Chapter 7

Certificate Path Verification in Hierarchical PKI

7.1 Introduction

In order to support Public Key Cryptography, X.509 public key certificates have become the most popular standard for securely binding the identity of an individual or a device to a public key. Before obtaining the public key from a certificate, it is necessary that the certificate is to be verified to determine the authenticity of that certificate, and specifically, the validity of all the certificates leading to a trusted public key, called a trust anchor. Through validating this certification path, the assertion of the binding made between the identity and the public key in each of the certificates can be traced back to a single trust anchor. The process by which an application determines this authenticity of a certificate is called certification path processing.\footnote{The research paper based on this work is published in LNCS-CCIS, Springer Berlin Heidelberg, Vol.70, ISSN:1865-0929 (Print) 1865-0937 (Online), ISBN:978-3-642-12213-2 (Print) 978-3-642-12214-9 (Online), Pages:320-324, April, 2010} Certificate path processing establishes a chain of trust between a trust anchor and a certificate. A Certificate path is an ordered sequence of certificates where the subject of each certificate in the path is the issuer of the
next certificate in the path. A certificate path begins with a trust anchor certificate and ends with an end entity certificate. In Hierarchical PKI, typically only one superior CA certifies each other CA. Certificate 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. Hierarchical PKIs have four attractive properties due to their simple structure and unidirectional trust relationships[14]. They are as follows:

- These are scalable
- Certification paths are easy to develop because they are unidirectional
- Certification paths are relatively short
- Users of a hierarchy know implicitly which applications a certificate may be used for, as a result, certificates used in a hierarchy may be smaller and simpler.

To describe the notation defined by X.509 to represent certificate path, the following relationships between CAs, RCA1, CA1N, and an end entity E1NN are initially assumed in Figure 7.1.

![Figure 7.1: The Hierarchical PKI](image)

E1NN is an end entity certified by CA1N and CA1N is directly subordinated to RCA1 in the hierarchy. Therefore, the notation for the above path is represented
The processing of certificate paths may be a very complicated and time demanding operation, depending on the length of the certificate path and the possible inclusion of relations using cross-certification. Cross-certification is required when users from different PKIs are to be able to trust each other’s certificates. The processing of a certificate path in order to verify its validity is composed of two steps:

- **Path determination** - the certificates are retrieved from a repository and the path is constructed.

- **Path validation** - the certificates in the path are checked for integrity and validity period, and information related to semantics is verified. Most commercial CAs and Web browsers for secure socket layer(SSL) authentication, as well as Secure Multipurpose Internet Mail Extension(S/MIME) use certificate chain as the method for path processing[8].

We can construct certificate path in two ways:

1. **Forward path construction in which the path is constructed from end entity certificate to the trust anchor certificate.**
   The Forward certificate path construction is a straightforward approach whereby we start constructing the path from target certificate to the root certificate via the intermediate CAs’ certificates. It is a straightforward approach because the path is unidirectional and unambiguous. Each certificate contains its issuer’s information and so it simplifies the task of path verification.

2. **Reverse path construction in which the path is constructed from trust anchor certificate to end entity certificate.**
Reverse path construction is not straightforward because it is difficult to determine the exact path from root CA to the target certificate directly.

7.1.1 Issues in the path building process

Certificate path verification involves the task of discovering a path and then verifying its validity. The path discovery is a complex task that involves discovering a network of CAs that trust each other. Several factors contribute to the complexity.

First, there are many possible trust models[79][83][143][144]: Single level, Hierarchical, Multi-rooted Hierarchical, Mesh, Hybrid and Bridge models. Among these, the Hierarchy of CAs offers the simplest form for path discovery and validation. The Mesh and Bridge-CA topologies necessitate the path discovery software (Relying Party or RP Software) to be intelligent enough to navigate through the network by avoiding loops and making appropriate decisions about the selection of links (i.e., which CA-CA trust path to be selected). However, many of the present implementations are not capable of handling these complex trust relationships[79][143][145][146]. One more obstacle is the changes in the trust hierarchies over time which necessitate changes in the RP, which is very tedious to achieve.

