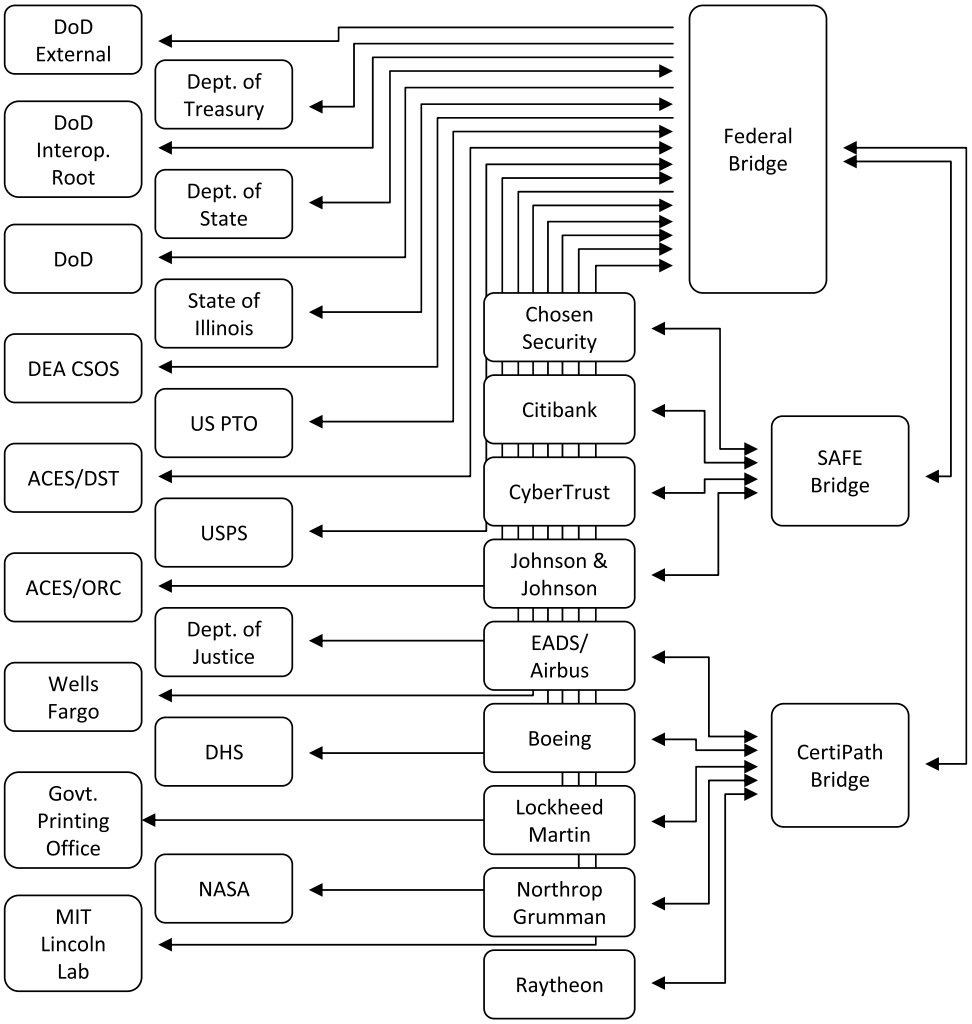

Fig. 1 A portion of the global PKI showing three U.S. PKI bridges [36, 37, 41]. See also [34]

2.2 CA certificate discovery

A relying party presented with an end-entity certificate needs to construct a certification path from that certificate to one of its trust anchors. This requires discovery and validation of intermediate CA certificates including that of the end-entity certificate's issuing CA. Modern PKIs usually require CAs to include in each certificate one or more pointers to locations where certificates issued to them (or by them) can be found. RFC 5280 stipulates that conforming CAs may include the `Authority Information Access` (AIA) extension with the access method `id-ad-caIssuers` referencing single certificates or repositories of certificates issued to them [7]. This provides a means for automatic certification path discovery.[3]

Unfortunately, many certification authorities do not populate these extensions [34]. Some of the most popular relying party software cannot process them, either. For example, the Apache Web Server which in January 2008 was running 44.8% of the world's 794,008 secure Web sites [42] does not use the AIA extension in its certification path construction algorithm.[4]

Operational constraints may also preclude dynamic path discovery by relying parties: access to the Internet may be unavailable due to lack of coverage (e.g., for traveling users) or because of restrictive security policies (e.g., at secure facilities). Sometimes network access must be carefully structured and "rationed" to reduce the attack surface of a system: for example, an airplane-based relying party may not want to download intermediate CA certificates in real time since it may expose potential security vulnerabilities more frequently [28].

[3]The AIA extension is useful when constructing certification paths in the forward direction, from an end-entity certificate to a trust anchor. A similar extension is defined to facilitate path construction in the reverse direction [7].

[4]Apache uses OpenSSL for SSL/TLS support. OpenSSL started to use the AIA extension only recently, in January 2009, in version 0.9.8j [43].

In such environments preloading of certificates issued to intermediate CAs is required [18]. Locating these certificates, especially when PKI bridges are involved, may not be trivial. A readily available Web-accessible service should simplify this task for PKI administrators.

