Managing Interoperability in Non-Hierarchical Public Key Infrastructures

Peter M. Hesse
Gemini Security Solutions
pmhesse@geminisecurity.com

David P. Lemire
A&N Associates
dlemire@anassoc.com

1 Introduction

This paper discusses considerations for certificate issuing systems and certificate processing applications, and directory systems in environments that employ non-hierarchical public key infrastructures (PKIs). The observations and recommendations here, while applicable to almost any non-hierarchical PKI, are most relevant to situations where the establishment of interoperability among the PKIs of disparate organizations is a primary goal. They are based on our work with a PKI interoperability testbed comprised of a bridge certification authority (CA) interconnecting multiple PKIs based on CA products from several vendors. Our view is that the more sophisticated aspects of X.509 certificate issuance and processing (e.g., certificate policies and mappings, name constraints) are tools that allow the PKI to establish the limits of security interoperability between organizations [1]. Consequently, we believe that the extensions for these X.509 features should be routinely populated by certificate issuing systems, and expected and processed by certificate processing applications. The goal of the recommendations herein is to promote interoperability among the PKI relying parties, while still allowing the owning organizations to maintain security control.

1.1 PKI Structures

This discussion is oriented toward PKIs or combinations of PKIs with non-hierarchical certificate graphs. Such a PKI may result through the implementation of a non-hierarchical mesh PKI, or the interconnection of hierarchical and non-hierarchical PKIs via cross-certification.

1.1.1 Hierarchical: A hierarchical PKI, depicted in Figure 1, is one in which all of the subscribers / relying parties trust a single "root" CA. The root CA certifies the public keys of subordinate CAs. These CAs then certify subscribers or may, in a large PKI, certify other CAs. In this architecture, certificates are issued in only one direction, and a CA never certifies another CA "superior" to itself. Typically, only one superior CA certifies each CA. Certification path construction in a hierarchical PKI is a straightforward process that simply requires the relying party to successively retrieve issuer certificates until a certificate is located that was issued by the trusted root.

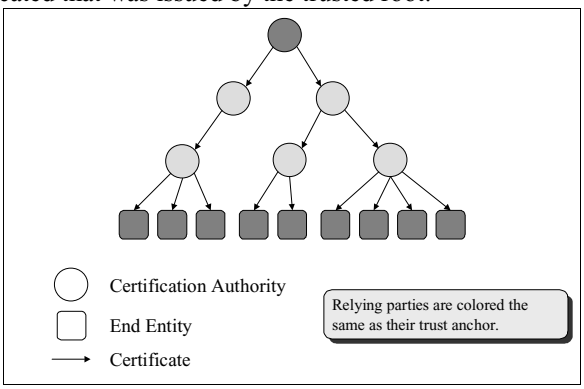

Figure 1 – Hierarchical PKI

A widely-used variation on the single-rooted hierarchical PKI is the inclusion of multiple CAs as trusted roots. Here, end entity certificates are validated using the same approach as with any hierarchical PKI. The difference is that a certificate will be accepted if it can be verified back to any of the set of trusted roots. Popular web browsers use this approach, and are shipped with trusted CA lists containing dozens of CAs. While this approach simplifies the implementation of a limited form of certificate verification, it also has numerous security weaknesses. For example, the user has little or no idea of the policies or operating practices of the various roots, and is usually unaware of which root was used to verify a given certificate. Also, the revocation of a trusted CA is nearly impossible, since it requires that every user somehow be notified of the CA's change of status. The organization must then rely on the users to take the necessary actions to remove the CA from the trust list.

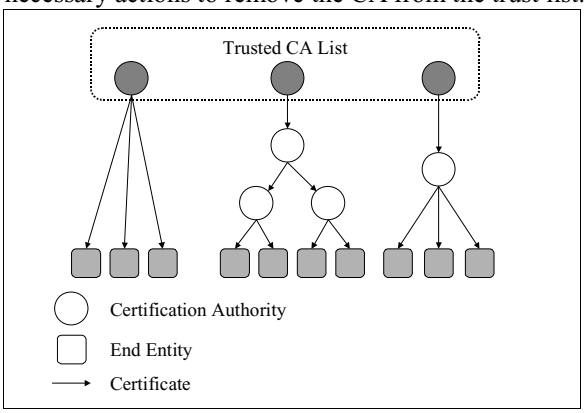

Figure 2 – Multi-Rooted Hierarchical PKI

1.1.2 Mesh: In a mesh style PKI (depicted in Figure 3), each subscriber trusts the CA that issued that subscriber's certificate(s). The CAs in this environment have no superior / subordinate relationship. In a mesh, CAs in the PKI cross-certify. That is, each CA both issues a certificate to and is issued a certificate by peer CAs in the PKI. The figure depicts a mesh PKI that is fully cross-certified, however it is possible to architect and deploy a mesh PKI with a mixture of unidirectional and cross-certifications. Certification path construction in a mesh PKI is more complex than in a hierarchical PKI due to the likely existence of multiple paths between a relying party's trust anchor and the certificate to be verified, and the potential for loops and "dead ends" in non-hierarchical certificate graphs. In addition, while a diagram of the mesh PKI may appear similar to the hierarchical PKI, the certificates for the bi-directional cross-certifications are typically stored as certificate pairs, rather than individual certificates. These pairs are stored in a different directory attribute than is normally used for the certificates in a hierarchical PKI.

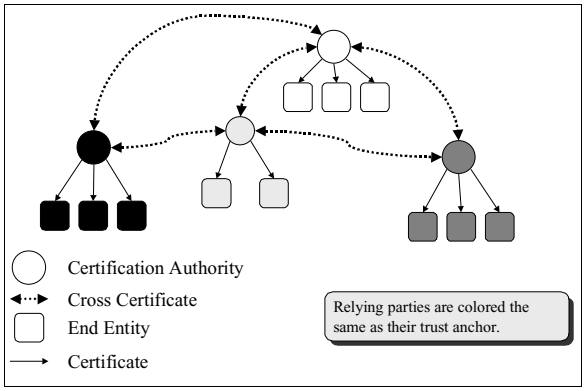

Figure 3 – Mesh PKI

1.1.3 Bilateral Cross-certification: PKIs can be connected via cross certification to enable the relying parties of each to verify and accept certificates issued by the other PKI. If the PKIs are hierarchical, cross-certification will typically be accomplished by each root CA issuing a certificate for the other PKI's root CA. This results in a slightly more complex, but still essentially hierarchical environment. If the PKIs are mesh style, then a CA within each PKI is selected, more or less arbitrarily, to establish the cross-certification, effectively creating a larger mesh PKI. Figure 4 depicts a hybrid situation resulting from a hierarchical PKI cross-certifying a mesh PKI.

