1. Defect Report Number: DR 222
   Title: Certificate Policy Mapping

2. Source: Santosh Chokhani, (US) and Tim Moses, (Canada)

3. Addressed to: ISO/IEC JTC 1/SC6 and ITU-T SG7

4. (a)
   (b)

5. Date circulated to WG Secretariat:

6. Deadline for Response from Editor:

7. Defect Report Concerning:
ITU-T X.509 (1997) | ISO/IEC 9594-8:1997

8. Qualifier:

Error and Omission

9. References in Document:

Clause 12.1 (omission); 12.4.3 (error)

10. Nature of Defect:

Based on the PKI implementation and experience we have learned of the following defects in the X.509 certificate policy and certificate policy related path processing:

a) Policy Mapping Defect: The current policy processing logic and the requirement to assert the issuer domain policy in a certificate means a domain ends up accepting a certificate it intended to reject. Specifically, this situation arises when policy mapping is inhibited and an intermediate CA certifies another CA using policy mapping.
b) Policy Accumulation Defect: The current policy mapping logic means that acceptable policies are accumulated as opposed to substituted when the policies are mapped.
c) Criticality Based Policy Processing: The current policy processing logic requires some processing to be done when the **certificatePolicy** extension is non-critical and additional processing when the extension is critical. Experience has revealed that the business requirements are better met with a single processing logic regardless of the criticality of the extension.

11. Solution recommended by the Source:

Add the following to Section 12.1:

Certificate policy

The authentication framework contains three types of entity: the certificate user, the certification authority and the certificate subject (or end-entity). Each entity operates under obligations to the other two entities and, in return, enjoys limited warranties offered by them. These obligations and warranties are defined in a certificate policy. A certificate policy is a document (usually in plain-language). It can be referenced by a unique identifier, which may be included in the certificate policies extension of the certificate issued by the certification authority, to the end-entity and upon which the certificate user relies. A certificate may be issued in accordance with one or more than one policy. Definition of the policy, and assignment of the identifier, are performed by a policy authority. And the set of policies administered by a policy authority is called a policy domain. All certificates are issued in accordance with a policy, even if the policy is neither recorded anywhere nor referenced in the certificate. The standard does not prescribe the style or contents of the certificate policy.

The certificate user may be bound to its obligations under the certificate policy by the act of importing an authority public key and using it as a trust anchor, or by relying on a certificate that includes the associated policy identifier. The certification authority may be bound to ***its*** obligations under the policy by the act of issuing a certificate that includes the associated policy identifier. And, the end-entity may be bound to ***its*** obligations under the policy by the act of requesting and accepting a certificate that includes the associated policy identifier and by using the corresponding private key. Implementations that do not use the certificate policies extension should achieve the required binding by some other means.

For an entity to simply declare conformance to a policy does not generally satisfy the assurance requirements of the other entities in the framework. They require some reason to believe that the other parties operate a reliable implementation of the policy. However, if explicitly so stated in the policy, certificate users may accept the certification authority s assurances that its end-entities agree to be bound by their obligations under the policy, without having to confirm this directly with them. This aspect of certificate policy is outside the scope of the standard.

A certification authority may place limitations on the use of its certificates, in order to control the risk that it assumes as a result of issuing certificates. For instance, it may restrict the community of certificate users, the purposes for which they may use its certificates and/or the type and extent of damages that it is prepared to make good in the event of a failure on its part, or that of its end-entities. These matters should be defined in the certificate policy.

Additional information, to help affected entities understand the provisions of the policy, may be included in the certificate policies extension in the form of policy qualifiers.

Cross-certification

A certification authority may be the subject of a certificate issued by another certification authority. In this case, the certificate is called a cross-certificate, the certification authority that is the subject of the certificate is called the subject certification authority and the certification authority that issues the cross-certificate is called an intermediate certification authority (see Figure 1). Both the cross-certificate and the end-entity s certificate may contain a certificate policies extension.

The warranties and obligations shared by the subject certification authority, the intermediate certification authority and the certificate user are defined by the certificate policy identified in the cross-certificate, in accordance with which the subject certification authority may act as, or on behalf of, an end-entity. And the warranties and obligations shared by the certificate subject, the subject certification authority and the intermediate certification authority are defined by the certificate policy identified in the end-entity s certificate, in accordance with which the intermediate certification authority may act as, or on behalf of, a certificate user.