2.3 Certificate revocation status checking

The standard certificate validation process always requires the checking of the certificate's revocation status. The revocation status of a certificate is checked either through a lookup in a certificate revocation list (CRL) [39] or a request to an Online Certificate Status Protocol (OCSP) responder [21]. OCSP was introduced in response to perceived drawbacks of the CRL mechanism [1]. One of the drawbacks is that CRLs may potentially become very large: CRLs of some well-known CAs have already achieved several megabytes and continue to grow [29, 32]. This may become prohibitive for bandwidth-constrained devices.

The standard OCSP model allows a third party to provide a service that translates CRLs into OCSP responses [1]. The PKI search engine we propose could expand this model by using a "mega-CRL" backend constructed from CRLs harvested from known certification authorities around the world (Fig. 2).

While OCSP is limited to checking only a certificate's revocation status, a more general-purpose protocol has recently been standardized by the IETF: the Server-based Certificate Validation Protocol (SCVP) [12]. The protocol allows relying parties to delegate certification path construction and validation to a third party. The third party (sometimes called a validation authority) may use certificates it receives in an SCVP request as well as those preloaded into its own database to develop and validate certification paths for its clients. As with OCSP, the PKI search engine could act as an SCVP validation authority whose database of certificates is continually updated from all available resources on the Internet.

2.4 Trust management

The targeted set of users for an application is defined by a combination of business and security needs. This set is denoted U_T in Fig. 3. If the application requires certificate-based authentication, U_T can be unambiguously mapped into the set C_T of acceptable end-user certificates from the global PKI. At the technical level, five elements define the set of end-entity certificates acceptable to a relying party:

1. The topology of the global PKI mesh.
2. The choice of trust anchors.
3. The set of cross-certificates issued by the trust anchor CAs to external CAs, and the validity-constraining extensions (such as certificate policy identifiers, certificate policy mappings, and basic, policy and name constraints [7]) placed in those certificates.
4. The validity-constraining extensions placed in the intermediate CA certificates on every certification path from C_T to the trust anchors.
5. The set of input parameters to the certification path validation algorithm (e.g., name constraints and acceptable certificate policy identifiers [7]).

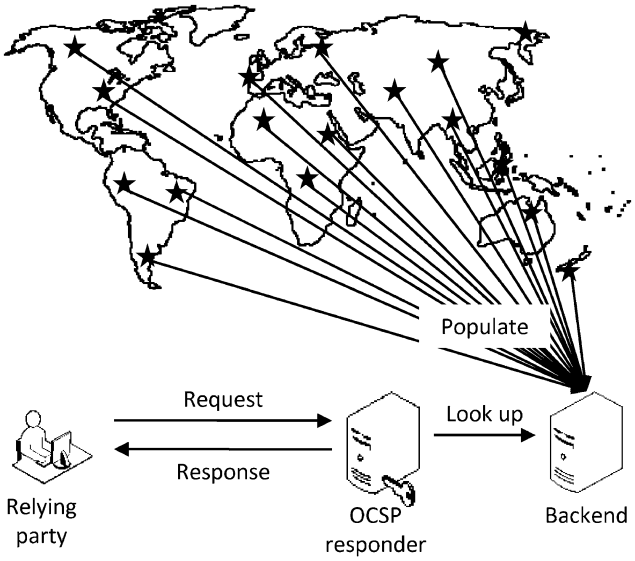

Fig. 2 Support for online certificate status checking in the proposed PKI search engine

A PKI administrator has a number of controls at his disposal to limit the set of end-entity certificates trusted by his relying party. For example, if all targeted end users come from the same organization, he may specify the organization's e-mail domain name in the `initial-permitted-subtrees` (name constraint) input parameter to the certification path validation algorithm [7]. If it is known that all acceptable certificates lie no further than one bridge away from his chosen trust anchor in the global PKI mesh, he may place the policy constraint `inhibitPolicyMapping` [7] with the value of 1 in the cross-certificate issued by that trust anchor CA. If all acceptable certificates are required to have the `Extended Key Usage` set to a particular value, he may specify it as a non-standard certification path validation input parameter if his relying party's software supports it [48].

Of these, only the second, third and fifth elements are under direct control of the relying party's organization. The goal of the PKI administrator is to carefully design and implement these elements so that the actual set C_A of acceptable certificates is as close to the set C_T (Fig. 3) as possible. Possession of a valid certificate (and the corresponding private key) is a necessary but not sufficient condition for gaining access to an application: an authorization step following

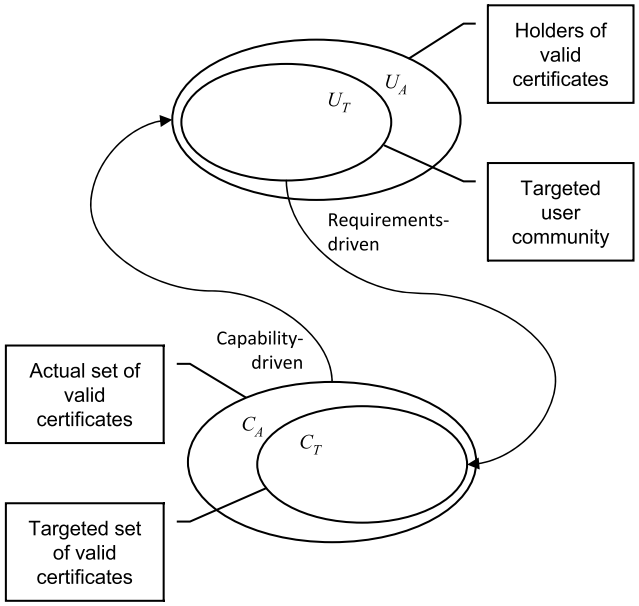

Fig. 3 The targeted and actual sets of end-user certificates acceptable to an application

