

How Trust Had a Hole Blown In It

The Case of X.509 Name Constraints

David Chadwick, University of Kent. d.w.chadwick@kent.ac.uk

Abstract

A different interpretation of the Name Constraints extension to that intended by ISO/ITU-T in its 1997 edition of X.509, was made by the IETF PKIX group in its certificate profile (RFC 2459). This has led to conflicting implementations and misalignment of the standard and its profile. This paper reviews the history of the Name Constraints extension, and how it has evolved to the present day from an original concept first described in Privacy Enhanced Mail. The paper concludes by suggesting possible ways forward to resolve this unfortunate conflict.

1. Introduction

The name constraints extension in X.509 was first introduced in the 1997 edition of X.509 [2]. But its history goes back further than that, back in fact to the early 1990's and Privacy Enhanced Mail (PEM) [1]. The extension has evolved over time since its first introduction, and, due to lack of precision in the original X.509 definition, varying interpretations of its meaning have evolved. This has now led to a divergence between the Internet PKIX group's profile of X.509 [3] and the latest edition of the X.509 standard [4, 8], which is about to be merged and published as X.509 (2005). This matters, because some certificates accepted as valid by one interpretation, will be treated as invalid by the other, and vice versa.

This paper tries to untangle the confusion surrounding the name constraints extension, and understand how we have got into the situation we are in today, where the X.509 standard and the RFC 3280 profile [5] disagree about both the syntax and the semantics of this extension. This paper then poses the question, "Where do we go from here?". This is still an unanswered question, but some possibilities are suggested in the final section of this paper. This will no doubt provoke some further discussion of the problem both within the standards settings groups and with implementers, and this might help to draw this misalignment to a successful conclusion.

This paper has been written mostly from the documents (standards and draft standards) published during the last 12 years, but also partly from the memories of those working in this area at the time [9]. It therefore could contain errors in the interpretation of what was actually published. However it is a best efforts attempt at trying to understand how the current problem has arisen. It also provides an interesting historical case study of the standardisation process which shows how original intentions evolve with time, but due to imprecise specifications, and a lack of dialogue, different conclusions about these intentions are reached by different groups of people. The contents of this paper are as follows. Section 2 describes a motivating example to show how and when name constraints can be useful. Subsequent sections refer to this to show

how it can (or cannot) be supported with the various flavours of name constraints as it has evolved with time. Section 3 provides a history of the early developments of the name constraints extension, up until 2000. Section 4 provides a more recent history of the extension, from 2001 to the current date. Section 5 then concludes and suggests answers to the question “Where do we go from here?” This might help to guide subsequent discussions on this topic.

2. A Motivating Example (or two)

Suppose organisations X, Y and Z all operate CAs, with a DNs of {cn=CA, o=X, c=GB}, {ou=admin, o=Y} and {o=Z, c=US}. Assume each issues certificates to its employees, who all have DNs under their respective organisational arcs {o=X, c=GB}, {o=Y} and {o=Z, c=US}. Some of the CAs may also issue certificates to other people, e.g. contractors, subsidiaries, business partners etc. We assume that these are named under different arcs to those of their employees.

Scenario 1. Suppose that any two of these three organisations wish to cross certify each other, and constrain the certificates they wish to trust to only those issued to their employees. This is easily achieved by placing a name constraints extension in each cross certificate issued to X, Y or Z indicating that only certificates starting with a DN of {o=X, c=GB}, {o=Y} or {o=Z, c=US} respectively will be trusted. Any other certificates issued to contractors, business partners etc. will not be trusted, providing their DNs are not in the employee’s name space.

Scenario 2. Suppose one of the organisations only wishes to trust a subset of the certificates issued to the employees of another of the CAs, for example, to employees within the marketing department. This can be achieved by using a name constraints DN of {OU=marketing, o=X, c=GB}, {OU=marketing, o=Y} or {OU=marketing, o=Z, c=US} respectively.

Scenario 3. Suppose a Bridge (or some other) CA exists that has cross certified each of the three organisational CAs, so that it trusts all the certificates issued by all of these CAs. Suppose however that one of these organisational CAs wants to limit the certificates that are deemed to be trustworthy via the Bridge CA e.g. X only wants to trust certificates issued by Y to its employees and not any certificates issued by Z. In this case, X issues a cross certificate to the Bridge CA that has a name constraints of {o=Y}, with a parameter to indicate that the first certificate in the chain (that of the Bridge CA) is not to be bound by the name constraints rule.

3. An Early History of Name Constraints (93-2000)

“Name constraints” was originally introduced as a concept to limit the X.509 certificates that could be issued to support Privacy Enhancements for Internet Electronic Mail (PEM). As RFC 1422 states below, the rationale was to try to ensure that each CA only issued certificates containing globally unique distinguished names, since this was a fundamental requirement of the X.500 standard, of which X.509 was an integral part.

RFC 1422 [1] states:

To complete the strategy for ensuring uniqueness of DNs, there is a DN subordination requirement levied on CAs. In general, CAs are expected to sign certificates only if the subject DN in the certificate is subordinate to the issuer (CA) DN. This ensures that certificates issued by a CA are syntactically constrained to refer to subordinate entities in the X.500 directory information tree (DIT), and this further limits the possibility of duplicate DN registration.