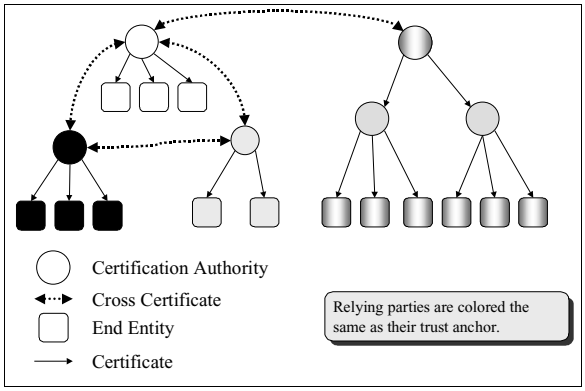

Figure 4: A Hybrid PKI

In current implementations, this situation creates a concern that the applications used under the hierarchical PKIs will not have path building capabilities robust enough to handle this more complex certificate graph. As the number of cross-certified PKIs grows, the number of the relationships between them grows exponentially. Two principal concerns about cross certification are the creation of unintended trust paths through transitive trust, and the dilution of assurance when a high-assurance PKI with restrictive operating policies is cross-certified with a PKI with less restrictive policies.

1.1.4 Bridge: Another approach to the interconnection of PKIs is the use of a "bridge" certification authority (BCA). A BCA is a nexus to establish trust paths among multiple PKIs. The BCA cross-certifies with one CA (known as a "principal" CA [PCA]) in each participating PKI. Since each PKI only cross-certifies with one other entity (i.e., the BCA), and the BCA cross-certifies only once with each participating PKI, the number of relationships in this environment grows linearly with the number of PKIs. However, when interconnecting PKIs in this way, the number and variety of PKIs involved can result in a complex non-hierarchical environment, as shown in Figure 5.

The U.S. Federal government is in the process of establishing a BCA intended to enable interoperability among U.S. Government agencies by cross-certifying with their PKIs. The recommendations in this paper are derived, in large part, from the authors' experience working on the second phase of the DoD-sponsored BCA technology demonstration. Detailed information about the BCA technology demonstration can be found in the final report [2].

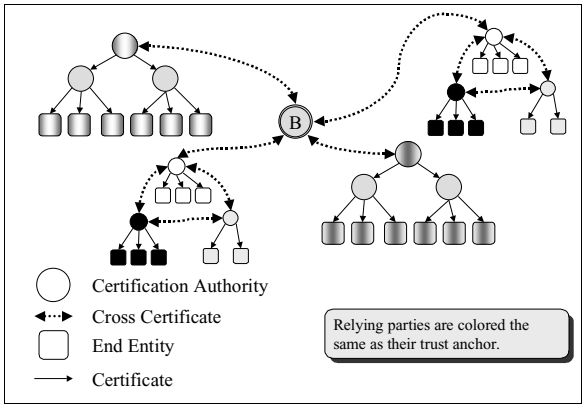

Figure 5: Cross-certification with a Bridge CA

1.2 Scope

The environment assumed for this paper is a non-hierarchical PKI built using the conventions of X.509 and the widely-followed PKIX (RFC 2459) profile [3]. We also assume the use of a directory system for the dissemination of certificates, certificate revocation lists (CRLs), and other information of interest to relying parties. Within the directory system, interactions among directory servers are conducted using X.500 Directory System Protocol (DSP); client access to the directory is typically via Lightweight Directory Access Protocol (LDAP). Finally, we expect that each organization will establish operating policies for its PKI, and that the establishment of cross-certification carries with it the need to determine the relationships between the operating policies of the PKIs involved.

2 Certification Authorities

This section addresses considerations for Certification Authorities (CAs). Certification Authorities are the heart of a PKI; they process certificate requests, sign public key certificates, create and sign CRLs, and publish certificates and CRLs to a directory system to make them readily available to certificate users. RFC 2459 requires conforming CAs to support the authority key identifier, subject key identifier, basic constraints, key usage, and certificate policies extensions. We believe that in a non-hierarchical PKI, support for the name constraints, certificate policy constraints, and certificate policy mapping extensions are also necessary. In addition, certain considerations apply to some of the other extensions.

2.1 Standard certificate components

We have broken our discussion of CA operating practices into those components of a public key certificate that are commonly employed (i.e., standard components) and other components, such as certificate policies, that might be considered "advanced" and less commonly employed, particularly in hierarchical PKIs. Our experience suggests that these "advanced" elements are essential both to achieving interoperability and to maintaining security controls in non-hierarchical PKIs.

2.1.1 Public key/signature: In general, a CA may either generate a key pair or receive, via a request, the public key to be placed in end-entity certificates it issues. However, support for receiving a public key in a certificate request, rather than generating a key pair, is essential when a CA is issuing certificates to other CAs; proper control of the subject CA's private signature key cannot otherwise be guaranteed.

2.1.2 Names: The authors' experience is primarily with systems that use X.500 hierarchical naming conventions. In a non-hierarchical PKI, proper management of the name space is extremely important. In the operation of a CA, the names it places in the subject name of the certificate must be constrained, either procedurally or technically, to the name space for which that CA has responsibility. Proper management of the name space is essential to the regulation of transitive trust in the PKI.

When a PKI is being deployed, the directory administrators should work with the policy administrators in advance to develop a naming architecture that will take into consideration the possibility of future interoperability. This will minimize future naming conflicts (namespace overlap) with external directories. If two directories are configured to chain, and their controlled namespaces overlap, the chaining will not function correctly. (More on directory chaining is found in section 3.1.3). This overlapped naming will cause reduced functionality and interoperability problems once deployed.