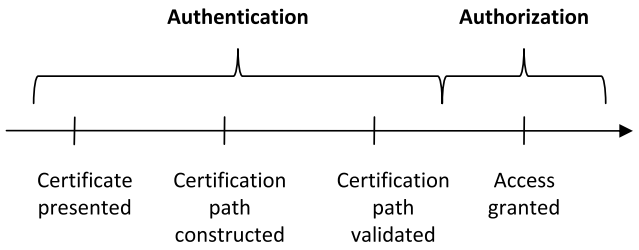

Fig. 4 Certificate-based authentication followed by authorization by a relying party

authentication ultimately establishes whether the user can proceed or must be stopped (Fig. 4). The authorization step will be responsible for granting access to the holders of certificates in the set C_T and denying it to the holders of certificates in the set $C_A \setminus C_T$.

Many other examples can be constructed. Indeed, the number of available controls may be quite large, and multiple options may exist to achieve the same outcome. In addition, the PKI administrator must have intimate knowledge of the relevant portions of the global PKI. For instance, the e-mail domain-based name constraint in the example above will be effective only if all end-user certificates reachable in the PKI mesh carry their holders' e-mail addresses in the `Subject Alternative Name` extension; if not, the name constraint will have no effect. The `inhibitPolicyMapping` constraint from the second example will preclude the relying party from accepting potentially valid certificates from a hierarchical PKI (across the bridge) that uses policy mappings, an unusual but not illegal situation. Finally, the global PKI mesh is work in progress: the set C_A optimal at a particular time may drift and include more end-user certificates as new CAs and certificates become part of the mesh.

These examples show that trust management for a relying party requires an approach highly tailored to the environment for which it is performed, and is likely to remain so. A PKI search engine, however, could provide visualization and analysis tools aimed at simplifying this task for PKI administrators.

2.5 Trust anchor management

Trust anchors are typically represented as self-signed certificates. Their distribution is managed through out-of-band mechanisms. Confidence in the integrity of a trust anchor is also established via an out-of-band mechanism, for example, by checking its fingerprint. Routine trust anchor re-key operations typically require similar out-of-band checks [26]. Manual fingerprint verification proved to be feasible for occasional use, but it does not scale well [31].

A reputable centralized service could provide self-signed certificates and verified fingerprints to enterprise administrators worldwide. By polling known sites and certificate repositories it should be able to quickly detect new self-signed certificates and make them available to its users for download.

Since trust anchors form a foundation of trust in an extra-enterprise PKI, some administrators may be reluctant to load self-signed certificates from a third party, however reputable. For them the PKI search engine can provide point of contact information for the respective CAs. Preliminary work and testing can be done using self-signed certificates downloaded from the engine, and production trust anchors can be established through direct contacts with the CAs' operating authorities.

2.6 Troubleshooting and testing

Troubleshooting of the path construction and validation process in PKI meshes remains a difficult and time-consuming task [24]. Usually testers have full information only about their part of the PKI treating the rest of the trust fabric as a "cloud." A global PKI where CAs possess multiple keys and support multiple levels of assurance, where PKI bridges are a norm and where multiple certification paths of varying quality can be constructed, makes testing difficult if only partial information is available.

A centralized service with a unified view of the global PKI mesh can significantly simplify testing and troubleshooting. The service should be able to provide on-demand visualization of the PKI graph and support filtering and zooming capabilities. The service should also be able to construct and present certification paths for end-entity certificates based on user-entered criteria.

3 Services for end users

The PKI search engine will provide a Web-based interface for (human) end users. The interface will include search and retrieval functions as well as functions for posting information to the search engine. End users will be able to register certification authorities by providing their self-signed certificates and point-of-contact information. Optionally, registration of certificate policy and certification practice statement documents will be supported. The PKI search engine will allow users to provide references to PKI objects, objects potentially containing PKI objects, and repositories with such objects. These references can be used to discover new resources not referenced by any of the resources already available to the search engine. Users will also be able to directly upload PKI objects (e.g., X.509 certificates) and objects potentially containing PKI objects (e.g., LDIF files, PKCS #7 objects, etc.).