There was much debate during this period about how globally unique distinguished names could be formed. Questions included: who would be the global naming authorities; who would manage the root of the Directory Information Tree (DIT); and what would be the contents of distinguished names, in terms of the allowed attribute types and values? There were no conclusive answers to this debate when the PEM RFCs were published, and so PEM neatly sidestepped this issue for user certificates, by saying that they would be named subordinate to the names of the CAs, assuming that each CA would have a globally unique name. This mindset tended to continue in the PKIX working group in subsequent years, and still continues in some quarters today, where some experts believe that a subject DN can only be regarded as globally unique if it is assumed to be subordinate to, or used in conjunction with, the name of the issuing CA. This name subordination was never an assumption of X.500, which instead, required that each user DN would be globally unique in its own right.

At the time the PEM standard was being released, in 1993, the second edition of

X.509 was also being released. Unfortunately X.509(93) did not contain any technical mechanism to indicate any sort of constraints on the subject names that a CA could place in the V2 certificates that it issued. A CA could issue a certificate with any valid subject DN. Thus the PEM standard had to ensure this constraint on subject names through procedural means that were placed on the CA (by the above wording in the PEM standard) and by a technical requirement to check name subordination during certificate path validation. Whilst these mechanisms are sufficient to enforce name subordination, they are very inflexible, since they can only cater for Scenario 1 above (and not for 2 and 3) since there is no information in the X.509 certificate to indicate how and when name subordination rules should be applied (or not). Consequently, as soon as X.509 (93) was released, work started on defining the policy rules that could be placed inside certificates, in order to allow much more flexibility in determining which certificates should be trusted. This work culminated in edition three of X.509, published in 1997 [2]. The primary work on edition three of X.509 was the technical definition of the protocol elements inside certificates that would support the policies and procedures of a CA. This was achieved by adding extensions to the X.509 V2 certificate format, to produce the V3 certificate format that we all use today. (Since V3 certificates are infinitely extensible there has never been a requirement since 1997 to define a V4 certificate format.)

During the four years that it took to produce the 1997 edition of X.509, several working drafts were produced. The name constraints extension was there

from the outset, and its syntax and semantics remained constant until 1996. Annex 1 shows the name constraints definition in the output produced by the Orlando meeting in December 1994 [6] and the Ottawa meeting in 1995 [7]. The only difference, shown by the underlined text, was some more explanation of the meanings of the various fields added in 1995. One can see that a primary requirement was to satisfy PEM's concerns to constrain which names a CA could issue to its subjects, but also to add greater flexibility in order to cater for all three scenarios described above (and more!). There are three notable features of this definition.

- Firstly, the only name form that was supported was the X.500 distinguished name (DN) and the way that a name space was constrained was via the subtreeSpecification directly imported from the X.501(93) standard. The subtreeSpecification allows *any* arbitrary DIT subtree to be defined, including chopped subtrees which define branches of the top level subtree that are to be chopped off. (Note that X.501 allows filtered (disjoint) subtrees as well, but X.509 stated that filtered subtrees should not be permitted in name constraints). The subtreeSpecification allows us to easily cater for Scenarios 1 and 2 above.
- Secondly, there were no loopholes. Any user certificate that did not fall within the scope of a specified name constraint, should not to be regarded as valid. The semantics of the extension could therefore be stated as **“every name that is not explicitly trusted is untrusted”** i.e. the name constraint specifies a white list of

trusted subtrees. Since all constrained names were based on distinguished names, there was no possibility that a constrained certificate could contain other than a name in X.500 DN format. This feature ensured that certificates issued to sub-contractors, business partners etc. who had different DNs would not be trusted inadvertently.

- Thirdly, not all certificates issued by subordinate CAs need be constrained. Two control mechanisms were provided for the certifying CA to specify which certificates did not fall within the scope of the name constraints extension. The certifying CA could either specify a set of certificate policies to which this constraint applied, or could specify how many CAs in the chain should be skipped before the constraint applied. This skipping mechanism allows us to cater for Scenario 3 above.

The net result of this extension was that the issuing (superior) CA could tightly control which (subject names in) certificates issued by cross certified (subordinate) CAs should be trusted. Any relying party (RP) using the superior CA as its root of trust could be sure that certificate path validation software would not trust any certificate falling outside these name constraints. We thus had a watertight trust model.