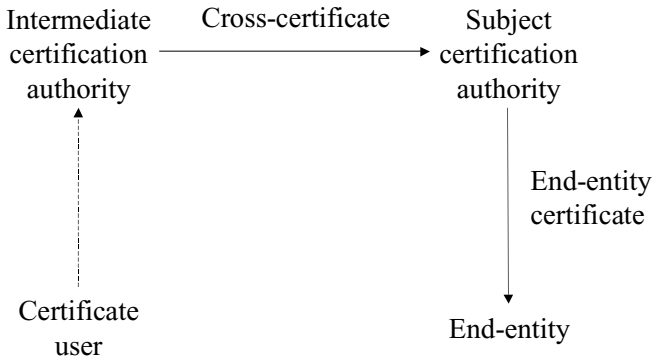

Figure 1 - Cross-certification

A certification path is said to be valid under the set of policies that are common to all certificates in the path.

An intermediate certification authority may, in turn, be the subject of a certificate issued by another certification authority, thereby creating certification paths of length greater than two certificates. And, since trust suffers dilution as certificate paths grow in length, controls are required to ensure that end-entity certificates with an unacceptably low associated trust level will be rejected by the certificate user. This is part of the function of the certification path processing procedure.

In addition to the situation described above, there are two special cases to be considered:

1. the certification authority does not use the certificate policies extension to convey its policy requirements to certificate users; and

2. the certificate user or intermediate certification authority delegates the job of controlling policy to the next authority in the path.

In the first case, the certificate should not contain a certificate policies extension at all. As a result, the set of policies under which the path is valid will be null. But, the path may be valid nonetheless. Certificate users must still ensure that they are using the certificate in conformance with the policies of the authorities in the path.

In the second case, the certificate user or intermediate certification authority should include the special value *any-policy* in the *initial-policy-set* or cross-certificate. Where a certificate includes the special value *any-policy*, it should not include any other certificate policy identifiers. The identifier *any-policy* should not have any associated policy qualifiers.

The certificate user can ensure that all its obligations are conveyed in accordance with the standard by setting the *initial-explicit-policy* indicator. In this way, only authorities that use the standard certificate policies extension as their way of achieving binding are accepted in the path, and certificate users have no additional obligations. Because authorities also attract obligations when they act as, or on behalf of, a certificate user, they can ensure that all their obligations are conveyed in accordance with the standard by setting requireExplicitPolicy in the cross-certificate.

Policy mapping

Some certification paths may cross boundaries between policy domains. The warranties and obligations according to which the cross-certificate is issued may be materially equivalent to some or all of the warranties and obligations according to which the subject certification authority issues certificates to end-entities, even though the policy authorities under which the two certification authorities operate may have selected different unique identifiers for these materially equivalent policies. In this case, the intermediate certification authority may include a policy mappings extension in the cross-certificate. In the policy mappings extension, the intermediate certification authority assures the certificate user that it will continue to enjoy the familiar warranties, and that it should continue to fulfill its familiar obligations, even though subsequent entities in the certification path operate in a different policy domain. The intermediate certification authority should include one or more mappings for each of a subset of the policies under which it issued the cross-certificate, and it should not include mappings for any other policies. If one or more of the certificate policies according to which the subject certification authority operates is identical to those according to which the intermediate certification authority operates (i.e. it has the same unique identifier), then these identifiers should be excluded from the policy mapping extension, but included in the certificate policies extension.

Policy mapping has the effect of converting all policy identifiers in certificates further down the certification path to the identifier of the equivalent policy, as recognized by the certificate user.

Policies should not be mapped either to or from the special value *any-policy*.

Certificate users may determine that certificates issued in a policy domain other than its own should not be relied upon, even though a trusted intermediate certification authority may determine its policy to be materially equivalent to its own. It can do this by setting the *initial-policy-mapping-inhibit input* to the path validation procedure. Additionally, an intermediate certification authority may make a similar determination on behalf of its certificate users. In order to ensure that certificate users correctly enforce this requirement, it can set inhibitPolicyMapping in a policy constraints extension.