2.1.3 Basic Constraints and Key Usage: RFC 2459 requires the basic constraints extension to be present and identified as critical in all CA certificates, and indicates that it should not be present in end-entity certificates. The key usage extension is considered optional in RFC 2459, but must be marked critical if present. In a non-hierarchical PKI, certificate processing systems can potentially benefit from the information provided by these extensions. The basic constraints extension simplifies the identification of CA certificates. The key usage extension provides useful information when a path building algorithm needs to choose among several candidates as the next certificate in a path. We recommend always including the key usage extension, populated according to the proper use of the certificate and any restrictions associated with the cryptographic algorithm in use, and always including the basic constraints extension in a CA certificate.

2.1.4 Authority and Subject Key Identifiers: The authority key identifier (AKID) and subject key identifier (SKID) can be placed in certificates to simplify the certificate path building process. However, section 3.1.4 identifies reasons why the authorityCertIssuer and authorityCertSerialNumber components of AKIDs should not be used. If the key identifier component is used, the AKID in certificate A should equate to the SKID in the certificate of the authority that issued certificate A. Key identifiers can be very beneficial to the efficiency of path building algorithms, and should always be populated.

Some certificate path verification algorithms erroneously require that SKIDs and AKIDs be matched as the path is verified. Some certificate path building algorithms will fail to locate valid paths if there are mismatches in the SKID/AKID values of certificates evaluated as candidates for inclusion in the path. To avoid both of these potential problems, it is extremely important that a CA ensure that the SKID in the certificate(s) issued to it match the AKIDs in the certificates it issues.

2.1.5 Subject Alternate Name: One of the most common applications for interoperable PKIs is electronic mail (email). We recommend always populating the subject alternate name of end-entity certificates with the email address (rfc822Name) of the subject, in order to allow a strong binding between an email address and the public key certificate being used for secure email. Implementors should be aware that a change in the email address will create the need to revoke and replace the certificate.

Additionally, certificate processing systems may wish to able to locate end entity certificates in the directory system for retrieval and use in encryption operations. If the subject DN of the desired certificate is known, the search is straightforward. However, particularly in secure electronic mail applications, the certificate processing system may only know the email address of the end-entity whose certificate is sought. It is our recommendation that the certification authority populate the mail attribute of the certificate subject's directory entry with the email address placed in the subject alternate name extension.

2.2 Advanced certificate components

This section addresses recommendations for other X.509 certificate extensions that are important to the management of interoperability in non-hierarchical PKIs.

2.2.1 Name Constraints: As mentioned in the section above on Names, management of the name space is a particularly important consideration in a non-hierarchical PKI. In a bridged environment, the name constraints certificate extension is the cornerstone tool for managing name space and controlling transitive trust. CAs operating in this environment must be able to populate this extension.

Figure 6 shows a BCA interconnecting three PKIs. In the certificate issued by the bridge to each PKI, a permitted subtree in the name constraints extension provides a definitive specification of the name space that the other PKIs in the environment will recognize for that PKI. For example, the certificate issued by the BCA to the PCA of the organization A PKI will include:

NameConstraints
PermittedSubtrees (c=US, o=A)

If Organization A issues any certificates outside this permitted name space, those certificates will be rejected by relying parties in Organizations B and C. Thus, in the larger community formed by the cross-certification of these PKIs, the Organization A PKI is restricted to a well-known name space.

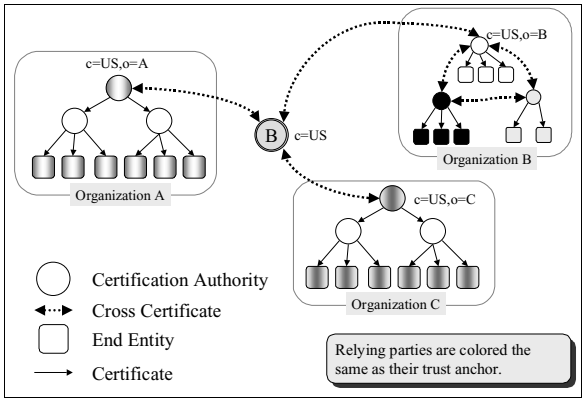

Figure 6: Name Constraints with a Bridge CA

This mechanism also provides a means to eliminate PKIs that an organization does not wish to trust. The use of an excluded subtree in the certificate from a PCA to the bridge will accomplish that goal.

For example, if the certificate issued by PCA-B to the BCA included:

NameConstraints
ExcludedSubtrees (c=US, o=C)

then relying parties within organization B will be able to validate certificates issued by Organizations A and B (and future Organizations D, E, …), but will reject certificates issued by Organization C.

Another benefit of this careful management of name space is the ability of an organization to protect its name space in the bridge environment. In our example, each organization's PCA can include an excluded subtree in the certificate issued to the BCA, with that excluded subtree specifying the organization's own namespace. For example, the certificate issued from the Organization A PCA to the BCA would contain:

NameConstraints
ExcludedSubtrees (c=US, o=A)

This name constraint ensures that relying parties within the Organization A PKI will reject certificates issued by PKIs reached through the BCA that contain names within the Organization A namespace. While certificates from a

"rogue" CA that have names within the Organization A namespace may be recognized by other PKIs, Organization A can be confident that it's own subscribers will not erroneously accept such certificates as valid.

The use of name constraints in an environment involving a Bridge CA requires that interoperating organizations trust the Bridge CA to set name constraints properly in all certificates it issues. See section 2.3.5 for more information.

The combination of clear allocation of the overall name space and routine inclusion of name constraints in the certificates issued by and to PCAs according to strict policy allows the organizations participating in this environment to regulate interoperability as they see fit.

2.2.2. Certificate Policy: In a single organization PKI, the use of certificate policies may be unnecessary. When connecting the PKIs of multiple organizations, however, certificate policies are an important tool for both regulating interoperability and maintaining the level of assurance required for high-sensitivity or high-dollar-value transactions. For example, one organization may invest considerable resources in a very high assurance PKI operating with hardware cryptographic tokens, and rigorously enforced procedures for personnel identification; another PKI connected through cross-certification may operate with software tokens and much less rigorous procedures, and a correspondingly lower level of assurance. The inclusion of certificate policies in all certificates issued by each PKI offers the opportunity for relying parties both within and outside of the PKI to take the PKI's level of assurance into account when accepting specific certificates for specific purposes.

