The Machine-to-Machine (M2M) Public Key Certificate Format

draft-ford-m2mcertificate-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 23, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

The X.509 public key certificate format is overly verbose for Internet-of-Things (IoT) constrained environments, where nodes with limited memory and networks with limited bandwidth are not uncommon. The Machine-to-Machine (M2M) certificate format is a pruned down and encoding-optimized replacement for X.509, which reuses much of the X.509 semantics but reduces certificate sizes by typically 40%. We are proposing that IETF recognize the M2M format as an optional replacement for X.509 in Internet applications including, but not limited to, TLS and DTLS.

Table of Contents

1. Introduction ................................................... 2
2. Restrictions Applied to the X.509 Model ......................... 3
3. Other Optimizations ............................................ 4
4. Certificate Size Estimates ..................................... 4
5. Certificate Format ............................................ 5
6. Rules for Omitting Algorithm Fields ............................ 8
   6.1. Algorithm Fields in CA Certificates ................. 9
   6.2. Algorithm Fields in End Entity Certificates ........ 9
7. Security Considerations ........................................ 9
8. IANA Considerations ........................................... 9
9. Conclusions ................................................... 9
10. References .................................................. 10
   10.1. Normative References ..................................... 10
   10.2. Informative References ................................ 10
11. Acknowledgments ................................................ 10
Appendix A. Certificate Field Size Breakdown Tables ............ 11
   A.1. EE Small Case ............................................ 11
   A.2. EE Medium Case ......................................... 12
   A.3. CA Certificate Case ..................................... 13

1. Introduction

The predominant public key certificate format has for many years been the X.509 format [RFC5280]. X.509 was designed to be extremely flexible and open-ended, in an environment of RSA and DSA signature technologies. X.509 is not, however, a good certificate format for Internet-of-Things constrained environments, where nodes with limited memory and networks with limited bandwidth are not uncommon. With RSA and DSA technologies, overheads in the certificate format were comparatively inconsequential because the large key and signature fields were the dominant certificate components size-wise. However, with the much more efficient ECC technology used today, the certificate format overheads become a very important factor in making
certificates efficient enough for low bandwidth constrained applications. In essence, the X.509 certificate format is too verbose for these applications.

There is a clear need for a more efficient certificate format than X.509. Because many fields are inherently variable-length, it still makes sense to use ASN.1 notation and encoding for this new format, allowing us to borrow heavily from the X.509 model and to reuse much of the X.509 semantics and some software.

The Machine-to-Machine (M2M) certificate format was designed to satisfy the above objectives. What was done was to strip down the X.509 format to eliminate features that are not needed today, while optimizing the encoding. The result is a certificate format that typically reduces certificate size by about 40% compared with X.509.

The M2M format supports various digital signature technologies including ECDSA, RSA, and Elliptic Curve Qu-Vanstone (ECQV) [SEC4]. No particular technology is required by this specification and we use ECDSA as the baseline for comparative certificate size calculations.

The M2M certificate format has been adopted by the NFC Forum for Near Field Communications signatures, and published by that organization [NFC-SIG]. However it is a general purpose design which is equally applicable to Internet-of-Things applications.

We are proposing that IETF recognize the M2M format as an optional replacement for X.509 in Internet applications including, but not limited to, TLS and DTLS. A companion Internet-Draft addresses the use of the M2M format with TLS and DTLS [TLS-M2M].

2. Restrictions Applied to the X.509 Model

The M2M certificate model restricts the X.509 model as follows:

- Directory Name (DN) names are limited to using the RFC 5280 mandatory attributes plus other attributes in common use, namely: country, organization, organizational unit, distinguished name qualifier, state or province name, common name, serial number, locality, domain component. There can be only one of each, no more than 4 total, and no multi-level names. M2M has also added an Object Identifier (OID) option (may prove a useful identifier for CAs) and an OCTET STRING option (may prove an efficient option for device identifiers).

- DN character encodings are limited to one string type, usually UTF8String (which a profile might limit to IAS characters).
o Modest length constraints are defined for all DN attributes.

o Criticality flags for extensions are eliminated (criticality may be implied by semantics).

o The following built-in extensions are defined for end-entity certificates: issuer key id, subject key id, key usage (first 7 bits only), certificate policies (one OID, no qualifiers), subject alternative name, issuer alternative name, extended key usage (one OID), authority information access (URI for OCSP responder only).

o The following additional built-in extension is defined for CA-certificates: basic constraints

o While it is not anticipated that applications will require extensions other than the built-in extensions noted above, there is a catch-all to optionally permit any standard X.509 extension from RFC 5280 to be included.

3. Other Optimizations

An optional optimization known as "parameter inheritance" is also provided. For applications where a certificate is always accompanied in its transmission by its superior certificate, we can eliminate the issuer field from the transmitted form of the certificate (it is still included for signature generation and verification purposes). Issuer name can be inherited from the subject field of the superior certificate. Therefore, this field is made optional in the syntax in order to allow two variants of every certificate: the full (to-be-signed) form and the transmitted form.

In addition the M2M format adopts from SEC4 MES the use of Unix time (rather than ASN.1 time types) to represent validity period. It also drops the redundant algorithm identifier from the certificate outer structure and makes sundry miscellaneous optimizations.