Many applications and protocols, whose purpose is not PKI-related, exchange data that, nevertheless, contain PKI-related information (see Sect. 1). These data can be captured and mined for PKI objects. For example, S/MIME messages may have signing and encryption certificates attached [25]. An XML document may contain a signature block carrying a `KeyInfo` structure with public keys, X.509 certificates, or references to them [49]. Logs of various protocol entities using PKI-based authentication or encryption may also provide wealth of PKI-related information (public keys, X.509 certificates, key identifiers, issuer and subject names, certificate serial numbers, etc.) [38]. The engine's frontend will allow users to upload these objects as well. Since these objects represent real operational data, they may contain sensitive information. The engine is only interested in already existing, public PKI objects, however; this additional information is not relevant to its work and can be safely removed on the client. Depending on the type of the raw data, it will be done via scripts (e.g., JavaScript), applets (e.g., Java), or browser extensions.

Search and retrieval functionality supported by the PKI search engine will allow users to find information about certification authorities by performing exact and approximate search by the CAs' distinguished name, key identifier, public key and certificate policy identifiers, and search for X.509 certificates and CRLs by providing subject names, subject alternative names (such as e-mail addresses, URLs, DNS names, IP addresses), key identifiers, public keys, key usage, validity dates, certificate policy identifiers and other parameters as search criteria.

The PKI search engine will provide a rich set of visualization tools to end users. It will be able to display various connectivity components of the global PKI mesh, layers (for example, based on assurance level), hierarchical PKIs and so on, based on user-entered criteria.

A number of services for programmatic access ought to be provided as well. To support mail agents relying on LDAP-based repositories for discovery of end-user encryption certificates (see Sect. 2.1), it will provide an LDAP interface to perform lookups based on the recipients' e-mail addresses and other attributes. The engine will support the HTTP-based certificate store access protocol defined in RFC 4387 [15]. That protocol supports certificate lookup operations based on several types of certificate hashes, key identifiers, and various subject name forms.

The PKI search engine will also host an OCSP responder [21] and an SCVP validation authority [12] to provide revocation status checking, and certification path development and validation services, respectively.[5] The engine will also host X-KISS `Locate` and `Validate` services [16] to allow clients to find X.509v3 certificates and public keys, and to validate X.509v3 certificates.[6]

4 Architecture

The architecture for the proposed PKI search engine is shown in Figs. 5 (backend) and 6 (frontend). The system's core is *Store* whose primary role is to act as a central repository of PKI objects. *Store* will also contain housekeeping information for the various components of the search engine: location identifiers (for example, Universal Resource Identifiers), metainformation about objects, processing histories, information about sites, etc. In addition, *Store* will contain information about certification authorities (CAs) and their points of contact.

PKI objects are populated into *Store* by *Content Manager*. *Content Manager's* responsibility is to take any objects found by other components and extract PKI objects from them. For example, an LDIF file representing a dump of an LDAP directory may contain certificates and CRLs, a PKCS #7 object—one or more X.509 certificates, a SOAP message—references to certificates or public keys, etc. *Content Manager* will be designed as a pluggable framework to accommodate new types of objects in the future.

Locator Manager will be responsible for supplying *Store* with locators. A locator is an identifier of a resource on the Internet. It may be a URL, International Resource Identifier (IRI), Extensible Resource Identifier (XRI), or any other established type of locator, but may also be of a proprietary type if a convenient format for a particular use has not been

[5] Since SCVP is not widely supported by relying party software, the engine will provide an SCVP proxy that accepts forms and produces results in the HTML format; the proxy will communicate with a true SCVP-based validation authority internally.

[6] The X-KISS protocol also supports PGP- and SPKI-based public key infrastructures which, for the moment, are outside the scope of the proposed PKI search engine.