Second is the trust policy or certificate policy issue. Each RP has an acceptable trust policy for the chain of CAs that it is trying to validate for a given certificate[11]. The minimum level of CA practices that it expects from each of the CAs on the path is defined in the certificate policy. The practices (policies) that each CA specifies in its certificate, must be followed by other CAs in the path[79][143]. As a consequence of a wide variety of such policies, the task of RP becomes more complex during path discovery since it should select only those links that are compatible with its (RP’s) own policy. Determining the policy compatibility is once again a complex task[81][82][83].

Third, the response time is also one of the issues for certificate path validation. For example, the RP cannot respond to its user’s request until the path validation
and the subsequent certificate validation are complete.

Fourth, the availability of CA and the certificate repository (for ex. LDAP) is also an important concern to an RP. To validate a path, all of the CAs and the relevant certificate and revocation information repositories need to be available well in time. If a trust path is long with several CAs, there could be a problem in data collection as some entities along the path may not be accessible during validation[79][83][146]. In other words, even though a CA-CA certificate is valid, due to the inability to access a CA’s data at the validation time, an RP may not be able to validate a certificate.

Finally, cost of validation is one more important issue for the RP. It would like to minimize this cost. If the entire process of validation is repeated for each certificate that the RP receives, the cost of path validation may increase. From the above discussion, it is clear that certificate validation is computation and communication intensive making the process complex[147]. Even though, there is no standard for certification path discovery, a few researchers have shown their interest in improving the performance of certification path building algorithms. The PKIX working group published a draft[117] that analyzes the performance issues and complexity of the algorithms and provides some recommendations for efficient path discovery. There are a few studies that highlight performance of different algorithms and also provide some recommendations[78][80]. Once the certification path is established, relying parties can take advantage of several services to validate the certificate path. One of the choices is to use Delegated Path Discovery (DPD)/Delegated Path Validation (DPV)[130]. This specification contains the definition of messages exchanged by the servers and relying parties. The purpose of separating these two services is to satisfy relying parties’ requirements separately. Some relying parties are only concerned about whether the certificate and the corresponding certification path are valid or not, but they do not require other information. A DPV server is suitable for this type of relying parties. It accepts a set of parameters by the relying party, constructs the certification path, and vali-
dates the path for the relying party. Other relying parties may require additional information. The DPD server, on the other hand, returns to the relying party the resulting certification path together with any other additional information, such as full CRLs, OCSP responses, etc. The requesters use a set of rules, called the path discovery policy, to determine which information to return. One more certificate validation protocol is Simple Certificate Validation Protocol (SCVP)[147]. Similar to DPD/DPV, SCVP allows relying parties to assign the task of certificate validation to a server. An SCVP server can provide a wider range of information about the target certificate. It can provide validity information of a single certificate, as well as the entire certification path. SCVP makes the client implementations simple and allows enterprises to centralize trust and policy management.

7.1.2 Criteria for Path Building

The developers of certificate path-building algorithms have to consider the following criteria[21].

Criterion 1: The implementation must build all potentially valid certification paths between the trust anchor and the target certificate.

Criterion 2: The implementation must be as efficient as possible. An efficient certification path-building implementation is defined to be one that builds paths that are more likely to validate before building paths that are not likely to validate. For example, if a particular path is entirely valid except for a single expired certificate, this is most likely the ‘right’ path. If other paths are developed that are invalid for multiple reasons, this provides little useful information.

7.2 Path Validation

Verification for the integrity, trustworthiness and appropriateness of all the certificates in a certification path, including the certificate issued to the subscriber is referred to as the Path Validation. This is carried out by the relying party. X.509
describes all of these processes as necessary for the relying party. Path validation includes a set of process inputs, the processes themselves, and a set of process outputs as shown in Figure 7.2.

![Figure 7.2: Path Validation Outline](image)

### 7.2.1 Path Validation Inputs

The certification path, relying party’s trust anchor public key, revocation status information for each certificate in the path, and any process constraints relevant to the particular instance of path validation to tailor it to the policy environment in which the transaction is being executed are the inputs for Path validation process.