Another extension was also being defined during this period, entitled the subject alternative name field. This extension defined *“one or more alternative names, using any of a variety of name forms, for the entity that is bound by the CA to the certified public key”*. Several possible alternative name forms for the certificate

subject were specified, including a DNS name, an RFC822 email address and an X.400 OR address. This extension underwent some growth during this period, starting out with just four alternative name forms and eventually ending up with nine. Its intention was to allow a certificate subject to have a variety of names in different formats, because it was recognized in the mid 1990s that there was not going to be a global X.500 directory service. If the X.509 standard could not cater for subjects with other name forms besides X.500 ones, then this would significantly limit its scope and applicability. Thus X.509 should support alternative name forms. In order to make the extension fully extensible and able to cater for future name forms that currently do not exist, the alternative name can also be an *other* name form, which is identified by a globally unique object identifier. Thus it is likely that a relying party might encounter a subject alternative name form that it is not able to recognize. In order to cater for this, the definition of this extension included the text “*a certificate-using system is permitted to ignore any name with an unrecognized or unsupported name form*”. The implicit assumption was however, that this was an alternative name for the subject, not a replacement name, and the subject would always have an X.500 distinguished name, even if it did not have an entry in an X.500 directory service. We shall see later that this ability to ignore unrecognized name forms probably indirectly led to the erosion of the trust model built into name constraints.

Yet another certificate extension that was being defined through this period was the one that eventually became known as the basic constraints extension. This had

something of a Jekyll and Hyde life. Initially known in the PDAM [6] as the *CA or end entity indicator*, it had virtually the same syntax and semantics as the basic constraints extension used today. It then grew in significance in the DAM [7], when it changed its name to basic constraints and added a simplified name constraints capability to its syntax, specifically, the ability to specify the set of permitted subtrees in which all subsequent certificate subject names should fall.

Dramatic changes to the X.509 draft standard occurred in April 1996 at the Geneva meeting, precipitated by amongst other things the Canadian national ballot comment. The Canadian ballot comment proposed three things:

- to introduce the syntactic construct *GeneralName*, in order to group together into one super-type all the name forms in the subject alternative name field
- to add further capability to basic constraints in two ways, firstly by allowing denied subtrees as well as permitted subtrees to be specified; and secondly to replace the X.500 distinguished name type with the *GeneralName* super-type.
- to remove the name constraints extension since it was no longer needed, as its main purpose was now usurped by the enhanced basic constraints extension being proposed in this ballot comment.

The outcome of the resolution of the Canadian and other national ballot comments is well documented; it is the 1997 edition of X.509 (see Annex 2). Precisely what technical discussions were had in order to get there have now largely been forgotten with time, but several

things are clear. The Canadian introduction of the GeneralName super-type was accepted, and this was used to specify the subject alternative name extension. The changes to basic constraints were rejected, and this extension reverted to its original 1994 definition. However, the intention of the ballot comment was accepted in principle, by modifying the name constraints extension to match the proposed basic constraints extension. In other words, name constraints was modified by replacing the X.500 distinguished name type with the GeneralName super-type, and deleting the policy and skip certs controls that limited when the name constraints should apply. The intention of name constraints was still very clear, as stated in the first sentence of the description “*indicates a name space within which all subject names in subsequent certificates in a certification path must be located*”. It can be seen that its purpose was to tightly constrain the names that the subordinate or cross certified CA could put into the subject field of the certificates that it issued, and more than that, to constrain all additional subordinate CAs further along the certification path. Whereas the original name constraints allowed certain groups of certificates to be specifically excluded, via the skipCerts and policySet fields, the new definition did not. The semantics were very definitely “**every name that is not explicitly trusted is untrusted, with no exceptions**”. In other words, the original trust model still held true, but was even tighter than before, because Scenario 3 can no longer be supported. This tight trust model is further shown by the Certificate Path Processing Procedure in Section 12.4.3 of the 97 standard, which states:

The following checks are applied to a certificate:

.....

e) *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.*

If any of the above checks fails, the procedure terminates, returning a failure indication and an appropriate reason code.

Unfortunately, when the GeneralName syntax replaced the X.500 DN syntax in the name constraints extension, it was not as straightforward as simply replacing one syntax with another. The text describing the name constraints extension should have been significantly enhanced, because new possibilities now existed that did not before. Enhancements were needed in a number of ways. Firstly, how was the name constraints extension to handle general names that were not hierarchically structured, such as IP addresses. How could one specify permitted and excluded subtrees for non-hierarchical names? The answer was to exclude these name forms from being applicable to this extension, as is indicated by the text “*only those name forms that have a well-defined hierarchical structure may be used in these fields*”. Secondly, what was a relying party to do if there was a mismatch between the various subject alternative names in a certificate, and the name constraints extension in the issuing CA’s certificate? Several new possibilities now exist: (i) the subject’s alternative names are a subset of the name forms listed in the CA’s name constraints; (ii) the subject’s alternative names are a superset of the name forms

listed in the CA's name constraints; (iii) the subject's alternative names intersect with the name forms listed in the CA's name constraints; (iv) the subject's alternative names do not overlap with the name forms listed in the CA's name constraints; and (v) the subject's alternative names are identical to the name forms listed in the CA's name constraints. Unfortunately the standard is strangely quiet on this aspect. This is clearly a bug. The fact that appropriate wording was not included to reflect the change of syntax can be seen from the first sentence of the definition, which continued to state "*indicates a name space within which all subject names in subsequent certificates*". In fact, with the introduction of General Names, it does not indicate a single name space any longer, but possibly many different name spaces. How a relying party should behave when all these new possibilities present themselves can be resolved in one of two ways, either conjunctively or disjunctively. Conjunctive resolution would require all the name forms in the certificate to match the specified name constraints, whereas disjunctive resolution would require just one name form in the certificate to obey any one of the name constraints. When this issue was recently debated on the X.500 mailing list, the X.500 rapporteur stated "*I considered (subject) alt names to be truly alternate forms of the subject name in the certificate. That subject name had to be within the scope of any name constraints, if specified. If the subject name was in scope, the alternative name would be considered within scope. I don't think we, meaning the x.509 group, ever considered what to do for any other conditions*".