CAs operating in non-hierarchical, and especially multi-organization, environments should routinely include the certificate policies extension in all certificates they issue. In order to simplify the use of mapping to establish equivalencies among the policies of different organizations, we also recommend that the certificate policies extension include all of the organization's policies that apply. For example, assume an organization has established two policies: "basic" and "enhanced", with the "enhanced" policy conveying a higher level of assurance. We recommend all certificates that assert the "enhanced" policy also assert the "basic" policy; this ensures that in situations where the 'less trustworthy' certificate is adequate, the 'more trustworthy' certificate is also useable. This also avoids the need for a user to be issued and have to choose among multiple certificates when a single certificate is sufficient.

The converse to this is when certificate policies are used in a cross-certified, multi-organizational environment to create isolated communities. An organization might create an "internal use only" certificate policy that is not intended to be verified by the relying parties of other, cross-certified PKIs. In order to ensure that such policies achieve their intent, they must appear in end-entity certificates without other policies that may be acceptable to other PKIs through policy mapping.

2.2.3 Certificate Policy Mapping: Certificate policy mapping is an important feature for managing trust in non-hierarchical PKIs. An important benefit of the use of policy mapping is that relying party applications need only be configured with the certificate policies of their own domain as acceptable policies. This allows the organization to manage policy interoperability through cross-certificate content, rather than needing to actively manage the acceptable policy settings of large numbers of relying party applications distributed throughout the organization. CAs need to populate the certificate policy mapping extension whenever they issue a cross-certificate to a CA in another domain.

Certificate policy mapping is a fundamental requirement in the United States Federal PKI Bridge CA (BCA) environment. The U.S. Federal BCA will issue certificates to other PKI asserting certificate policies unique to the BCA, with mappings to the certificate policies of the subject domain. Cross-certifying principal CAs are correspondingly expected to issue certificates to the BCA that assert their own issuer domain certificate policies, with mappings to the BCA policies. This concept is illustrated in Figure 7.

In the example illustrated, a relying party application within Organization B's PKI configured to accept "B High" as an acceptable certificate policy will, through policy mapping, recognize and accept "A High" as an equivalent policy. When adding a new PKI to the BCA community, the BCA's policy authority will evaluate the policies and procedures of the new PKI and define policy equivalencies. When cross-certifying with the BCA, an organization can make an independent determination of the correspondence between the BCA's policies and those of the organization, and provide that determination to the organization's relying parties using the certificate policy mappings.

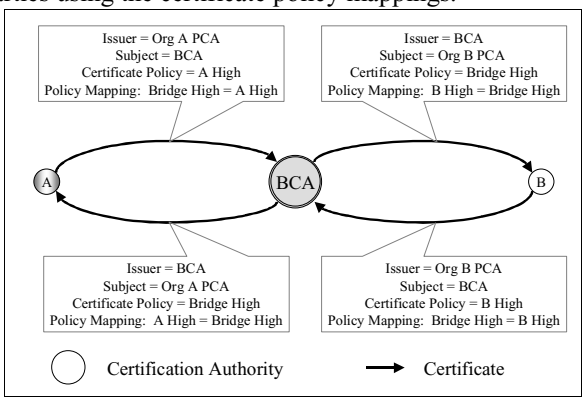

Figure 7: Policy Mapping with a Bridge CA

2.2.4 Certificate Policy Constraints: Certificate policy constraints can be used to control the certificate policy

processing through a trust chain. BCA cross-certificates will often contain certificate policy constraints extensions, with the inhibitPolicyMapping field populated. The certificate from the PCA to the BCA will contain an inhibitPolicyMapping field with the skipCerts value set to 1. This ensures that relying parties from the PCA domain will allow the policy mapping from the Bridge to other PCAs' policies, but no further mappings within or from the other domains are permitted. The certificate from the BCA to the PCA will contain an inhibitPolicyMapping field with the skipCerts value set to 0. This ensures that relying parties from other domains will not accept policy mappings that occur within distant domains. CAs should be able to populate certificate policy constraints in this fashion if they intend to interoperate in a BCA environment.

2.2.5. CRL Distribution Points: Various commercial PKI products have different conventions for where revocation information is published. It may be found in an attribute of the CA's directory entry, stored in another directory location, retrieved from a specified URI, etc. In a single-vendor implementation, the knowledge of where revocation information is published may be built into the applications. However, in environments where multiple vendor products may be employed, it cannot be assumed that applications will have prior knowledge of the usual conventions or specific configuration of any CA. In this environment, CAs should routinely populate the CRL distribution points (DP) extension of the certificates they issue in order to maximize the likelihood that all certificate processing systems will successfully locate the revocation information they need.

2.3 Issues

2.3.1 Cross Certificate Requests: Certification authority products support many different request/response formats for cross-certification. Some of these formats are self-signed certificate exchange, PKCS#10 requests, and requests using the PKIX Certificate Management Protocol (CMP) [4]. The two most important components of the request format are the subject public key, and the subject key identifier. The AKID used in the certificates the requesting CA issues must be passed as the SKID in the request because not all products calculate SKIDs from the public key in the same fashion. For example, in Figure 6, Organization A cross-certifies with the Bridge CA. If the Organization A PCA places an AKID in the certificates it issues, the Bridge CA must use that AKID as the SKID in the cross-certificate issued to the Organization A PCA for maximum interoperability. Certification authority products should be able to fulfill cross-certificate requests for specific public keys and SKIDs.

2.3.2 Policy Mapping: CAs operating in non-hierarchical PKIs may need to deal with multiple certificate policies. In this environment, a CA must be able to map to certificate policies other than those of its own domain. It may also need to include multiple policy mappings in a single cross-certificate. While it is reasonable for a CA to be constrained to only assert its own domain's policy OIDs in the certificates it issues, CA implementers should be wary of imposing arbitrary constraints on the CA's ability to specify the policy mappings in the CA certificates it issues.

2.3.3. Population of Cross Certificates in Directory: Directory systems are not trusted in PKI; they serve as repositories for trusted (signed) objects, but there are no security requirements on the directories themselves. However, it is important to prevent unauthorized modification of directory information because corrupted directory information will lead to a denial of service for directory users. It is important therefore to employ access controls to prevent unauthorized directory modifications.