- A trust anchor is a CA that is trusted by the relying party, and its public key is the basis upon which other trust decisions are made. So there must be some schemes that are used to convey trust anchor public keys securely to the relying party. To maintain security, relying party community may be restricted to use a single trust anchor, the local CA, for all instances of
path validation. Practically, sometimes, the root keys of trust anchor(s) are pre-installed in popular browsers.

- A certification path is a sequence of one or more certificates from the relying party’s trust anchor to the subscriber. In some applications, such as S/MIME, a certification path or a portion of a certification path may be provided to the relying party by the subscriber. In other environments, the relying party must locate and retrieve the necessary certificates from some repositories such as LDAP.

- If the subscriber no longer meets the requirements for membership of the domain, the certificate will be revoked by the CA. Current revocation status information is required for each of the certificates in the certification path. If a certificate is revoked, the certificate issuing CA generally makes this status change available on a Certificate Revocation List (CRL). Revocation status can be communicated to relying parties. CRLs are regularly updated enabling relying parties to determine the status of the certificates before they go for validation[148][149].

- The relying party community may use different certificate policies based on the application for which they are issued the certificates. In order to support different certificate policies in the transactions, process constraints are used to partition the relying party community based on the certificate policies. While the most obvious environment for use of process constraints is in a federated environment where the goal is to restrict relying parties to trust only a subset of the entities in the federation, there are other uses that can be made of process constraints. Process constraints can also be used to enable a simple authorization service. If role information is included in subscriber certificates, relying parties can authenticate the subscriber as well as make basic authorization decisions based upon the subscriber’s authenticated identity and/or associated role information. Path validation, with an appropriate
set of business controls, automatically provides this service.

7.2.2 The Path Validation Process

The path validation process consists of a number of individual processes that fall into three basic categories: path structure, certificate validity and business controls. The X.509 standard includes an outline of a procedure for path validation. While standard conformance demands that an implementation be functionally equivalent to the external behavior resulting from those outlines, it is not mandatory for implementations to use either of those algorithms directly.

7.2.2.1 Path Structure

In order to ensure that the certification path is properly constructed, path structure process consists of four tests. If all certificates in the path pass these tests, the certification path is a proper path in that the intermediate entities are all CAs and the path forms an unbroken chain of certificates to the subscriber public key. If any of these individual processes fails, the path validation process fails. These four tests are described below:

- **Name Chaining**
  
  In Name Chaining test, the subject field in one certificate is compared with the issuer field in the subsequent certificate. If these values match for each pair of adjacent certificates in the path, the certification path is truly an unbroken chain of entities relating directly to one another and that it has no missing links.

- **Key Chaining**
  
  The key certified in each certificate is verified with the digital signature on the subsequent certificate in Key Chaining test. This check establishes cryptographic trust from the relying party’s trust anchor to the subscriber’s public key. This check is nothing but matching the `subjectKeyIdentifier`
extension in one certificate with the `issuerKeyIdentifier` extension in the subsequent certificate.

- **Duplicate Certificates**
  The duplicate certificate test checks for redundant certificates in the path. If the path includes two or more certificates that have matching issuer names and identical serial numbers, they are considered duplicates or redundant. This check ensures that a loop does not adversely alter the effect of business controls, changing the outcome of proper path validation. A duplicate certificate could impact the length of the certificate path.

- **Subject Entity Type**
  This test involves checking Root CAs as well as intermediate CAs are legitimate issuers of certificates. This check is performed by ensuring that the `basicConstraints` extension is present in the certificate, with the `cA` element set to `TRUE`.

### 7.2.2.2 Certificate Validity Process

Certificate validity processing ensures that each certificate is a valid X.509 public-key certificate and that the basic requirements for use of the public key are satisfied. If any of these processes result in failure for any certificate in the path, the path validation process fails.

- **Syntax Check**
  This check is meant for checking the syntax of certificates in the path. The syntax for certificates is defined in X.509. Each certificate extension defines its own syntax. A standard set of extensions is specified in X.509. If there are unknown extensions present, their syntax cannot be checked.