As soon as X.509 (97) was published, the IETF PKIX group started to work on their profile for X.509 public key certificates. The first version of this was published in 1999 as RFC 2459 [3]. In an attempt to guide implementers in their coding, it had to work out what the intended X.509 semantics were when there was a mismatch between the name forms in a subject's certificate and those in the name constraints extension of the issuing CA. Therefore RFC 2459 added the following two critical sentences to its specification "*Restrictions apply only when the specified name form is present. If no name of the type is in the certificate, the certificate is acceptable.*" Precisely why these sentences were added is not known. It might have been a best efforts interpretation of how the subject alternative names logic, that stated that unknown name forms could be safely ignored, applied to name constraints. On the other hand it might have been a poor attempt at resolving mismatches between name forms in subject names and name constraints.

Unfortunately, and perhaps without realizing it, the RFC 2459 wording was also flawed in two ways. Firstly it does not explicitly cover all the five cases listed above. Specifically what rule should apply when the certificate simultaneously has no name of the type specified in name constraints but also has a name of the type specified in name constraints (cases (ii) and (iii) above). Should it be trusted or not? But more importantly, it has introduced a potentially massive security hole in the trust relationship between the superior CA issuing the certificate with a name constraints extension and the subordinate (or cross certified) CA receiving it. In fact, it has completely reversed the X.509

trust model into one of “**every name form that is not explicitly untrusted is trusted**” i.e. name constraints now become black lists rather than white lists. For example, referring to Scenario 1 above, where organization X cross certifies organization Y, suppose that unknown to organization X, organization Y’s CA is somewhat untrustworthy, or it simply changes its rules, and decides it will issue certificates with other name forms as well as or instead of X.500 DNs, for example RFC822 names. A user, Freddie Fraudster (who may or may not be employed by Y), with the email address nice.guy@cheap.goods.com wants to obtain a certificate that will be trusted by organization X’s CA, so it asks organization Y’s CA to issue him with a certificate containing only his email address. Using the RFC 2459 semantics of “trust all except”, the certificate will be trusted by relying parties who have a root of trust in organization X’s CA. However, using the X.509 “untrust all except” semantics, the certificate will not be trusted. This reversal of semantics has now blown an unblockable hole in the trust relationship between the two CAs. The reason is that the number of subject alternative name forms is infinite, through using the *other* name form variant. Since it is impossible to list an infinite number of name forms, it is impossible to list all the name forms that are trusted (according to RFC 2459) or untrusted (according to X.509 (97)). Thus it is much safer for name constraints to contain white lists rather than black lists.

3. A Recent History of Name Constraints(2001-05)

Despite its publication in January 1999, the RFC 2459 trust hole and reversal of the X.509 trust semantics, went largely unnoticed by the X.509 standards body

for several years. So much so that the third edition of X.509 was published in 2001 [4] with almost exactly the same wording for the name constraints extension as the 1997 edition. This lack of awareness is perhaps not that unusual, since RFC 2459 was only a profile of X.509, designed to give implementers recommendations on which options of X.509 to implement and which not to. It was not meant to be redefining the logic of X.509, and certainly not reversing it, although it might serve to further explain the intended logic to implementers. Consequently the two critical sentences of RFC 2459 were not added to the X.509 standard. Whilst many companies had implemented the X.509 semantics, including Entrust, some companies had implemented the RFC 2459 reversed semantics. In essence, the market place was in chaos. An attempt at reconciliation was attempted in late 2001 by the X.509 editor. This entailed a change of syntax and semantics to the X.509 standard, so that it could capture both the “trust all except” (black list) and “untrust all except” (white list) semantics. The expectation (at least in some quarters) was that the proposed update of RFC 2459 would adopt the new X.509 syntax and semantics. The change to X.509 was published in October 2001 as a technical corrigendum [8]. This is shown in Annex 3. The update to RFC 2459 was published in April 2002 as RFC 3280 [5]. Perhaps surprisingly, RFC 3280 contained exactly the same text as RFC 2459 and made no attempt at profiling the revised version of X.509 which had attempted to resolve the conflict.

The important things to note about the revised X.509 (2001) version are,

- a new object identifier was allocated to the revised extension, so that the

original name constraints extension was no longer part of the X.509 standard,

- in an attempt to align with the reversed RFC semantics, the original syntax had the new “trust all except” semantics applied to it, whilst the new syntax had the original “untrust all except” semantics applied to it,
- the new syntax added a “required name forms” field, with the semantics that each subsequent certificate in the chain “*must include a subject name of at least one of the required name forms*”. Thus disjunctive logic was used to resolve the many possibilities for mismatch between name forms in the certificate and name forms in the name constraints.
- it still does not cater for Scenario 3 since there is no way of skipping one or more certificates in a certificate chain before the names constraints takes effect.