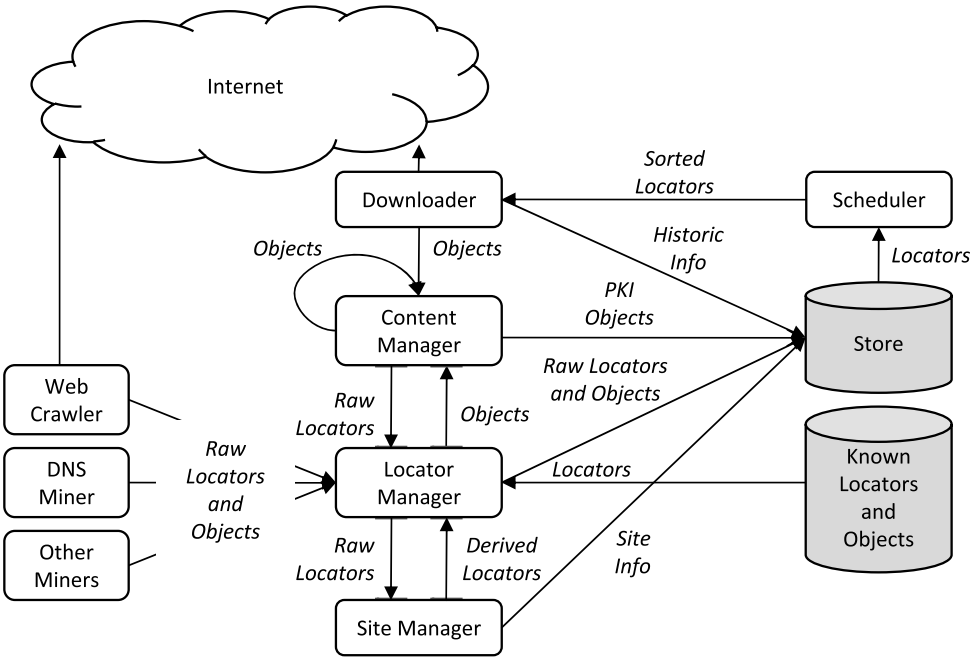

Fig. 5 Backend architecture for the PKI search engine

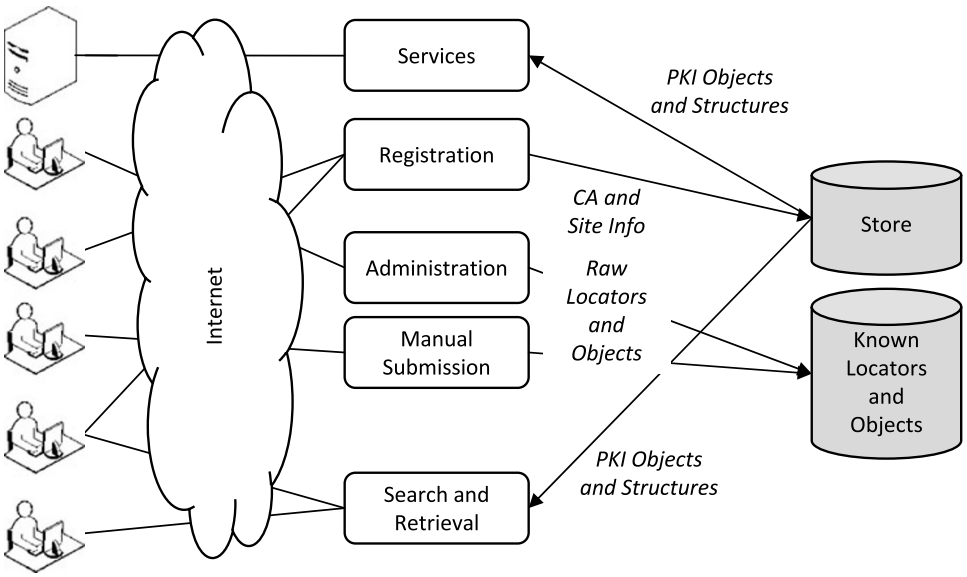

Fig. 6 Frontend architecture for the PKI search engine

established by the industry. *Locator Manager* receives locators from:

- *Internet Miner*s shown on the left in Fig. 5.
- *Content Manager* that may discover locators as part of PKI objects and other objects.
- *Known Locators and Objects Store* populated either by the search engine administrators or end users.
- *Site Manager* that, given a locator, may derive additional locators related to the same site.

An auxiliary role performed by *Locator Manager* will be to supply raw locators to *Site Manager* for further analysis. *Site Manager* will be responsible for identifying Internet sites from locators supplied to it by other components and managing site information in *Store*. It will also supply derived locators to *Locator Manager* for further processing.

Downloader will be responsible for downloading objects from the Internet given their locators. The objects will be passed to *Content Manager* for processing. *Downloader* will be fed by *Downloader's* queue which will contain a list of object locators prioritized for downloading. *Downloader* may be implemented as a multi-threaded or a distributed component; whatever the implementation, it will respect priorities assigned by *Scheduler* whose role will be to analyze the known locators and objects in *Store* and decide on the relative deadlines for their updates.

*Internet Miner*s will be responsible for mining the Internet and supplying the search engine with potentially interesting objects. For example, a Web crawler may discover objects (PKCS #7 files, certificates, CRLs, etc.) referenced by Web pages. The search engine may use its own Web crawler and/or use services of external Web crawlers such as those maintained by established search engines (Google, Yahoo!, Bing, A9, etc.). *DNS Miner* will be responsible for discovering DNS domains and probing them for PKI-related information stored, for example, in CERT or SRV resource records (see Sect. 5). Other *Internet Miner*s may be added to the search engine in the future if needed.

Several interactive applications will be part of the search engine (Fig. 6). First, *Search and Retrieval Application* will allow users to search for end-entity and CA public keys, public certificates and CRLs. *Search and Retrieval Application* will also be able to display the global PKI network, its various connectivity components, layers, etc. In addition, *Search and Retrieval Application* will support lookup of point-of-contact information for registered CAs. *Manual Submission Application* will allow users to "manually" submit locators and objects. The objects may be PKI objects or composite objects requiring further parsing and object extraction. *Registration Application* will allow users to register certification authorities (CAs) and their points of contact by providing the CAs' self-signed certificates and filling out forms containing information about the CAs' administrators and managers. It will also allow site registration. *Administration Application* will be provided for administrators to manage all aspects of the search engine.

In addition to applications for human users, the frontend will provide a variety of services for programmatic access as discussed in Sect. 3. All of them are represented by the *Services* component in Fig. 6.

5 Supported objects and repositories

The fundamental objects the PKI search engine must support are (a) public keys, (b) public X.509 version 3 certificates [7, 39], and (c) X.509 version 2 certificate revocation lists (CRLs) [7, 39]. Certificates and CRLs, of course, serve as containers for public keys that can be extracted from them for storage and further processing. They can also be viewed as hyperlinked documents carrying references to PKI repositories in such extensions as `Authority Information Access`, `Subject Information Access`, and `(Freshest) CRL Distribution Points` [7]. In addition, the `Certificate Policies` extension (in a CA certificate) may contain pointers to the respective certification policy statements and notices that can be extracted and presented to the PKI engine's users as part of metainformation about the CA.