These access controls can complicate the population of cross certificates. Cross certificates are stored together in the crossCertificatePair attribute in the directory; this attribute contains an ASN.1 encoded sequence of certificates. Unfortunately, the two components of the pair are developed and signed by two different entities. The same pair is also reversed and placed in another entry. If CA X cross-certifies with CA Y, two cross certificate pairs are created:

- In CA X's entry, there is a pair with CA X as the subject in the forward component, and CA Y as the subject in the reverse component.
- In CA Y's entry, there is a pair with CA Y as the subject in the forward component, and CA X as the subject in the reverse component.

Either both CA X and CA Y must have access to read and write each others' crossCertificatePair attribute, or some manual process must be created which allows the data to be updated properly. Certificate issuing systems and directory administrators should consider this when cross-certifying with other domains. Mechanisms, either technical or procedural, must be established to ensure that both elements of the cross-certificate pair are correctly encoded and populated.

2.3.4. Criticality of Extensions: Certain extensions have their criticality defined by RFC 2459. Other extensions, such as key usage, certificate policies, and CRL distribution points can be marked critical or not at the CA's discretion. Marking an extension critical will require it to be processed by relying parties, but will cause failures with any software incapable of processing that extension. This balance between maintaining security controls and allowing interoperability is an important one to consider. Before marking extensions critical, authorities should consider whether it would reduce interoperability with relying parties in cross-certified domains.

2.3.5. Interoperating with a Bridge CA: The population of certain extensions and settings must be done properly in order to reduce the risk of unintended trust. This is especially true in environments involving a Bridge CA. If the Bridge CA violates the trust of the organizations it is involved with and issues certificates improperly (e.g. without proper name constraints or policy constraints specified), relying parties may trust certificates they did not expect or intend to trust. To reduce this risk, the organizations along with the Bridge CA operators need to clarify how certificates and their components are issued. This will most often be specified through certificate profiles as a part of the certificate policy and certification practices of the Bridge CA and PCAs.

3 Certificate Processing Systems

A certificate processing system is a system that is acting on behalf of a relying party in order to determine the validity of a certificate. The most common example of a certificate processing system is found in S/MIME email applications. These applications validate certificates used to verify signatures of incoming messages, as well as validate key management certificates used when encrypting outgoing messages. These applications, hereafter referred to as "thick clients," perform all the necessary processing to build and validate certificate chains. It is expected that some client software that needs to validate certificates will be developed as a "thin client," offloading some or all of the required processing to a server component [5]. In these cases, the certificate processing system is the server component. This section of the document discusses how certificate processing systems need to behave in order to properly process certificates within a non-hierarchical PKI structure. The recommendations here apply to both thick clients and to server components performing these functions on behalf of thin clients. Generally, certificate processing systems perform two logical functions: building the certificate path, and validating the certificate path.

3.1 Certificate Path Building

Certificate path building is the process by which the certificate processing system obtains the certificate path between the trust root and the target certificate. Different implementations build the certificate path in different ways; it is not our intent to recommend a single "best" way to perform this function. Guidance is provided on the technical issues that surround the path building process, and the capabilities a path building implementation must have in order to build certificate paths successfully in non-hierarchical PKI structures.

3.1.1. Certificate Path Completeness: A certificate path is an ordered list of certificates starting with a certificate issued by the relying party's trust root, and ending with the target certificate that needs to be validated. Figure 8 shows an example of a certificate path. In this figure, the arrows represent certificates.

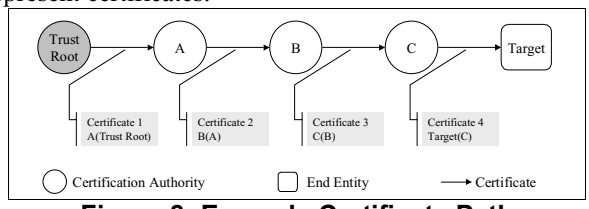

Figure 8: Example Certificate Path

The notation for the certificates is **Subject(Issuer)**. In all certificate paths, the issuers and subjects match in this relationship so that the issuance relationship flows from the trust root to the target, with zero or more intermediate certificates between. In order for the certificate path to successfully validate, the names and signatures must chain throughout the path. In the example in Figure 9, the issuer distinguished name in certificate 1 must match the distinguished name of the trust root. Additionally, the signature on certificate 1 must be verified cryptographically using the trust root public key. Likewise, the issuer name in certificate 2 must match the subject name in certificate 1, and so forth. This must be true for all certificates in the path. Once this information has been verified for all certificates in the path, the certificate path is considered to be complete.

3.1.2. Building Paths in Non-Hierarchical PKI Structures: In a hierarchical PKI, the process of building a path is quite simple; the application starts with the target certificate, obtains the issuer's certificate, and repeats this process until a certificate is encountered that is issued by the trust root. This method, to be referred to as the "find issuer" method, will always find a complete certificate path if one exists in the strictly hierarchical PKI. However, this simple method cannot be used in non-hierarchical PKI structures for a few different reasons.

Dead-ends In a non-hierarchical PKI structure the simple "find issuer" algorithm may fail prematurely without finding an existing path due to a "dead-end". Consider the example in Figure 9:

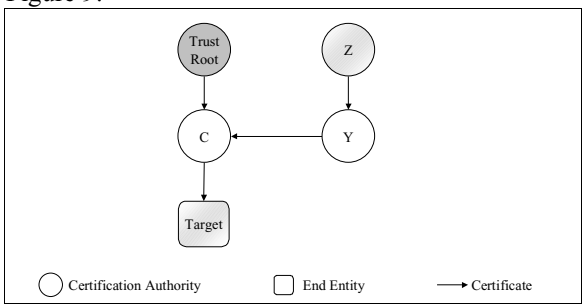

Figure 9: Dead-end Example

Note that in this example, C has two certificates: one issued by Y, and the other issued by the Trust Root. Let us suppose that the "find issuer" algorithm is used, and the order in which it retrieved the certificates was Target(C), C(Y), Y(Z), Z(Z). In this case, Z has no certificates issued by any other entities, and so the path building process stops. Since Z is not the relying party's trust root (Z may be the Target's trust root), the certificate path is not complete, and will not validate. This example shows that in a non-hierarchical PKI, additional complexity must be included to handle the cases in which entities are issued multiple certificates from different issuers.