In summary, the various editions of X.509 and their RFC profiles have remained out of synchronisation over name constraints for all of their lifetimes, with the latest version of X.509 (the 2001 corrigendum) and RFC 3280 being out of synchronisation for the last 4 years. The situation has recently been brought to the attention of the X.509 standards community again, through the issuing of defect report 314 by the RFC 3280bis design team. This recommends that X.509 reverts to the original 1997 and 2001 syntax but keeps the new “trust all except” (black list) semantics instead of its original “untrust all except” (white list) semantics, and, in addition, X.509 should define a new certificate extension that will capture the original “untrust all except” (white list) semantics.

4. Conclusions and Way Forward

This is clearly a sorry tale of continually changing syntaxes and semantics, misunderstandings between two standards creating bodies, the IETF and ISO/ITU-T, a lack of communication and perhaps even lethargy at dealing with issues in a timely manner. The obvious question to ask now is “where do we go from here”. Clearly there are several possibilities. This paper lists some of them, primarily from a technical perspective without considering the commercial or political implications of any one of them. The other considerations that will also need to be taken into account when coming to a resolution of the problem, are trust and usability, and how relying parties should behave or adapt when they are presented with either of the trust paradigms “trust all except” and “untrust all except”. Different user communities may prefer different trust paradigms.

Some of the different technical possibilities envisaged by the author are:

1. The ITU-T/ISO X.509 group could accede to the RFC 3280bis design team’s request, and revert the X.509 name constraints syntax to that of 1997 and 2001, whilst keeping the new “trust all except” (black list) semantics. A new extension would then need to be defined that encapsulated the original “untrust all except” (white list) semantics, along with the original exclusion control mechanisms from the 94/95 drafts i.e. of specifying policy sets and certificate path skipping that control which sets of certificates the constraint applies to. In this case the IETF would need to do nothing to its profile. Implementers who conform to the IETF semantics would not need to do anything unless and until the

new “white list” extension is defined and they decide to add it to their implementations. The new extension is important to cater for Scenario 3.

2. The ITU-T/ISO X.509 group could revert to the 1997 and 2001 syntax and original “untrust all except” (white list) semantics and add additional clarifying text to make clear that a disjunctive logic is used to resolve name form conflicts between the subject names and name constraints. This would be in the spirit of the original extension, although it would fail to cater for Scenario 3 or those implementations that support the IETF semantics. In this case the IETF would need to take this change of semantics into account when revising RFC 3280, which would mean deleting the two critical sentences that they added in RFC 2459. ISO/ITU-T should then consider enhancing the extension, or creating a new one, so that it can cater for Scenario 3, which is an important use case to consider.

3. A more dramatic solution might be to add an optional parameter (e.g. integer) to the 1997 syntax with the semantics “don’t check (n) CA certificates”, in order to cater for Scenario 3. This would be similar to the skipCerts integer that was present in the 94/95 draft standard. Part of the rationale given to the author for the current RFC semantics, is so that end entities and CAs could have different name forms, and then only the end entity name forms would be constrained by the name constraints. In other words, to achieve Scenario 3 by using different name forms for CAs and end users. The addition of a specific parameter which indicates that this is what is required, is semantically better than the current method of reversing the trust semantics to

“trust all except” which is too loose in its control capability, and open to abuse.

4. Either of the standards bodies could create a completely new certificate extension with a more sophisticated ASN.1 data type that could precisely specify which names are to be trusted and which are not, and when in the certificate chain the constraint should come into effect. For example, the extension could contain a sequence of permitted, excluded, and required name forms and their name spaces, along with a “Skip N Certificates” parameter. This is the clean sheet approach of taking the requirements and starting from scratch. I am not sure how successful this would be, given the current large installed user base.

5. Finally, the resolution could simply be to do nothing to the latest X.509 syntax and semantics, since this allows both “trust all except” and “untrust all except” semantics to be specified. The IETF PKIX group can then decide to either profile the original X.509 syntax, as they currently do, and keep their existing syntax and semantics, or migrate to profiling the latest version of X.509. Since the IETF has been out of synchronisation with the X.509 name constraints extension ever since their first RFC was published in 1999, being out of synchronisation for another few years should not pose any significant problems to them or to implementers. However, their current approach to solving Scenario 3 type use cases is less than optimal.

In summary, what lessons have we learnt from this development? Clearly writing IT standards is hard, and perhaps writing security standards is even harder. Even though the editors try hard to remove ambiguities and incomplete specifications

from standards, nevertheless they still exist. Standards have bugs in them just like software, and just like software, you don't know what bugs are there until someone finds them. Cross fertilisation of experts between base standards writers and profile writers will clearly help identify poor specifications, but this is not always practical given the constituencies of the two communities. Finally, given that we are human, errors will always occur. The real test of human ingenuity and adaptability is not that we never generate errors, but rather that we can resolve them effectively when they do occur. Sadly in this case we appear to have failed the test so far.