Table 1 PKI-related LDAP attributes [33]

Attribute name	Description
`authorityRevocationList`	A CRL reflecting revocation status of certificates issued to certification authorities
`cACertificate`	A CA certificate
`certificateRevocationList`	A CRL
`crossCertificatePair`	A cross-certificate pair that contains a cross-certificate issued by the CA to another CA and/or a cross-certificate issued to the CA by the other CA
`deltaRevocationList`	A delta CRL containing information on certificates revoked after a certain version of the main (base) CRL or after a particular point in time [1]
`userCertificate`	An end-entity certificate

The PKI search engine will be able to process many types of composite objects that may contain PKI objects:

- PKI-related LDAP attributes enumerated in Table 1.
- LDAP Data Interchange Format (LDIF) files as defined by RFC 2849 [14]. Since an LDIF file represents a dump of a subtree of an LDAP-based directory, the PKI search engine will mine these files for RFC 4523-defined object classes and attributes (Table 1).
- Objects in the Cryptographic Message Syntax format defined by PKCS #7 [20] and RFC 3852 [17]. These objects may contain public keys, certificates, CRLs and references to them.
- DNS CERT resource records (RRs) defined by RFC 4398 [19]. This standard defines multiple types of objects stored in CERT RRs; of interest to the PKI search engine are `PKIX` records representing X.509 certificates and `IPKIX` records representing references to X.509 certificates.
- DNS SRV resource records for `_PKIXREP`, a service type defined by RFC 4386 [3]. These resource records contain pointers to LDAP- and HTTP-based PKI repositories as well as OCSP responders.

The PKI objects and composite objects potentially containing PKI objects may be found in many types of repositories on the Internet. The PKI search engine's backend will be able to mine LDAP-based repositories for these objects. It will also support DAP, the Directory Access Protocol [40]. Although it is largely superseded by LDAP, DAP-based directories are still used to store PKI-related objects.

The search engine will be able to access repositories using the FTP and HTTP protocols, for example, when given references (URLs) to single certificates, certificate revocation lists, or PKCS #7 objects. The engine will also support

the specialized HTTP-based certificate store access protocol defined in [15]. This protocol can be used to look up certificates based on client-defined criteria. Similarly, the PKI engine's backend will be able to access X-KISS `Locate` service [16] end points on the Internet to find X.509 certificates using subject names, key names, key identifiers, and public keys.

The PKI search engine's backend will also take advantage of the PKI Resource Query Protocol (PRQP) [22]. A PRQP responder acts as a directory service for PKI-related resources and may be used to locate certificates and certificate revocation lists.

Finally, the search engine will support well-known community repositories such as Trans-European Research and Education Networking Association's Academic CA Repository (TACAR) [47], possibly implementing proprietary protocols.

6 Discussion

Given the scope of the proposed PKI search engine it is expected that it will need to store large quantities of information and service a very large user community. The engine will have distributed architecture with a main site holding a writable copy of the persistent store and a number of replicated mirror sites each holding a read-only copy. End users will be directed to the mirror sites through the use of round robin DNS and load balancers. This architecture will provide high availability, scalability, and performance required by the service. Each copy of the store will be implemented using a commercial relational database management system (RDBMS) such as Oracle Database or Microsoft SQL Server. Commercial state-of-the-industry hardware with virtualized environments running the Linux operating system will be used to host all components of the search engine. Most of the software will be developed in Java; performance-sensitive modules will be written in C and C++.

To ensure availability, best practices in IT management and business continuity management will be followed by the service providers hosting the engine's components. Special attention will also be given to security. Appropriate technical, physical, procedural, and personnel controls will be put in place to guarantee continuous operation of the engine. Safe Web application development guidelines such as those provided by the Open Web Application Security Project (OWASP) [44] will be followed when developing the engine's frontend (Fig. 6). In addition to sound development and management practices, the engine's distributed architecture should help mitigate against security compromises, including denial of service attacks. The main site will not be accessible from the Web, and all replication links will operate in one direction, from the main site to the mirrors. Each link will be secured using an IPSec-based virtual private network (VPN) [9]. Requests coming through the frontend will not be able to modify the mirror stores. Because of the built-in redundancy the engine should be able to continue to operate even when some of the mirror sites are under attack.

Even if the PKI search engine is known to be secure, why should end users trust information they receive from it? What level of assurance can be given by its certificate validation services? We expect that enterprises (private corporations, governments, military institutions) will continue building their own certificate validation infrastructures and will use the proposed engine only when designing, implementing, testing and troubleshooting them. They may, however, encourage continual use of our engine's discovery services by their end users if it is known to provide fast and accurate results. The engine can also attract the "unattached" consumer community; here its validation services may find their biggest market as in most cases consumers will seek only reasonable practical assurance that the results provided by the engine are correct. If the engine provides services as expected, over time it will build a good reputation in the user community, and more users will be willing to trust it. In addition, the engine's practices could be certified by an independent organization such as WebTrust or TRUSTe. Further, a customized certification and auditing regime by a third party could be developed for the engine's validation services, and that, in turn, may lead to the acceptance by communities with higher assurance requirements.