Certification path processing

The certificate user faces a choice between two strategies:

1. it can require that the certification path be valid under at least one of a set of policies pre-determined by the user; or

2. it can ask the path validation module to report the set of policies for which the certification path is valid.

The first strategy may be most appropriate when the certificate user knows, a priori, the set of policies that are acceptable for its intended use.

The second strategy may be most appropriate when the certificate user does not know, a priori, the set of policies that are acceptable for its intended use.

In the first instance, the certification path validation procedure will indicate the path to be valid only if it is valid under one or more of the policies specified in the *initial-policy-set*, and it will return the sub-set of the *initial-policy-set* under which the path is valid. In the second instance, the certification path validation procedure may indicate that the path is invalid under the *initial-policy-set*, but valid under a disjoint set: the *authorities-constrained-policy-set*. Then the certificate user must determine whether its intended use of the certificate is consistent with one or more of the certificate policies under which the path ***is*** valid. By setting the *initial-policy-set* to *any-policy*, the certificate user can cause the procedure to return a valid result if the path is valid under any (unspecified) policy.

Self-issued certificates

There are three circumstances under which a certification authority may issue a certificate to itself:

1. as a convenient way of encoding its public key for communication to, and storage by, its certificate users;

2. for certifying key usages other than certificate and CRL signing (such as time-stamping); and

3. for replacing its own expired certificates.

These types of certificate are called self-issued certificates, and they can be recognized by the fact that the issuer and subject names present in them are identical. For purposes of path validation, self-issued certificates of type one are verified with the public key contained in them, and if they are encountered in the path, they shall be ignored.

Self-issued certificates of type two may only appear as end certificates in a path, and shall be processed as end certificates.

Self-issued certificates of type three (also known as self-issued intermediate certificates) may appear as intermediate certificates in a path. As a matter of good practice, when replacing a key that is on the point of expiration, a certification authority should request the issuance of any in-bound cross-certificates that it requires for its replacement public key before using the key. Nevertheless, if self-issued certificates are encountered in the path, they shall be processed as intermediate certificates, with the following exception: they do not contribute to the path length for purposes of processing the pathLenConstraint component of the basicConstraints extension and the *skip-certificates* values associated with the *policy-mapping-inhibit-pending* and *explicit-policy-pending* indicators.

Replace section 12.4.3 with the following:

12.4.3 Certification path processing procedure

Certification path processing is carried out in a system which needs to use the public key of a remote end entity, e.g. a system which is verifying a digital signature generated by a remote entity. The certificate policies, basic constraints, name constraints, and policy constraints extensions have been designed to facilitate automated, self-contained implementation of certification path processing logic.

The following is an outline of a procedure for validating certification paths. A conformant implementation shall be functionally equivalent to the external behaviour resulting from this procedure. But, the algorithm used by a particular implementation to derive the correct output(s) from the given inputs is not standardized.

The inputs to the certification path processing procedure are:

a) a set of certificates comprising a certification path;

b) a trusted public key value or key identifier (if the key is stored internally to the certification path processing module), for use in verifying the first certificate in the certification path;

c) an *initial-policy-set* comprising one or more certificate policy identifiers, indicating that any one of these policies would be acceptable to the certificate user for the purposes of certification path processing; this input can also take the special value *any-policy*;

d) an *initial-explicit-policy* indicator value, which indicates whether an acceptable policy identifier must appear in the certificate policies extension field of all certificates in the path;

e) an *initial-policy-mapping-inhibit* indicator value, which indicates whether policy mapping is forbidden in the certification path; and

f) the current date/time (if not available internally to the certification path processing module).

The values of c), d), and e) will depend upon the policy requirements of the user-application combination that needs to use the certified end-entity public key.

The outputs of the procedure are:

a) an indication of success or failure of certification path validation;

b) if validation failed, a diagnostic code indicating the reason for failure;

c) The set of authorities-constrained policies and their associated qualifiers in accordance with which the certification path is valid, , or the special value *any-policy*;

d) The set of user-constrained policies, formed from the intersection of the *authorities-constrained-policy-set* and the *initial-policy-set*;

e) *explicit-policy-indicator*, indicating whether the certificate user or an authority in the path requires that an acceptable policy be identified in every certificate in the path; and

f) details of any policy mapping that occurred in processing the certification path.