References

- [1] S.Kent. "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. RFC 1422. February 1993.
- [2] ISO 9594-8/ITU-T Rec. X.509 (1997) The Directory: Authentication framework
- [3] R. Housley, W. Ford, W. Polk, D. Solo. "Internet X.509 Public Key Infrastructure Certificate and CRL Profile". RFC 2459. January 1999.
- [4] ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Public-key and attribute certificate frameworks
- [5] R. Housley, W. Polk, W. Ford, D. Solo "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". RFC 3280, April 2002
- [6] ISO/IEC JTC 1/SC 21 N9214 "Proposed Draft Amendments PDAM 4 to ISO/IEC 9594-2, PDAM 2 to ISO/IEC 9594-6, PDAM 1 to ISO/IEC 9594-7, and PDAM 1 to ISO/IEC 9594-8", Orlando, USA, Dec 1994
- [7] ISO/IEC JTC 1/SC 21/WG 4 and ITU-T Q15/7 Collaborative Editing

Meeting on the Directory "Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 9594-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions", Ottawa, Canada, July 1995

[8] ITU-T. "Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks Technical Corrigendum 1". Oct 2001.

[9] Private communications with Hoyt Kesterson (X.500 rapporteur), Warwick Ford (national delegate), and Steve Kent (PKIX chair).

Acknowledgements

The author would like to thank Hoyt Kesterson for providing the historical ISO/ITU-T documents on which this paper is based.

Annex 1. The original PDAM Definition of Name Constraints

12.5.2.2 Name constraints field

This field specifies a set of constraints with respect to the names for which subsequent CAs in a certification path may issue certificates. The following ASN.1 type defines this field:

```
nameConstraints EXTENSION ::= {
  SYNTAX          NameConstraintsSyntax
  IDENTIFIED BY { id-ce 11 } }

NameConstraintsSyntax ::= SEQUENCE OF SEQUENCE {
  policySet      [0] CertPolicySet OPTIONAL,
  -- If policySet is omitted, the constraints
  -- apply to all policies for which the
  -- certificate is applicable
  nameSpaceConstraint [1] NameSpaceConstraint OPTIONAL,
  nameSubordConstraint [2] NameSubordConstraint OPTIONAL }

NameSpaceConstraint ::= SEQUENCE OF SubtreeSpecification
  (CONSTRAINED BY { -- specificationFilter is not permitted -- })

NameSubordConstraint ::= SEQUENCE {
  subordType ENUMERATED {
    subordinateToCA (0),
    subordinateToCAsSuperior (1) }
    DEFAULT subordinateToCAsSuperior,
  skipCerts INTEGER DEFAULT 0 }
```

This extension is always critical. The fields are interpreted as follows:

- policySet: **This indicates those certificate policies to which the constraints apply. If this component is omitted, the constraints apply regardless of policy.**
- **nameSpaceConstraint**: If this constraint is present, a certificate issued by the subject CA of this certificate should only be considered valid if for a subject within one of the specified subtrees. Any subtree class specification may contain a chop specification; if there is no chop specification, a subtree is considered to extend to the leaves of the DIT.
- **nameSubordConstraint**: This constraint is associated with a nominated CA in the certification path, being either the subject CA of this certificate or a CA which is the subject of a subsequent certificate in the certification path. If the value **subordinateToCA** is specified then, in all certificates in the certification path starting from a certificate issued by the nominated CA, the subject name must be subordinate to the issuer name of the same certificate. If the value **subordinateToCAsSuperior** is specified then, in all certificates in the certification path starting from a certificate issued by the nominated CA, the subject name must be subordinate to the name of the immediately superior DIT node of the issuer of the same certificate. The value of **skipCerts** indicates the number of certificates in the certification path to skip before the name subordination constraint takes effect; if value 0, the constraint starts to apply with certificates issued by the subject CA of this certificate.

Notes

1 The name constraint capability provided through the **subtreesConstraint** field in the basic constraints extension may be adequate for many applications. The name constraints extension is an alternative which offers a

more powerful range of constraining options, including the ability to fully reflect Internet Privacy Enhanced Mail [RFC 1422] rules.

2. The **subordinateToCA** alternative is provided only for compatibility with the Internet Privacy Enhanced Mail [RFC 1422] conventions. The **subordinateToCAsSuperior** rule is more powerful and its use is recommended in new infrastructures.

Imported from X.501(93)

```
SubtreeSpecification ::= SEQUENCE {
    base [0] LocalName DEFAULT { },
    specificationFilter [4] COMPONENTS OF ChopSpecification,
    -- empty sequence specifies whole administrative area
    Refinement OPTIONAL }

ChopSpecification ::= SEQUENCE {
    specificExclusions [1] SET OF CHOICE {
        chopBefore [0] LocalName,
        chopAfter [1] LocalName } OPTIONAL,
    minimum [2] BaseDistance DEFAULT 0,
    maximum [3] BaseDistance OPTIONAL }
```