Privacy may also be of concern for users supplying information to our PKI engine. Although each certificate or CRL contains publicly available data, their aggregation may leak information and allow unwanted inferencing. Development of policies and algorithms that balance the engine's need to be powerful and useful, on the one hand, and the users' need for privacy, on the other, is one of the subjects of our future work.

7 Related work

Many of the standards referenced in this paper were designed with the goal of improving usability and simplifying management of Public Key Infrastructures. Some define protocols and data formats aimed at discovery of PKI-related objects and services by human users and software agents [3, 7, 15, 16, 19, 22, 33, 46]. Others govern the use of (delegated) certificate validation services [12, 16, 21]. The standards do not define the scope of PKI repositories or services they describe, and leave it to implementers. For example, a certificate repository may service a single certification authority, a hierarchical PKI, a group of related CAs, a community of users, etc. Our proposal is complementary to these standards as it defines an *operational* service implementing them in a single context, that of the global PKI.

Practical needs prompted many certification authorities, bridges, communities of interest, and others to establish well-known PKI repositories facilitating access to commonly needed resources (see, for example, [47]). The proposed PKI search engine differs from them in its scope and scale, and also in the mechanism it uses for gathering information: most of it will be automatically collected on the Internet through mining (although manual registration by end users will also be available).

Global-scale PKI services have also been proposed. Recently Pala introduced the *Public Key System* (PKS), a PKI resource discovery infrastructure based on distributed hash tables [23]. In his scheme each CA has a responder in the PKS that advertises services on its behalf (for example, certificate revocation checking); the responder's network address is computed based on the fingerprint of the CA's public key. Responders form an overlay network that supports efficient routing and node discovery. Relying parties access the PKS via local PKS servers. The PKS is different from the PKI search engine we propose in that it is CA-centric; our service, in addition to providing CA-related information, will allow to locate end-user certificates based on various search criteria and to perform relying party-centric services such as certification path construction and validation. (In the future it could use the PKS as one discovery mechanism to locate services provided by a CA given that CA's public key.) The PKS also requires clients to implement a new protocol (to access a local server); our engine, on the other hand, will rely only on existing standards.

8 Conclusions and future work

This paper proposes a PKI search engine, a service that does for the global PKI what search engines such as Google do for the Web. On the one hand, the Public Key Infrastructure is vital to proper functioning of many Internet protocols and applications such as document signing, Web server security, secure e-mail, Web Services security, virtual private networks, electronic commerce, and single sign-on [10]. On the other hand, understanding, management and troubleshooting of the global PKI remain problematic. As the global PKI mesh becomes more complex, it will present further challenges to end users and administrators alike.

By combining Internet-mining capabilities with direct voluntary registration the PKI search engine can accumulate—in a single repository—large amounts of information about the global PKI. This information can be used to provide valuable tools to end users to locate individual PKI objects (certificates, certificate revocation lists, and public keys) or PKI repositories, perform certificate validation, or visualize the global PKI mesh to better understand trust relationships that exist between an individual user or organization and the rest of the world.

The user community of the search engine may include end users of PKI, for example, ordinary users that need to send encrypted e-mail and locate public certificates of message recipients, as well as PKI administrators that need to design, implement, test, maintain, and troubleshoot their organizations' PKIs.

Our future work will focus on implementing a realistic proof of concept of the proposed PKI search engine. First, we plan to implement the backend infrastructure shown in Fig. 5. Then we will develop *DNS Miner* and a Web crawler to feed the persistent store. In parallel, we will develop rudimentary versions of *Registration Application*, *Manual Submission Application*, and *Search and Retrieval Application* (Fig. 6) with the goal of enhancing them over time to bring them to the expected level of usability and performance. Once these components are in place, we will focus on programmatic access to the service. Google's experience showed that a realistic prototype (that, among other things, allows to evaluate scalability of the product) can be built inexpensively with readily available software and hardware [5]. We plan to use this experience in our work.

Acknowledgements The author would like to thank Jeff Nigriny (CertiPath) for reviewing an earlier draft of this paper and providing valuable comments, and Vijay Takanti (Exostar) for his support in publishing this paper.

References

1. Adams C, Lloyd S (2002) Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley, Reading
2. Adams C, Just M (2004) PKI: Ten years later. In Proc of the third annual PKI R&D workshop, pp 69–84
3. Boeyen S, Hallam-Baker P (2006) Internet X.509 public key infrastructure repository locator service. RFC 4386, IETF
4. Boyce J, Sheresh B, Sheresh D (2007) Microsoft Office Outlook 2007 inside out. Microsoft Press
5. Brin S, Page L (1998) The anatomy of a large-scale hypertextual Web search engine. In Proc of the 7th int'l world wide web conference, pp 107–117
6. Cantor S, Kemp J, Philpott R, Maler E (eds) (2005) Assertions and protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS
7. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W (2005) Internet X.509 Public Key Infrastructure certificate and certificate revocation list (CRL) profile. RFC 5280, IETF
8. Cooper M, Dzambasow Y, Hesse P, Joseph S, Nicholas R (2005) Internet X.509 Public Key Infrastructure: certification path building. RFC 4158, IETF
9. Doraswamy N, Harkins D (1999) IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall, New York
10. Doyle P, Hanna S (2003) Analysis of June 2003 survey on obstacles to PKI deployment and usage. OASIS
11. Elkins M, Del Torto D, Levien R, Roessler T (2001) MIME security with OpenPGP. RFC 3156, IETF
12. Freeman T, Housley R, Malpani A, Cooper D, Polk W (2007) Server-based certificate validation protocol (SCVP). RFC 5055, IETF

