

PKI Architectures and Interoperability

A preliminary draft discussion paper for the Federal PKI TWG

Introduction

The PKIs that are currently implemented can be divided into three categories:

- hierarchically oriented
- network oriented
- browser oriented

These are illustrated in Figure 1 and their properties summarized in Table 1

Hierarchical PKI's

In a strictly hierarchical PKI all end-entities base their trust on the public key of a single “root” CA, as shown in Figure 1 (a). This key is generally stated in a “self-signed” CA certificate, in which the CA signs its own public key. The self-signed root CA certificate must be passed to all relying parties by some reliable out of band mechanism.

In such a PKI, CAs are arranged hierarchically and may issue certificates to subordinate CAs below them in the hierarchy, or to users. Any certificate may be verified by verifying the *certification path* of certificates that leads back to the root CA. Alice verifies Bob's certificate, issued by CA 4, then CA 4's certificate, issued by CA 2, and then CA 2's certificate issued by CA 1, the root, whose public key she knows;

Each CA has a single parent CA. In some implementations there is also at least some limited support for cross-certification via cross-certificate pairs at the top of the hierarchy. It is comparatively easy for end-entities to find certification paths, simply by backing up the hierarchy from each CA to it's parent until the root is reached. Names are often aligned with the PKI hierarchy.

The hierarchical PKI architecture has some advantages:

- The organizational management structure of many organizations such as the government is largely hierarchical. Trust relationships are frequently aligned with organizational structure, so it is natural to align the certification path with the organizational structure;
- The hierarchy may be aligned with hierarchical directory names;
- The certification path search strategy is straightforward;
- Important existing Federal PKI components are designed hierarchically;
- Each user has a certification path back to the root. The user can provide his path to any other user and any user can verify the path, since all users know the root's public key.

A strictly hierarchical certification path architecture also has some disadvantages:

- It is improbable that there will be a single root CA for the world PKI;
- Commercial and business trust relationships are not necessarily hierarchical;
- Compromise of the root private key is catastrophic and recovery requires the secure distribution of the new public key to every user.

Network PKI's

In contrast to the hierarchical architecture, where certification paths all start from a root CA, and where paths consist only of unidirectional CA-certificates, in a Networked PKI, illustrated in Figure 1 (b), there is no one particular CA that is trusted by all end-entities. Rather end-entities may choose to base their trust on the public key of any CA in the PKI, although usually they use the key of the CA that issued them their own Certificate.

Independent CA's then cross-certify each other (that is issue CA-certificates to each other, which are combined in a construct called a *cross-certificate pair*) The result is a general mesh network of bi-directional trust relationships between CAs. A user knows the public key of a CA near himself, generally the one that issued his certificate, and verifies the certificates of other users by verifying a certification path of certificates that leads back to that trusted CA. So, for example, Alice knows the public key of CA 3, while Bob knows the public key of CA 4. There are several certification paths that lead from Bob to Alice, but the shortest requires Alice to verify Bob's certificate, issued by CA 4, then CA 4's certificate issued by CA 5 and finally CA 5's certificate, issued by CA 3. CA 3 is Alice's CA and she trusts CA 3 and knows its public key.

The network PKI architecture has some advantages:

- It is flexible, facilitates ad hoc associations and trusted relationships, and reflects the bilateral trust relationships of business;
- A user must trust at least the CA that issued its own certificate in any PKI, and it is reasonable to make this the foundation of all trust relationships;
- CAs that are organizationally remote, but whose users work together with a high degree of trust, can be directly cross-certified under a high trust policy that is not extended to other CAs and is higher than would be practical through a long, hierarchical chain of certificates;
- It allows direct cross-certification of CAs whose users communicate frequently, reducing certification path processing load;
- Recovery from the compromise of any CA's private key requires only that the new public key (and certificates signed with the corresponding new private key) be securely distributed to the holders of certificates from that CA.

The network PKI also has at least two disadvantages:

- Certification path search strategies can be more complex;
- A user cannot provide a single certification path that is guaranteed to enable verification of his signatures by all other users of the PKI.