> NOTE — If validation is successful, the certificate-using system may still choose not to use the certificate as a result of values of policy qualifiers or other information in the certificate.

The procedure makes use of the following set of state variables:

a) *authorities-constrained-policy-set:* A table of policy identifiers and qualifiers from the certificates of the certification path (rows represent policies, their qualifiers and mapping history, and columns represent certificates in the certification path);

b) *permitted-subtrees*: A set of subtree specifications defining subtrees within which all subject names in subsequent certificates in the certification path must fall, or may take the special value *unbounded*;

c) *excluded-subtrees*: A (possibly empty) set of subtree specifications (each comprising a subtree base name and maximum and minimum level indicators) defining subtrees within which no subject name in a subsequent certificate in the certification path may fall;

d) *explicit-policy-indicator*: Indicates whether an acceptable policy must be explicitly identified in every certificate in the path;

e) *path depth*: An integer equal to one more than the number of certificates in the certification path for which processing has been completed;

f) *policy-mapping-inhibit-indicator*: Indicates whether policy mapping is inhibited;

g) *pending-constraints*: Details of explicit-policy and/or inhibit-policy-mapping constraints which have been stipulated but have yet to take effect. There are two one-bit indicators called *explicit-policy-pending*, and *policy-mapping-inhibit-pending* together with, for each, an integer called *skip-certificates* which gives the number of certificates yet to skip before the constraint takes effect.

The procedure involves an initialization step, followed by a series of certificate-processing steps. The initialization step comprises:

a) Write *any-policy* in the zeroth and first columns of the zeroth row of the *authorities-constrained-policy-set* table;

b) Initialize the *permitted-subtrees* variable to *unbounded*;

c) Initialize the *excluded-subtrees* variable to an empty set;

d) Initialize the *explicit-policy-indicator* to the *initial-explicit-policy* value;

e) Initialize *path-depth* to one;

f) Initialize the *policy-mapping-inhibit-indicator* to the *initial-policy-mapping-inhibit* value;

g) Initialize the two *pending-constraints* indicators to unset.

Each certificate is then processed in turn, starting with the certificate signed using the input trusted public key. The last certificate is considered to be the end certificate; any other certificates are considered to be intermediate certificates.

The following checks are applied to a certificate:

a) Check that the signature verifies, that dates are valid, that the certificate subject and certificate issuer names chain correctly, and that the certificate has not been revoked.

b) For an intermediate certificate, if the basic constraints extension field is present in the certificate, check that the **cA** component is present and set to true. If the **pathLenConstraint** component is present, check that the current certification path does not violate that constraint (ignoring intermediate self-issued certificates).

c) If the certificate policies extension is not present, then set the *authorities-constrained-policy-set* to null by deleting all rows from the *authorities-constrained-policy-set* table.

d) If the certificate policies extension is present and the value in *authorities-constrained-policy-set*[0, *path-depth*] is not *any-policy* and the value in the extension is not *any-policy*, then set the *authorities-constrained-policy-set* to the intersection of the *authorities-constrained-policy-set* with the set of policies present in the certificate. To do this, first add the policy qualifiers from the extension to the *authorities-constrained-policy-set* table by, for each policy identifier value in the extension, locate all rows in the *authorities-constrained-policy-set* table whose [*path-depth*] column entry contains the same value as that in the extension and attach the policy qualifiers from the extension to the policy identifiers in the table, then delete all rows for which the [*path-depth*] column did not contain one of the values in the extension.

e) If the certificate policies extension is present and the value in *authorities-constrained-policy-set*[0, *path-depth*] is not *any-policy* but the value in the extension is *any-policy*, then attach the policy qualifier (if present) from the extension to each policy identifier value in the [*path-depth*] column of the *authorities-constrained-policy-set* table.

f) If the certificate policies extension is present and the value in authorities-constrained-policy-set[0, path-depth] is any-policy, then set the authorities-constrained-policy-set to the intersection of the authorities-constrained-policy-set with the set of policies present in the certificate. To do this, add new rows to the table by duplicating the zeroth row a number of times equal to the number of policy identifiers in the extension minus one, and write the policy identifiers and qualifiers from the extension in authorities-constrained-policy-set[0, path-depth] and the path-depth column of each new row (this step must be performed even if the value in the extension is any-policy).

g) If the certificate is not an intermediate self-issued certificate, check that the subject name is within the name-space given by the value of permitted-subtrees and is not within the name-space given by the value of excluded-subtrees.