- **Integrity Check**
  The integrity of each certificate must be verified to ensure that certificate
content has not been altered from that signed by the CA. This check involves verifying the digital signature on the certificate. If the signature on any certificate in the path fails to verify, then that certificate has been corrupted and cannot be trusted.

- **Validity Period Check**
  This check is meant for checking the validity period of a certificate. After expiry of a certificate, the certified public key is considered as invalid for use. Generally, the certificate validity period is compared against the current time.

- **Revocation Check**
  Even if all certificates are within their validity period, it is possible that one or more may have been revoked by its issuer. Current revocation status information must be checked for each certificate in the path. For example, this process could ensure a relying party did not accept the signature of a recently dismissed disgruntled employee on a business transaction.

- **Criticality Check**
  All certificate extensions include an indication of whether their processing is critical to the acceptance of that certificate for use. The criticality flag provides a backward/forward compatibility mechanism that enables CAs to state what should happen when a relying party that does not yet support the new extension encounters it. If an extension is flagged critical, but the relying party path validation process does not support that extension, the certificate cannot be used. Unknown non-critical extensions can safely be ignored and other processing continued.

- **Key Usage Check**
  The **keyUsage** certificate extension indicates the general use of the certified public key. For the subscriber’s certificate this may be set to any one of a number of values. For example, if the **keyUsage** extension is set to
“digitalSignature”, the certificate can be used for only client authentica-
tion and if it is set to “keyCertSign”, then the certificate can be used to
verify the subject CA’s signature on subsequent certificates.

7.2.2.3 Business Controls Process

Business controls processes ensure that all certificates in the path adhere to busi-
ness and policy constraints imposed on the path validation process through infra-
structure constraints and/or process constraints. Process constraints are introduced
at the beginning of the path validation process through initialization of a set of
process variables. As such, process constraints automatically apply to the com-
plete set of certificates in the path. There is no criticality flag associated with
process constraints. All process constraints are treated as if flagged critical and
must be adhered to. Most infrastructure constraints apply only to subsequent cer-
tificates, but some also impact the complete set of certificates in the path. Some
infrastructure constraints can be flagged non-critical allowing unknown extensions
to be ignored by relying party path validation processes that do not support them.
The set of standard business controls can be broadly categorized as constraints
imposed through naming, policy and length of acceptable certification paths.

• Name Controls

Name controls ensure that the certificate subject names are unambiguous by
assigning them within a hierarchical structure such as distinguished names,
RFC822 names, DNS names, etc. The name control can be done by setting
the relevant fields of the nameConstraints extension.

• Policy Controls

Certificate policy is the standardized mechanism used by a CA to indicate
the applicability of a certificate to a particular community and/or class of
application with common security requirements, such as the authentication
of a user to a home-banking application or the encryption of email messages.
Each certificate policy has a unique identifier. Certificates may be issued under one or more policies as indicated in the `certificatePolicies` extension of that certificate. If two PKI administrative domains issue certificates under equivalent certificate policies, but use different identifiers for those policies, they can equate those identifiers through the `policyMapping` extension in the cross-certificates issued between those CAs. The `certificatePolicies` extension can be included in both subscriber and CA certificates. The `policyMapping` extension can only be included in CA certificates. Unless otherwise constrained, a remote policy, mapped to an equivalent local policy, would be considered a match for its equivalent local policy. There is also a special identifier for “anyPolicy” that can be included in certificates. If the policy mapping is not inhibited by a related business control, this identifier is considered a match for any specific policy identifier.

- **Path Length Controls**
  The final standardized business control enables CAs to constrain the length of a certification path. If one CA imposes a constraint on the acceptable length of a certification path, subsequent certificates in the path can only further reduce the acceptable length. Path length constraints in subsequent certificates that have a higher value must be ignored. At present, the base standards only enable this control as an infrastructure constraint through the inclusion of a `basicConstraints` extension with the `pathLenConstraint` field present.

### 7.2.3 Process Outputs

The path validation process results in either success or failure. Success indicates that the subscriber’s certificate, based on the inputs to the process, is acceptable for the intended use. Specifically:

- The path was properly constructed and all tests in section 7.2.2.1 passed
• All certificates in the path are considered valid and all tests in section 7.2.2.2 passed

• All business controls imposed as process constraints as well as in infrastructure constraints have been satisfied.