Browser oriented PKI's:

These are, in terms of the number of users, undoubtedly the most widely implemented. They rely on the fairly rudimentary client capabilities of the two nearly ubiquitous leading browser products. Each browser user has a local file of trusted self-signed "root" certificates. One browser, for example, now ships with an initial file of 36 trusted CA certificates. Control of that file may be managed by the individual user, or organizations may have provision for loading or managing it from a central network management server. The browsers normally are distributed with an initial set of root certificates. Users can add to this set or delete certificates from it.

The certificate path processing capability of the web browsers is currently quite limited. In general, they can process an ordered certificate list leading from the sender of a message back to a certificate signed under one of the self-signed certificates in the trusted certificate file. The browsers have a limited capability to retrieve a certificate from an LDAP directory, as requested

by the user, but no capability to automatically find an ordered, complete multi-step certification path. A few standardized extensions, including Key Usage and Basic Constraints are supported. Some private extensions for code signing and SSL may be supported by browser products. The X.509 name constraints, certificate policy and policy constraints extensions are not supported. The concept of cross-certificates is not directly supported.

The browsers can use certificates to sign, verify, encrypt and decrypt S/MIME email messages. They can use certificates to establish Secure Socket Layer (SSL) sessions. SSL is a transport level session encryption and authentication protocol that is being adopted by the IETF as the Transport Security Protocol (TSP). In an SSL session, users can, for example, submit forms to a server or receive information from a server in an encrypted, authenticated transport session. Browsers can also verify the signatures on signed code.

SSL is a client-server transport level protocol. Web server products from the browser vendors and other server vendors, then implement the SSL server functionality. A number of Certification Authority (that is certificate server) products are available that can issue client certificates to browsers and SSL server certificates to web servers. Several commercial CAs also offer certificates to browsers and servers over the Internet. LDAP products and some browser configuration management products complete the suite of currently available PKI tools for these systems.

Since certificate policies are not supported by browsers, each CA must issue certificates under a single policy. If an CA wants to distinguish between "high," "medium" and "low" assurance certificates, then three separate CAs with separate CA keys are needed, and the appropriate self-signed CA certificates must be loaded into the browser trusted certificate files. This, however, is at best an awkward solution, since no particular tools are provided in the browsers to manage separate trusted certificate files, and select one, depending on the type of transaction being evaluated. Multiple users are supported for each browser, so, as a sort of kludge, one might create several browser users, each with a different set of trusted certificates. Perhaps, then, some rough equivalent to the X.509 initial policy set might be achieved.

These tools, however, suffice to build quite useful single-enterprise scale applications, where all the users have certificates from a single CA, and where all certificates subjects can be equally trusted. They can also allow some limited hierarchical structure, if all users provide a certificate list back to a root CA, and all browsers are initialized to accept that CA. It is far from clear, however, that they are adequate for an "open" national or international PKI. The absence of direct support for cross certification and certificate policies limits the use of browsers for large-scale open PKI applications, as does their inability to automatically find certification paths.

Moreover, the initialization and management of a file of possibly numerous "trusted CA certificates" in every workstation in a sizable agency or organization is a somewhat questionable proposition, as a way to manage enterprise trust relationships. It is not clear that there is any practical way to prevent either users, or others with access to their workstations from making unauthorized alterations to this file. Certainly, the process of distributing an initial file of such self-signed certificates with browser products that are routinely downloaded over the Internet, is not a secure way to initialize the trusted keys. Moreover, with only two browser products of commercial consequence, the initial distribution certificate choices by browser vendors are bound to distort the market for open PKI services. In effect, the browser vendors choice of which self-signed certificates to include with their "out of the box" product may establish a kind of defacto CA accreditation.

The browser-oriented PKI architecture has several advantages:

- It is simple, straightforward and comparatively easy to implement;
- The end user has total control of the contents of the file of CAs he trusts;
- It would work well with a simple On-line Certificate Status Protocol (OCSP), since there is not a complex certification path to validate, rather the status request is addressed only to the CA in the trusted certificate file. (note that browsers do not now implement such a protocol);
- Certification paths are generally simple, which should speed processing.