Loop Detection In a non-hierarchical PKI structure, the "find issuer" algorithm may become caught in a loop without finding an existing path. Consider the example in Figure 10:

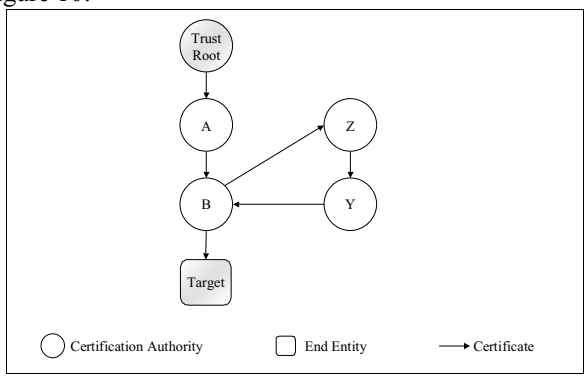

Figure 10: Loop Example

Let us suppose that in this example the "find issuer" algorithm is used, and the order in which certificates are retrieved is Target(B), B(Y), Y(Z), Z(B), B(Y), Y(Z), Z(B), B(Y)… You can see that a loop has formed, which will cause the correct path (Target, B, A) to never be found. The certificate processing system must recognize loops before they form to allow the certificate path building process to continue and find valid paths. We recommend that the loop detection not only detect the repetition of a certificate in the path, but also detect the presence of the same subject distinguished name / subject public key pair occurring twice in the path. A name/key pair should only appear once in the path, otherwise the trust chain is diluted.

3.1.3. Certificate Retrieval in Non-Hierarchical PKI Structures: If the certificate path building function uses an X.500 or LDAP repository to obtain certificates, there are multiple attributes that must be examined within the directory entries for the entities in the path. There are two common protocols for accessing information from a directory system, Directory Access Protocol (DAP) and LDAP. LDAP has become the protocol of choice for client-to-directory communications. Many directory products available today support client-to-directory communications via either DAP or LDAP, while using Directory System Protocol (DSP) for directory-to-directory communications.

As mentioned earlier, directory systems are capable of interoperating via the use of DSP chaining. This chaining relationship allows for a distributed directory system, thereby taking the burden off of the client when performing retrievals. It is important to note that DSP chaining eliminates the need to configure client software to access multiple directories. The DSP chained directory system will obtain the information requested by the client (if it is available through one of its chaining relationships) in the client's original request.

Some directory systems are not capable of performing DSP chaining or are intentionally not chained to other directories for various reasons. In these cases, LDAP referrals are often used. When queried for information that is not available in the directory system, the system may respond with a "not found" error, and an LDAP referral in the extra information portion of the LDAP response. The referral contains a list of one or more other servers that should be contacted to check for the information [6]. When directories without chaining relationships are used, directory clients must be capable of receiving LDAP referral information, and following the referrals to obtain the required data.

Typically, end-user certificates are found within the userCertificate attribute, and CA certificates are found within the cACertificate attribute. In non-hierarchical PKIs, the crossCertificatePair attribute must also be examined to obtain cross certificates issued between entities. The object within the crossCertificatePair attribute is not itself a certificate; it is an ASN.1 encoded sequence of two certificates. Certificate processing systems that use X.500 or LDAP repositories to obtain certificates must be able to retrieve and parse crossCertificatePair attributes in order to successfully build paths in non-hierarchical PKIs.

3.1.4. Improving Path Building Efficiency using Key Identifiers: Key identifiers refer to two extensions that may appear in certificates; subject key identifier (SKID), and authority key identifier (AKID). These extensions were created to assist in the path building process. Consider the example in Figure 11:

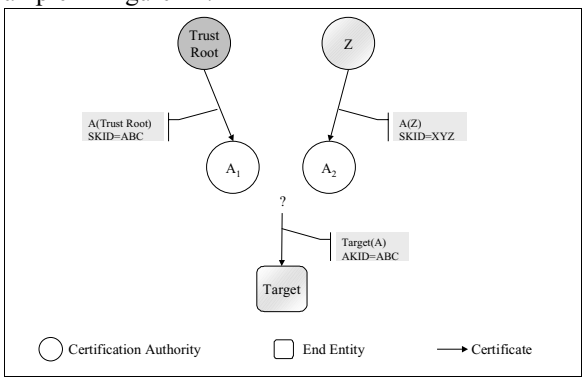

Figure 11: Key Identifier Example

In this example, our target has one certificate, but there are two certificates for the issuer A. These two certificates have the same subject distinguished name, but different public keys and subject key identifiers. The path building algorithm should favor the certificate in which the target's AKID matches the issuer's SKID when choosing between the two issuer certificates. This will most likely lead the path building in the correct direction.

There are two issues with the use of AKID and SKID that we will present. The first is the use of the authorityCertIssuer and authorityCertSerialNumber parts of the AKID to identify the issuer of a certificate. This type of AKID is not useful in finding the issuer of a certificate in a cross-certified environment. This is because the issuer of a certificate may itself be the subject of many certificates from different issuers, and the AKID may not have been developed from the point of view of the relying party. Consider the example in Figure 12:

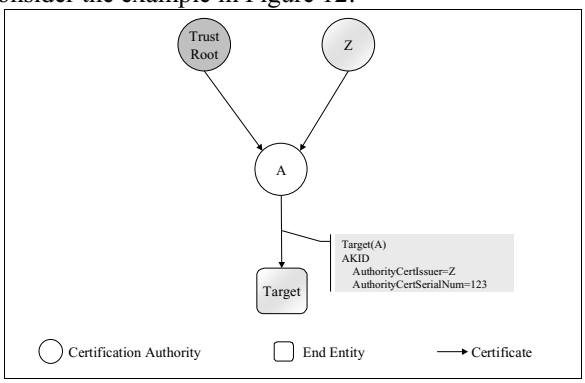

Figure 12: Authority Key Identifier Example

If the target certificate contains an AKID with the authorityCertIssuer and authorityCertSerialNumber components, the information in that extension may lead to the certificate A(Z), which leads away from the relying party's trust root. Therefore, in a non-hierarchical PKI, AKIDs containing authorityCertIssuer and authorityCertSerialNumber should be ignored by certificate processing systems.

The second issue is a weakness of many current certificate processing systems. Although the AKID and SKID extensions may be used to assist in the path development process, no standard requires them to match or be present in order for a path to be valid. Certificate processing systems should use AKID and SKID values to assist in path building, but should not require them to be available or to match appropriately in order to successfully build and validate a complete certificate path.