Annex 2. The X.509 (1997) Standard Definition of Name Constraints

12.4.2.2 Name constraints field

This field, which shall be used only in a CA-certificate, indicates a name space within which all subject names in subsequent certificates in a certification path must be located. This field is defined as follows:

```
nameConstraints EXTENSION ::= {
    SYNTAX NameConstraintsSyntax
    IDENTIFIED BY id-ce-nameConstraints }

NameConstraintsSyntax ::= SEQUENCE {
    permittedSubtrees [0] GeneralSubtrees OPTIONAL,
    excludedSubtrees [1] GeneralSubtrees OPTIONAL }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
    base GeneralName,
    minimum [0] BaseDistance DEFAULT 0,
    maximum [1] BaseDistance OPTIONAL }
```

BaseDistance ::= INTEGER (0..MAX)

If present, the **permittedSubtrees** and **excludedSubtrees** components each specify one or more naming subtrees, each defined by the name of the root of the subtree and, optionally, within that subtree, an area that is bounded by upper and/or lower levels. If **permittedSubtrees** is present, of all the certificates issued by the subject CA and subsequent CAs in the certification path, only those certificates with subject names within these subtrees are acceptable. If **excludedSubtrees** is present, any certificate issued by the subject

CA or subsequent CAs in the certification path that has a subject name within these subtrees is unacceptable. If both **permittedSubtrees** and **excludedSubtrees** are present and the name spaces overlap, the exclusion statement takes precedence.

Of the name forms available through the **GeneralName** type, only those name forms that have a well-defined hierarchical structure may be used in these fields. The **directoryName** name form satisfies this requirement; when using this name form a naming subtree corresponds to a DIT subtree. Conformant implementations are not required to recognize all possible name forms. If the extension is flagged critical and a certificate-using implementation does not recognize a name form used in any **base** component, the certificate shall be handled as if an unrecognized critical extension had been encountered. If the extension is flagged non-critical and a certificate-using implementation does not recognize a name form used in any **base** component, then that subtree specification may be ignored. When a certificate subject has multiple names of the same name form (including, in the case of the **directoryName** name form, the name in the subject field of the certificate if non-null) then all such names shall be tested for consistency with a name constraint of that name form.

NOTE — When testing certificate subject names for consistency with a name constraint, names in non-critical subject alternative name extensions should be processed, not ignored.

The **minimum** field specifies the upper bound of the area within the subtree. All names whose final name component is above the level specified are not contained within the area. A value of **minimum** equal to zero (the default) corresponds to the base, i.e. the top node of the subtree. For example, if **minimum** is set to one, then the naming subtree excludes the base node but includes subordinate nodes.

The **maximum** field specifies the lower bound of the area within the subtree. All names whose last component is below the level specified are not contained within the area. A value of **maximum** of zero corresponds to the base, i.e. the top of the subtree. An absent **maximum** component indicates that no lower limit should be imposed on the area within the subtree. For example, if **maximum** is set to one, then the naming subtree excludes all nodes except the subtree base and its immediate subordinates.

This extension may, at the option of the certificate issuer, be either critical or non-critical. It is recommended that it be flagged critical, otherwise a certificate user may not check that subsequent certificates in a certification path are located in the name space intended by the issuing CA.

If this extension is present and is flagged critical then a certificate-using system shall check that the certification path being processed is consistent with the value in this extension.

From Section 12.3.2.1

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {

otherName	[0]	INSTANCE OF OTHER-NAME,
rfc822Name	[1]	IA5String,
dNSName	[2]	IA5String,
x400Address	[3]	ORAddress,
directoryName	[4]	Name,
ediPartyName	[5]	EDIPartyName,
uniformResourceIdentifier	[6]	IA5String,
iPAddress	[7]	OCTET STRING,
registeredID	[8]	OBJECT IDENTIFIER }

OTHER-NAME ::= TYPE-IDENTIFIER

Annex 3. The 2001 Corrigendum Definition of Name Constraints

8.4.2.2 Name constraints extension

This field, which shall be used only in a CA-certificate, indicates a name space within which all subject names in subsequent certificates in a certification path must be located. This field is defined as follows:

nameConstraints EXTENSION ::= {
 SYNTAX **NameConstraintsSyntax**
 IDENTIFIED BY **id-ce-nameConstraint }**

NameConstraintsSyntax ::= SEQUENCE {
 permittedSubtrees **[0] GeneralSubtrees OPTIONAL,**
 excludedSubtrees **[1] GeneralSubtrees OPTIONAL,**
 requiredNameForms **[2] NameForms OPTIONAL }**

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
 base **GeneralName,**
 minimum **[0] BaseDistance DEFAULT 0,**
 maximum **[1] BaseDistance OPTIONAL }**

BaseDistance ::= INTEGER (0..MAX)

NameForms ::= SEQUENCE {
 basicNameForms **[0] BasicNameForms OPTIONAL,**
 otherNameForms **[1] SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER OPTIONAL }**
(ALL EXCEPT ({ -- none; i.e.: at least one component shall be present --}))

BasicNameForms ::= BIT STRING {
 rfc822Name **(0),**
 dnsName **(1),**
 x400Address **(2),**
 directoryName **(3),**
 ediPartyName **(4),**
 uniformResourceIdentifier **(5),**
 iPAddress **(6),**
 registeredID **(7) } (SIZE (1..MAX))**