The browser-oriented PKI architecture has several disadvantages:

- The browser user has total control of the contents of the file of CAs he trusts. It is hard to see how organizations can reliably manage this as organizations and prevent individual users from altering these files;
- The inclusion of a default list of trusted CA's in browsers as they come, out of the box, is hardly an authenticated process for initializing the known public keys upon which all else rests;
- The careful management of the potentially large file of trusted CA certificates may be too tedious and difficult for most end users;
- Little support is provided for finding certification paths;
- There is no direct support for cross-certificate pairs, limiting the role of CAs in managing trust;
- Certificate Policies are not supported, requiring that a Certification Authority entity operate several CAs, if it is to support different certificate policies and assurance levels. This in turn ads more Certificates to the browser's trusted certificate file;
- Browsers now support no automated method of verifying the status of certificates, nor of revoking them.

The most important advantage of the browser-oriented PKI is not inherent in the architecture: support for S/MIME and SSL is available out of the box, in current browser products. These browser clients are now ubiquitous. The World Wide Web and browsers are the dominant network technology today, and the main platform for implementing distributed applications. Web servers also support the browser-oriented PKI model. Any PKI application, that hopes to reach the public at large, must find this existing ubiquitous capability compelling. Moreover, web technology is now the platform of choice for many intranet applications as well. Whatever the abstract merits of the PKI architecture implemented by browsers, the reality of their pervasive availability, and their key position, makes hard to overstate the importance of this architecture.

Interoperability/Convergence

[author's note: I'm really struggling here. These are just a few sketchy, disorganized thoughts I'm throwing out to stimulate discussion. I'm not advocating anything at this point. None of this should be construed in any way as a NIST position on anything. Please give me any ideas that you have.]

It is possible that we may wind up with a "high security" Federal PKI, for many internal government purposes, that fully implements a hierarchical or network model, and use a more or less separate "low security" browser-oriented PKI in other applications and to deal with the public. What follows, however, are just some rough ideas about possible ways to minimize the division, and maximize the likelihood that a certificate could work in any of these systems.

Certificate Policies

It would not be terribly difficult to implement basic certificate policy processing (without, perhaps, policy mapping) in browsers, and it might have a significant pay-off. It should, for example make for a reduction in the number of CA certificates required in the trusted-CA file. It would also help to increase the likelihood that the self-signed certificate for a CA in a network PKI could usefully be included in the browser's trusted CA file;

Coalescence of "National" CAs

Perhaps there eventually will be only a very few large, widely recognized "national" CA's. End users will then need a certificate from one of these big CA's, if they need a widely accepted certificate. In such a model, a user might have a half dozen national CA certificates, plus one or two "local" CA certificates, in his browser's trusted certificate file. Similarly, he would have a personal certificate from one of the national CA's plus, perhaps, several local CA certificates, for particular applications.

If this coalescence takes place, then the limited certification path processing of the browser would matter less. This scheme would also probably point to either a single government-wide CA, or government contracts with one or more of the commercial national CAs. Perhaps we should decide to consciously work toward this end.

Certificate List Tools

Perhaps tools for finding and building certificate lists that browsers can process would be helpful. They could assist users to build the certificate lists needed for browser processing. A user might choose to post a certificate list back to a widely recognized CA in his repository, or include it in his S/MIME message, rather than just the certificate he got from his local CA.

Certificate Path Processing Plug-in

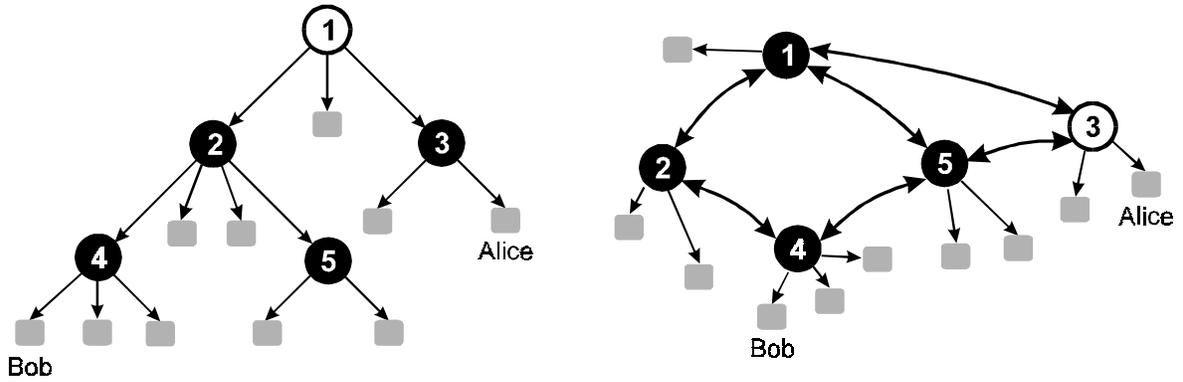
Would it be reasonable to develop a publicly available browser plug-in for fuller certificate-path processing?