3.1.5. Caching and Retrieval Efficiency: Certificate processing systems operating in a non-hierarchical PKI will often need to retrieve certificates and certificate revocation lists (CRLs) from a source outside the application protocol. Typically, these objects are retrieved from an X.500 or LDAP repository, an Internet URI, or some other non-local source. Due to the delays associated with both the establishing of connections as well as network transfers, certificate processing systems should attempt to be as efficient as possible when retrieving data from external sources. Certificate processing systems that are consistently very slow during processing will be disliked by users and will be slow to be adopted into organizations. Certificate processing systems should do whatever possible to reduce the delays associated with requesting and retrieving data from external sources.

Perhaps the best way in which retrieval efficiency can often be improved is by the use of a caching mechanism. Certificate processing systems should cache data retrieved from external sources for some period of time, not to exceed the useful period of the data (i.e., an expired certificate should not be cached). Although this comes at a cost of increased memory/disk consumption by the system, the benefit of reducing network transmissions is great. In the testing performed during the NSA Bridge CA Phase II demonstration [2][7], the applications that performed caching consistently demonstrated significantly improved performance the second or subsequent time a certificate path was processed, or when similar (e.g., same issuer) target certificates needed to be validated.

3.1.6. Other Optimizations: The Certificate Path Development Library (CPDL), developed by Entrust CygnaCom, provides a freeware implementation of a path building algorithm that works in non-hierarchical PKI structures. This algorithm performs a few additional optimizations that may be worthwhile to implement in other certificate processing systems. The first is filtering. While building paths, the CPDL filters out any certificate that can be ruled out because it would cause the current path to be invalid. Two examples of filters that are used are filtering expired certificates and certificates that will invalidate the policy requirements of the current path. In these cases, if a choice needs to be made between multiple certificates that may all be the next part of the certificate path, the filtering sometimes allows the choices to be reduced.

The second optimization performed by the CPDL is certificate sorting. Once again, at some point during the path building process the algorithm may be faced with the choice of more than one certificate that may complete the path. The CPDL sorts these certificates according to rules designed to favor certificates that are more likely to lead to a valid certificate path. Two examples of sorting rules are sorting certificates that contain consistent signature algorithms ahead of those that do not, and sorting certificates that have more relative distinguished name (RDN) components in common with the trust root ahead of those that have fewer. Complete details on the filtering and sorting rules performed by the CPDL are available at the CPDL website, http://www.cygnacom.com/products

3.1.7. Building Direction: A certificate path can be built in two common directions. These directions are sometimes referred to by "forward" and "reverse", but those names can be misleading. They will be referred to here as "building from the root" and "building from the target". In the case of building from the root, the algorithm generally starts with the root, and attempts to retrieve certificates that are the subject of the current certificate, until ending at the target certificate. Building from the target starts with the target certificate, and attempts to retrieve the certificate(s) of the of the current certificate's issuer, until ending with a certificate issued by the root.

Both methods may be possible in differing PKI structures. It may be difficult to build from the root in a strict hierarchy; it may be difficult to build from the target in a highly cross-certified environment. A paper entitled "Building Certificate Paths: Forward vs. Reverse" [8] was presented at the 2001 NDSS conference, and favored building from the root. Building from the root, while appealing because of the optimizations that can be performed during the path building process, has some practical problems. The first is requiring the presence of reverse cross certificates. Building from the root can only be performed when the reverse element of crossCertificatePair components are present for most (if not all) of the CA certificates in the path. Unfortunately, the standard LDAP v3 schema requires the presence of the forward element of crossCertificatePair, but allows the reverse element to remain optional (this aspect of the LDAP v3 schema is being changed to require populating both elements). Therefore the certificates desired may not be present. The second is the act of performing signature validation while building certificate paths. Building from the root allows path completeness to be checked as each new certificate is added to the path—to ensure no certificate with an invalid name or signature is ever added to the path. In practice, this approach may be counterproductive because cryptographic verification operations are relatively slow compared to other, simpler checks of certificate content. If many subject certificates are present (which might be the case in an environment containing a Bridge CA), it is possible that many unnecessary signature validations may occur which would slow down the path building process.

This document does not favor either direction for building. Implementors of building algorithms should consider the points made in the aforementioned paper as well as the items discussed here before making a decision on building direction.

3.2 Certificate Path Validation

Certificate path validation is not much different in a non-hierarchical PKI than it is in a strict hierarchy. This is because once the certificate path has been constructed, it will, in any case, be presented to the validation algorithm as a simple chain of certificates, starting with a certificate issued by the root, and ending with the target. The main concern for certificate processing systems is that the validation algorithm is able to process and validate basic certificates along with the certificate extensions outlined in Sections 2.1 and 2.2 of this document. Standards such as PKIX RFC 2459 do not mandate support for all of those extensions; in order to maintain all security controls while allowing interoperability, these extensions must be processed. (A chart detailing differences between the requirements of PKIX RFC 2459 and this document is found in the conclusion) The extensions should be processed exactly as outlined in the X.509 or PKIX RFC 2459; no special handling outside those standard recommendations is needed.

Experience has identified certain issues regarding certificate path validation in non-hierarchical PKI structures. The remainder of this section will address those issues and provide recommendations.

3.2.1. Extensions in Trusted Root Certificates: In the previous section entitled Certificate Path Building, a certificate path was defined as the set of certificates between the target and a certificate issued by the trust root. It is the opinion of the authors that a certificate path does not include the self-signed trusted root certificate. The X.509 standard leaves room for interpretation as to whether or not the self-signed trusted root certificate should be part of the certificate path, and used in the validation process. After much discussion and debate with people involved in both the X.509 and PKIX standards processes, the authors have concluded that self-signed trusted root certificate should not be part of the validation process. The self-signed trusted root certificate simply provides a convenient container to store both a distinguished name and public key for the trusted root. Extensions in the self-signed trusted root certificate such as basic constraints, name constraints, certificate policy constraints, etc. should be ignored during the validation process. If the trusted root wishes to impose these constraints on the certificate path, these extensions should be present in all certificates issued by the trusted root in order to ensure they are processed by all certificate processing systems.