If present, the **permittedSubtrees** and **excludedSubtrees** components each specify one or more naming subtrees, each defined by the name of the root of the subtree and optionally, within that subtree, an area that is bounded by upper and/or lower levels. If **permittedSubtrees** is present, subject names within these subtrees are acceptable. If **excludedSubtrees** is present, any certificate issued by the subject CA or subsequent CAs in the certification path that has a subject name within these subtrees is unacceptable. If both **permittedSubtrees** and **excludedSubtrees** are present and the name spaces overlap, the exclusion statement takes precedence for names within that overlap. If neither permitted nor excluded subtrees are specified for a name form, then any name within that name

form is acceptable. If **requiredNameForms** is present, all subsequent certificates in the certification path must include a name of at least one of the required name forms.

If **permittedSubtrees** is present, the following applies to all subsequent certificates in the path. If any certificate contains a subject name (in the **subject** field or **subjectAltNames** extension) of a name form for which permitted subtrees are specified, the name must fall within at least one of the specified subtrees. If any certificate contains only subject names of name forms other than those for which permitted subtrees are specified, the subject names are not required to fall within any of the specified subtrees. For example, assume that two permitted subtrees are specified, one for the DN name form and one for the rfc822 name form, no excluded subtrees are specified, but **requiredNameForms** is specified with the **directoryName** bit and **rfc822Name** bit present. A certificate that contained only names other than a directory name or rfc822 name would be unacceptable. If **requiredNameForms** were not specified, however, such a certificate would be acceptable. For example, assume that two permitted subtrees are specified, one for the DN name form and one for the rfc822 name form, no excluded subtrees are specified, and **requiredNameForms** is not present. A certificate that only contained a DN and where the DN is within the specified permitted subtree would be acceptable. A certificate that contained both a DN and an rfc822 name and where only one of them is within its specified permitted subtree would be unacceptable. A certificate that contained only names other than a DN or rfc822 name would also be acceptable.

If **excludedSubtrees** is present, any certificate issued by the subject CA or subsequent CAs in the certification path that has a subject name (in the **subject** field or **subjectAltNames** extension) within these subtrees is unacceptable. For example, assume that two excluded subtrees are specified, one for the DN name form and one for the rfc822 name form. A certificate that only contained a DN and where the DN is within the specified excluded subtree would be unacceptable. A certificate that contained both a DN and an rfc822 name and where at least one of them is within its specified excluded subtree would be unacceptable.

When a certificate subject has multiple names of the same name form (including, in the case of the **directoryName** name form, the name in the subject field of the certificate if non-null), then all such names shall be tested for consistency with a name constraint of that name form.

If **requiredNameForms** is present, all subsequent certificates in the certification path must include a subject name of at least one of the required name forms.

Of the name forms available through the **GeneralName** type, only those name forms that have a well-defined hierarchical structure may be used in the **permittedSubtrees** and **excludedSubtrees** fields. The **directoryName** name form satisfies this requirement; when using this name form a naming subtree corresponds to a DIT subtree.

The **minimum** field specifies the upper bound of the area within the subtree. All names whose final name component is above the level specified are not contained within the area. A value of **minimum** equal to zero (the default) corresponds to the base, i.e. the top node of the subtree. For example, if **minimum** is set to one, then the naming subtree excludes the base node but includes subordinate nodes.

The **maximum** field specifies the lower bound of the area within the subtree. All names whose last component is below the level specified are not contained within the area. A value of **maximum** of zero corresponds to the base, i.e. the top of the subtree. An absent **maximum** component indicates that no lower limit should be imposed on the area within the subtree. For example, if **maximum** is set to one, then the naming subtree excludes all nodes except the subtree base and its immediate subordinates.

This extension may, at the option of the certificate issuer, be either critical or non-critical. It is recommended that it be flagged critical, otherwise a certificate user may not check that subsequent certificates in a certification path are located in the name space intended by the issuing CA.

Conformant implementations are not required to recognize all possible name forms.

If the extension is present and is flagged critical, a certificate-using implementation must recognize and process all name forms for which there is both a subtree specification (permitted or excluded) in the extension and a corresponding value in the **subject** field or **subjectAltNames** extension of any subsequent certificate in the certification path. If an unrecognized name form appears in both a subtree specification and a subsequent certificate, that certificate shall be handled as if an unrecognized critical extension was encountered. If any subject name in the certificate falls within an excluded subtree, the certificate is unacceptable. If a subtree is specified for a name form that is not contained in any subsequent certificate, that subtree can be ignored. If the **requiredNameForms** component specifies only unrecognized name forms, that certificate shall be handled as if an unrecognized critical extension was encountered. Otherwise, at least one of the recognized name forms must appear in all subsequent certificates in the path.

If the extension is present and is flagged non-critical and a certificate-using implementation does not recognize a name form used in any **base** component, then that subtree specification may be ignored. If the extension is flagged non-critical and any of the name forms specified in the **requiredNameForms** component are not recognized by the certificate-using implementation, then the certificate shall be treated as if the **requiredNameForms** component was absent.