7.3 Certificate Path Verification in Hierarchical PKI with Forward and Reverse path Constructions

Certificate Path Verification is building a trusted path between the trust anchor certificate and the target entity certificate based on the trust relationship among the CAs of the PKI and validating the certificates.

As mentioned in section 7.1, Forward certificate path construction is a straightforward approach since there exists a single path from end entity certificate to the root. Reverse path construction is not straightforward because it is difficult to determine the exact path from root CA to the target certificate directly. One of the simple procedures is to transform Hierarchical PKI to a Binary tree using codeword algorithm[150][151] so that we can build the path without any ambiguity.

7.3.1 The Code Word Algorithm

The Code Word algorithm can be implemented on single PKI. According to the algorithm, the representation of a general PKI (ordered) tree T is obtained by transforming T into a binary tree T’. The transformation can be performed as follows:

1. For each node u of T, there is an internal node u’ of T’ associated with it.

2. If u is an external node of T and does not have a sibling immediately following it, then u’ is an external node of T’.

121
3. If u is an internal node of T and v is the first child of u in T, then v′ is the left child of u′ in T′.

4. If node v has a sibling w immediately following it, then w′ is the right child of v′ in T′.

![Figure 7.3: Conversion of Hierarchical structure to a binary tree](image)

In Figure 7.3, T is the input tree for the codeword algorithm and T′ is the output which is an equivalent binary tree.

In the binary tree T′, the left link is labeled with bit 0 and the right link with bit 1. Each CA node in the hierarchy holds a codeword consisting of the accumulated 0-1 sequence from the root to the target node in question. Each parent CA assigns to its subordinate CAs with its codeword plus a new distinguished code. The number of '0's in the codeword represents the level in the tree. Since the depth of the nodes increases after transforming the general tree to a binary tree, the path verification time also increases.

The methods of Forward certificate path construction and Reverse path construction(Codeword algorithm) are implemented using Java with OpenSSL tool. It can be seen from Table 7.1 and Figure 7.4 that, the time required for certificate verification using Forward path construction method is less than that of the Reverse path construction method.
Table 7.1: Time required for certificate verification using Forward path construction and Reverse path construction method (using codeword algorithm)

<table>
<thead>
<tr>
<th>No. of certificate verifications</th>
<th>Verification time using Forward Path construction (in milliseconds)</th>
<th>Verification time using Codeword algorithm (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>620</td>
<td>1090</td>
</tr>
<tr>
<td>20</td>
<td>1250</td>
<td>1560</td>
</tr>
<tr>
<td>30</td>
<td>1920</td>
<td>2030</td>
</tr>
<tr>
<td>40</td>
<td>2500</td>
<td>2570</td>
</tr>
<tr>
<td>50</td>
<td>2950</td>
<td>3220</td>
</tr>
<tr>
<td>60</td>
<td>3400</td>
<td>4600</td>
</tr>
<tr>
<td>70</td>
<td>3850</td>
<td>5260</td>
</tr>
<tr>
<td>80</td>
<td>4350</td>
<td>5910</td>
</tr>
<tr>
<td>90</td>
<td>4850</td>
<td>6580</td>
</tr>
<tr>
<td>100</td>
<td>5330</td>
<td>7230</td>
</tr>
</tbody>
</table>
7.4 Proposed method of Certificate Path Verification

In business transactions, the method of building a trusted path between the trusted anchor and target entity constructs a path as each certificate is retrieved from a repository via LDAP (Lightweight Directory Access Protocol), a protocol employed for repository access operations. The simplicity of such a method resides in the fact that only one certification path is possible in the case of Hierarchical PKI.

In the proposed method, the process of certificate path verification needs a local cache stored in the client environment, containing all certificates that constitute the path between the user's issuing CA and its hierarchy root CA, including the latter.