3.2.2. Order of Validation Operations: Both the X.509 and PKIX RFC 2459 standards provide example algorithms for the validation of certificate paths. These algorithms provide a reference to establish the correct results of path processing, but are not necessarily the best or most optimized way to validate certificate paths. More likely, these algorithms were chosen because of the ability to describe them fully and accurately. Certificate processing systems are free to implement whatever validation algorithm they choose, as long as the end results are guaranteed to be the same as these standard algorithms.

It is our recommendation that the algorithm used to validate certificate paths perform all checks on the path in the order from fastest checks to slowest checks. Consider three aspects of certificate path validation: name chaining, policy chaining, and signature chaining. Checking name chaining is extremely fast, so that might occur first. Checking policy chaining is also quite fast, although a bit more involved than name chaining, so it might be performed second. Checking signature chaining requires cryptographic operations that are computationally intensive, and may involve an external cryptographic device such as a smart card. This is potentially quite slow, in relative terms, so it should be performed last. Performing these checks in order from fastest to slowest allows invalid paths to be ruled out quickly, so that the next possibly valid path can be built and validation attempted. Reducing unnecessary delays will increase the usability of any certificate processing system.

3.2.3. Revocation Checking: As a part of the certificate path validation process, all certificates should be checked to ensure they have not been revoked. In certain PKIs this is handled by the use of certificate revocation lists (CRLs); in others, an on-line protocol such as the online certificate status protocol (OCSP) is used. A requirement of any certificate processing system wishing to operate in a specific PKI environment is to be able to retrieve and process revocation data for that environment in whatever form it may be provided. Applications operating in non-hierarchical, multi-vendor PKIs may encounter several different approaches to publishing revocation data in the validation of a single certificate path.

OCSP Processing A certificate validation algorithm that issues an OCSP request will encounter some delays in the network communication associated with this request. As was mentioned before, it is important to consider these communication delays while developing the algorithm for the validation process to ensure as minimal a delay as possible.

CRL Retrieval While certificate status checking is typically considered part of the validation process, the certificate processing system may actually wish to obtain CRLs during the path building process. This is because while certificates for intermediate CAs are being retrieved during the building process, the additional delay and overhead to obtain the associated CRL at the same time may be small enough to warrant this extra processing. However, if CRLs are expected to be large, or if the PKI is highly cross-certified, leading to many incorrect retrievals before the correct path is found, it may not be as beneficial to retrieve CRLs during the certificate path building process.

3.2.4. CRL Distribution Point Processing: The CRL Distribution Points (CRL DP) extension is commonly found in certificates operating in both hierarchical and non-hierarchical structures. This extension identifies a list of one or more locations where the CRL that may contain the current certificate can be found. Certificate processing systems should process this extension to maximize interoperability.

4 Conclusion

We have successfully applied these concepts in the DOD BCA technology demonstration to show how cross domain PKIs can work with each domain controlling how much to trust another domain. The demonstration illustrates that the more sophisticated aspects of X.509 certificate issuing and processing, especially certificate policies, policy mappings, and name constraints, are effective tools that allow the PKI to establish the limits of security interoperability between organizations in tangible form. Consequently, these X.509 features should be populated by certificate issuing systems, and expected and processed by certificate processing applications. We believe that this will promote interoperability among the relying parties within non-hierarchical PKIs, while still allowing the participating organizations to maintain security control.

The following chart outlines the differences between the recommendations of this paper and guidance given by PKIX RFC 2459 on particular certificate extensions used when working in non-hierarchical PKI structures.

	RFC 2459	This Document
Basic Constraints	Populate and process	Populate and process
Key Usage	Populate and process	Populate and process
Certificate Policies	Populate and process	Populate and process
Authority and Subject Key Identifiers	Populate, recommend process	Populate and process (see 3.1.3.1)
Certificate Policy Mappings	Optional	Populate and process
Name Constraints	Optional	Populate and process
CRL Distribution Points	Optional	Populate and process
Subject Alternative Name	Optional	Populate

Specific recommendations for those that are using or plan to use a non-hierarchical PKI structure are listed below for convenience. For more information about each recommendation, see the referred section.

Certification Authorities must be able to…

- Populate all the extensions listed in sections 2.1 and 2.2
- Populate subject key identifier with a requested value provided by the authority being cross-certified (section 2.3.1)

- Perform policy mappings to policies other than those configured by the system at installation time (section 2.3.2)
- Perform multiple policy mappings within the same certificate (section 2.3.2)
- Populate the crossCertificatePair attribute in a directory if applicable (section 2.3.3)
- Configure the criticality of extensions as allowed by PKIX RFC 2459

Certificate Processing Systems must be able to…

- Build certificate paths using cross certificates (section 3.1.2)
- Build certificate paths in the presence of dead-ends and loops in the PKI structure (section 3.1.2)
- Build certificate paths without requiring the presence of properly configured key identifiers in all certificates (section 3.1.4)
- Processing all extensions listed in sections 2.1 and 2.2 as outlined in PKIX RFC 2459
- Ignore extensions in trusted root certificates when validating certificate paths (section 3.2.1)
- Be as efficient as possible when building and validating certificate paths (sections 3.1.5 and 3.2.2)

5 References

1. ITU-T Recommendation X.509: Information Technology – Open Systems Interconnection, "The Directory: Public-key and Attribute Certificate Frameworks", March 2000.
2. National Security Agency. "Phase II Bridge Certification Authority Interoperability Demonstration Final Report", prepared by A&N Associates, 2001. Available at http://www.anassoc.com/Techno.htm
3. R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure--Certificate and CRL Profile", IETF Request for Comments No. 2459, January 1999.
4. C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure--Certificate Management Protocols", IETF Request for Comments No. 2510, March 1999.
5. A. Malpani, P. Hoffman, R. Housley, T. Freeman, "Simple Certificate Validation Protocol (SCVP)", IETF PKIX Working Group Internet Draft, July 2000 (work in progress).
6. M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", IETF Request for Comments No. 2251, December 1997.
7. National Security Agency. "Technical Interoperability Profile for the Bridge Certification Authority (BCA) Interoperability Demonstration Phase II", prepared by A&N Associates, 2001.
8. Y. Elley, A. Anderson, S. Hanna, S. Mullan, R. Perlman, S. Proctor. "Building Certification Paths: Forward vs. Reverse", In *Network and Distributed System Security Symposium Conference Proceedings: 2001*