13. Gast M (2005) 802.11 wireless networks: the definitive guide. O'Reilly
14. Good G (2000) The LDAP data interchange format (LDIF) technical specification. RFC 2849, IETF
15. Gutmann P (ed) (2006) Internet X.509 Public Key Infrastructure operational protocols: certificate store access via HTTP. RFC 4387, IETF
16. Hallam-Baker P, Mysore SH (2005) XML key management specification (XKMS 2.0), version 2.0. W3C Recommendation, W3 consortium
17. Housley R (2004) Cryptographic message syntax (CMS). RFC 3852, IETF
18. Jokl J, Basney J, Humphrey M (2004) Experiences using bridge CAs for grids. Technical Report YCS-2004-380, Dept of Computer Science, University of York
19. Josefsson S (2006) Storing certificates in the Domain Name System (DNS). RFC 4398, IETF
20. Kaliski BS, Kingdon KW (1997) Extensions and revisions to PKCS #7. An RSA Laboratories Technical Note, Version 1.6. RSA Data Security Inc
21. Myers M, Ankney R, Malpani A, Galperin S, Adams CX (1999) 509 Internet Public Key Infrastructure: online certificate status protocol—OCSP. RFC 2560, IETF
22. Pala M (2009) PKI resource query protocol (PRQP). Internet draft draft-pala-prqp-04, IETF
23. Pala M (2010) A proposal for collaborative Internet-scale trust infrastructures deployment: the Public Key System (PKS). In Proc of the 9th symposium on identity and trust on the Internet, pp 108–116
24. Polk WT, Hastings NE (2000) Bridge certification authorities: connecting B2B public key infrastructures. National Institute of Standards and Technology, Geithersburg
25. Ramsdell B (2004) Secure/multipurpose Internet mail extensions (S/MIME) Version 3.1 message specification. RFC 3851, IETF
26. Reddy R, Wallace C (2009) Trust anchor management requirements. Internet draft draft-ietf-pkix-ta-mgmt-reqs-04, IETF
27. Rescorla E (2001) SSL and TLS: designing and building secure systems. Addison-Wesley, Reading
28. Robinson RV, Li M, Lintelman SA, Sampigethaya K, Poovendran R, von Oheimb D, Busser J-U (2007) Impact of public key enabled applications on the operation and maintenance of commercial airplanes. In Proc of the AIAA aviation technology integration, and operations (ATIO) conference
29. Schwartz M (2005) Scale is everything for Pentagon's digital security. In: Enterprise Strategies Journal
30. Sinnreich H, Johnston AB (2006) Internet communications using SIP: delivering VoIP and multimedia services. Wiley, New York
31. Straub T (2005) Usability challenges of PKI. Ph.D. dissertation, Darmstadt Technical University
32. Walsh BM (2004) Johnson & Johnson: use of public key technology. In Proc of the summit and workshop for deploying PKI to end users in higher education
33. Zeilenga K (2006) Lightweight directory access protocol (LDAP) schema definitions for X.509 certificates. RFC 4523, IETF
34. Zhao M, Smith SW (2006) Modeling and evaluation of certification path discovery in the emerging global PKI. In Proc of the third European PKI workshop (EuroPKI), pp 16–30
35. Zhu L, Tung B (2006) Public key cryptography for initial authentication in Kerberos (PKINIT). RFC 4556, IETF
36. CertiPath LLC (2008) Cross certification ceremonies
37. Federal Public Key Infrastructure Policy Authority (2008) Entities that are cross-certified with the FBCA
38. Sun Microsystems, Inc (2004) Debugging SSL/TLS connections
39. ITU-T Recommendation X.509 (1997) Information technology—open systems interconnection—the directory: authentication framework
40. ITU-T Recommendation X.519 (2001) Information technology—open systems interconnection–the directory: protocol specifications
41. SAFE-BioPharma Association (2008) Issuers and cross certification
42. Netcraft (2008) Netcraft secure server survey, January 2008. http://www.netcraft.com/
43. The OpenSSL project (2009) OpenSSL change log
44. OWASP (2005) A guide to building secure Web applications and Web services 2:0
45. PGP Corp (2008) PGP desktop email quick start guide, version 9:8
46. The Open Group (2005) S/MIME secure messaging architecture, version 1:0
47. TACAR (2008) TACAR: TERENA academic CA repository
48. Cygnacom Solutions, Inc (2007) Webcullis configuration manual
49. W3 Consortium (2008) XML signature syntax and processing, 2nd edn. W3C recommendation