For an intermediate certificate, the following constraint recording actions are then performed, in order to correctly set up the state variables for the processing of the next certificate:

a) If the **nameConstraints** extension with a **permittedSubtrees** component is present in the certificate, set the *permitted-subtrees* state variable to the intersection of its previous value and the value indicated in the certificate extension.

b) If the **nameConstraints** extension with an **excludedSubtrees** component is present in the certificate, set the *excluded-subtrees* state variable to the union of its previous value and the value indicated in the certificate extension.

c) If policy-mapping-inhibit-indicator is set:

— process any policy mappings extension by, for each mapping identified in the extension, locate all rows in the *authorities-constrained-policy-set* table whose [*path-depth*] column entry is equal to the issuer domain policy value in the extension and delete the row.

d) If *policy-mapping-inhibit-indicator* is not set:

process any policy mappings extension by, for each mapping identified in the extension, locate all rows in the *authorities-constrained-policy-set* table whose [*path-depth*] column entry is equal to the issuer domain policy value in the extension, and write the subject domain policy value from the extension in the [*path-depth*+1] column entry of the same row. If the extension maps an issuer domain policy to more than one subject domain policy, then the affected row must be copied and the new entry added to each row. If the value in *authorities-constrained-policy-set*[0, *path-depth*] is *any-policy*, then write each issuer domain policy identifier from the policy mappings extension in the [*path-depth*] column, making duplicate rows as necessary and retaining qualifiers if they are present, and write the subject domain policy value from the extension in the [*path-depth*+1] column entry of the same row.

— if the *policy-mapping-inhibit-pending* indicator is set and the certificate is not self-issued, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set the *policy-mapping-inhibit-indicator*.

If the **inhibitPolicyMapping** constraint is present in the certificate, perform the following. For a **SkipCerts** value of 0, set the *policy-mapping-inhibit-indicator*. For any other **SkipCerts** value, set the *policy-mapping-inhibit-pending* indicator, and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *policy-mapping-inhibit-pending* indicator was already set).

e) For any row not modified in either step c) or d), above (and every row in the case that there is no mapping extension present in the certificate), write the policy identifier from [path-depth] column in the [path-depth+1] column of the row.

f) Increment path-depth.

For all certificates, the following actions are then performed:

a) If explicit-policy-indicator is not set:

— if the *explicit-policy-pending* indicator is set and the certificate is not a self-issued intermediate certificate, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set *explicit-policy-indicator*.

If the **requireExplicitPolicy** constraint is present in the certificate perform the following. For a **SkipCerts** value of 0, set *explicit-policy-indicator*. For any other **SkipCerts** value, set the *explicit-policy-pending* indicator, and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *explicit-policy-pending* indicator was already set).

For the end-certificate, the following actions are then performed:

a) If explicit-policy-indicator is set, check that the authorities-constrained-policy-set table is not empty.If any of the above checks were to fail, then the procedure shall terminate, returning a

failure indication, an appropriate reason code, explicit-policy-indicator and null values in the user-constrained-policy-set and the authorities-constrained-policy-set table.

If none of the above checks were to fail on the end certificate, then the *user-constrained-policy-set* shall be calculated by making a copy of the *authorities-constrained-policy-set* table, locating the left-most column whose zeroth row does not contain *any-policy* and deleting all rows which do not contain one of the identifiers in the *initial-policy-set* in this column. If all the columns contain *any-policy* in the zeroth row, then the table shall not be modified. Then the procedure shall terminate, returning a success indication together with the *explicit-policy-indicator*, the *authorities-constrained-policy-set* table and the *user-constrained-policy-set*.

The authorities-constrained-policy-set is the left-most column in the authorities-constrained-policy-set whose zeroth row does not contain the identifier any-policy. If there is no column that qualifies, then the authorities-constrained-*policy-set* is *any-policy*.