A Certificate chain is a structure composed of a set of certificates, from the CA immediately below the root CA to the user, signed and enveloped by the root CA of the hierarchy. The certificate chain may become too long and there is redundancy in path determination resulting in more time. To reduce time required for certificate path verification, an efficient method for path processing and validation
is proposed. This method eliminates the above said problems of certificate chains. In the proposed method, during the first verification of any certificate, the verifier builds the path and stores it in its local cache. In the subsequent verification of the same certificate, the verifier will first determine whether the path for the target certificate is already constructed and found in the local cache. If the certificate path is found in the local cache, no need of constructing the path once again, thus avoiding the redundancy in constructing the path and only validation of the certificates is required. This reduces certificate path verification time significantly. If the certificate path is not found in the local cache, construct the path, validate the certificates and store it in the local cache.

The algorithm for the proposed method is given below:

Retrieve the certificate of the user to be authenticated from the repository
If the path of the certificate is found in the local cache
{
    Validate the path
    Terminate the process
}
Else
{
    Determine the Distinguished Name(DN) of the certificate’s issuing CA;
    Retrieve the certificate of the issuing CA;
    While(The certificate of the issuing CA is not self-signed)
    {
        Validate the certificate
        Include the certificate in the path
        Retrieve the certificate of the issuing CA
    }
If the certificate of the issuing CA is self-signed {
    Announce successful construction of the path
    Store the path in the cache
}
Else {
    Announce unsuccessful construction of the path
    Terminate the process
}

The process diagram for path verification using the proposed method is shown in Figure 7.5.
7.5 Implementation and Experimental Results

The proposed method of path verification is implemented using Java with OpenSSL tool. Table 7.2 and Figure 7.6 show the comparison of Normal method (i.e., Forward path verification method) with the proposed method for different number of certificate verifications. We can observe that the verification time in the proposed method reduces significantly as compared to the normal method. As shown in Table 7.2 and Figure 7.6, the proposed method gives better performance when the cache hit is more.
Table 7.2: Time required for certificate verification using normal method and the proposed method

<table>
<thead>
<tr>
<th>No. of certificate verifications</th>
<th>Verification time using normal method (in milliseconds)</th>
<th>Verification time using Proposed method with 25% cache hit</th>
<th>Verification time using Proposed method with 50% cache hit</th>
<th>Verification time using Proposed method with 75% cache hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>620</td>
<td>465</td>
<td>310</td>
<td>155</td>
</tr>
<tr>
<td>20</td>
<td>1250</td>
<td>937.5</td>
<td>625</td>
<td>312.5</td>
</tr>
<tr>
<td>30</td>
<td>1920</td>
<td>1440</td>
<td>960</td>
<td>480</td>
</tr>
<tr>
<td>40</td>
<td>2500</td>
<td>1875</td>
<td>1250</td>
<td>625</td>
</tr>
<tr>
<td>50</td>
<td>2950</td>
<td>2212.5</td>
<td>1475</td>
<td>737.5</td>
</tr>
<tr>
<td>60</td>
<td>3400</td>
<td>2550</td>
<td>1700</td>
<td>850</td>
</tr>
<tr>
<td>70</td>
<td>3850</td>
<td>2887.5</td>
<td>1925</td>
<td>962.5</td>
</tr>
<tr>
<td>80</td>
<td>4350</td>
<td>3262.5</td>
<td>2175</td>
<td>1087.5</td>
</tr>
<tr>
<td>90</td>
<td>4850</td>
<td>3637.5</td>
<td>2425</td>
<td>1212.5</td>
</tr>
<tr>
<td>100</td>
<td>5330</td>
<td>3997.5</td>
<td>2665</td>
<td>1332.5</td>
</tr>
</tbody>
</table>
7.6 Summary

In Hierarchical PKI, certificate path is unidirectional, so certificate path development and validation is simple and straightforward. Forward path verification is the most popular technique of building certificate path in Hierarchical PKIs. In this chapter, a novel method of certificate path verification is proposed. The proposed method uses a local cache on the client side with the Forward path verification technique so that it gives better performance than that of the normal Forward path verification technique for certificate path verification. The experiment is conducted by considering 25%, 50% and 75% cache hit. It is observed that, as the cache hit increases, the path verification time decreases.

The next chapter explains an efficient method developed for certificate path verification in Mesh or Peer-to-Peer PKIs.