Certificate (and Private Key) Management Tools

If there are to be possibly large files of trusted CA certificates in browsers, then good tools for managing them would be appropriate. Similarly, if a national PKI, unified by extensive cross-certification of CAs, and full certification-path processing in clients, is a never to be realized fantasy, then it is likely that users will need numerous certificates, and good tools to manage them, and their corresponding keys, will be important.

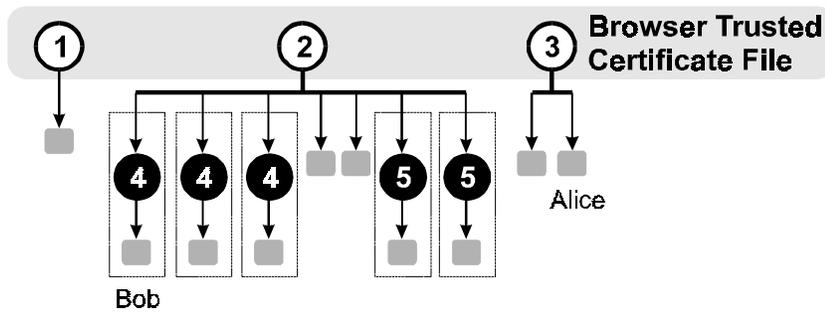
Certificate Status/Revocation

The present inability of browsers to deal with CRLs, or to provide any automated check on current certificate status, seems a nearly crippling disability, for a "serious" browser-based PKI. Should we consider encouraging support for OCSP? *[this seems to me almost to amount to abandoning the idea of a PKI unified by extensive cross-certification of CAs, and full certification path processing by clients, since I can't envision how OCSP would cope with complex certification paths].*



a. hierarchical infrastructure

b. network infrastructure



c) one browser view of infrastructure

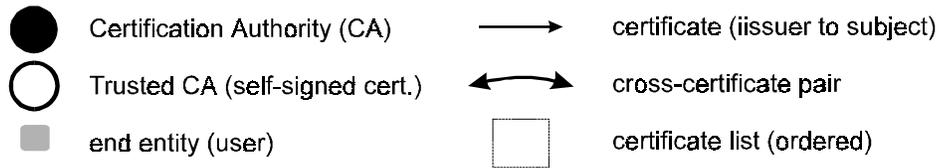


Figure 1 -Different PKI Certification Path Architectures

Table 1 - A comparison of PKI Architecture Characteristics

<i>Characteristics</i>	<i>Network</i>	<i>Hierarchical</i>	<i>Browser</i>
Trusted key(s)	CA that issued user's certificate	"root" CA	file of (usually) many trusted CA certificates in each browser
Trust paths	mesh of bi-directional cross certificate pairs	chain of parent-child certificates	pre-ordered certificate-list
Trust path finding	directory based, complex	directory based, comparatively simple	minimal; find individual certificates in LDAP directory
Cross-certification	basis of PKI	may be supported	no direct capability
Extensions			
key usage	yes	yes	yes
basic constraints	yes	yes	yes
authority key id	yes	yes	yes
subject key id	yes	yes	yes
certificate policy	yes	usually	no
policy mapping	probably needed	less need	no
name constraint	?	?	no
Certificate Status	Certificate Revocation List	Certificate Revocation List	none (may add support for On-line Certificate Status Protocol in future)