4. Certificate Size Estimates

Below are exemplary certificate size estimates for each of the formats M2M and X.509 (we include SEC4 MES for the one case where it applies). All sizes are for 224-bit ECC. We have used the following example cases (note EE=end-entity certificate):

o EE Small case: ECDSA minimal data, one-component 8-character names and 7-bit key usage extension.
o EE Medium case: ECDSA more general example, with two-component 16-character names and these extensions: 7-bit key usage, certificate policy OID, 20-character OCSP URL, 10-character subject alternate name.

o CA Certificate case: A certificate for an ECDSA Sub-CA ECDSA-signed by a root CA. Two-component 16-character names and extensions: 7-bit key usage, basic constraints, 20-character OCSP URL.

<table>
<thead>
<tr>
<th></th>
<th>M2M with parameter inheritance</th>
<th>M2M</th>
<th>X.509</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE Small</td>
<td>136</td>
<td>155</td>
<td>241</td>
</tr>
<tr>
<td>EE Medium</td>
<td>189</td>
<td>218</td>
<td>364</td>
</tr>
<tr>
<td>CA Cert</td>
<td>N/A</td>
<td>207</td>
<td>338</td>
</tr>
</tbody>
</table>

Figure 1 Certificate Sizes in Bytes

In summary, a standalone M2M EE certificate is roughly 140-to-220 bytes (compare 240-to-360 for X.509) and a two-certificate ECDSA chain is roughly 340-to-420 bytes (compare 570-to-690 for X.509).

We assume Algorithm OIDs of the form 1.3.11111.1.1 (5 octets value field) with no algorithm parameters and certificate serial numbers of 8 octets.

For more detail of the calculations leading to the above size comparisons see Appendix A.

5. Certificate Format

The M2M certificate format is defined using Abstract Syntax Notation One (ASN.1) [X.680].

```asciidoc
-- Machine-to-Machine certificate format
M2M-Certificate-Definition
   {1 3 186 asn1-modules (5) m2m-certificate (0)}
-- Structure MUST be DER encoded
DEFINITIONS AUTOMATIC TAGS ::= BEGIN
```

Ford & Poeluev  Expires September 23, 2015  [Page 5]
Certificate ::= [APPLICATION 20] IMPLICIT SEQUENCE {
  tbsCertificate TBSCertificate, -- To be signed certificate
  cACalcValue OCTET STRING -- Contains signature for a signed
    -- certificate or public key derivation value
    -- for an ECQV certificate
}

-- The APPLICATION 20 tag is intended to make the M2M format
-- apparent by inspecting the first byte of the encoding

TBSCertificate ::= SEQUENCE {
  version INTEGER {v1(0) } DEFAULT v1,
  serialNumber OCTET STRING (SIZE (1..20)),
  cAAlgorithm OBJECT IDENTIFIER OPTIONAL, -- Identifies CA
    -- algorithm, hash function & optionally
    -- other required parameters (e.g., for ECC the
    -- curve).
    -- Required for signature verification but may
    -- be omitted from the transmitted cert and
    -- filled in from the pKAlgorithm of the
    -- superior cert (provided not root cert)
    -- prior to signature verification
  cAAlgParams OCTET STRING OPTIONAL,
    -- Identifies CA algorithm parameters.
    -- This specification does not provide for
    -- omitting this field in transmission and
    -- subsequently replacing it from the superior
    -- certificate for signature verification
  issuer Name OPTIONAL, -- Required for signature
    -- verification but may be omitted from the
    -- transmitted cert and filled in from the
    -- subject field of the superior cert (provided
    -- not root cert) prior to signature verification
  validFrom OCTET STRING (SIZE (4..5)) OPTIONAL,
    -- Unix time. If omitted no validity specified
  validDuration OCTET STRING (SIZE (1..4)) OPTIONAL,
    -- # of seconds. If omitted no expiry specified
  subject Name,
  pKAlgorithm OBJECT IDENTIFIER OPTIONAL,
    -- Default is same as cAAlgorithm in this
    -- certificate
}
pKAlgParams OCTET STRING OPTIONAL,
pubKey OCTET STRING OPTIONAL, -- Omit for an ECQV cert
authKeyId AuthKeyId OPTIONAL,
subjKeyId OCTET STRING OPTIONAL,
keyUsage OCTET STRING (SIZE (1)) OPTIONAL, -- Critical
   -- One byte containing a bit string, as described below.
basicConstraints INTEGER (0..7) OPTIONAL, -- If absent this
   -- is an end-entity cert; max intermed path length for CA cert
certificatePolicy OBJECT IDENTIFIER OPTIONAL,
subjectAltName GeneralName OPTIONAL,
issuerAltName GeneralName OPTIONAL,
extendedKeyUsage OBJECT IDENTIFIER OPTIONAL,
authInfoAccessOCSP IA5String OPTIONAL, -- OCSP responder URI
cRLDistribPointURI IA5String OPTIONAL,
   -- CRL distribution point URI
x509extensions X509Extensions OPTIONAL,
...

Name ::= SEQUENCE SIZE (1..4) OF AttributeValue

AttributeValue ::= CHOICE {
country           PrintableString (SIZE (2)),
organization      UTF8String (SIZE (1..32)),
organizationalUnit UTF8String (SIZE (1..32)),
distinguishedNameQualifier PrintableString (SIZE (1..32)),
stateOrProvince   UTF8String (SIZE (1..4)),
locality          UTF8String (SIZE (1..32)),
commonName        UTF8String (SIZE (1..32)),
serialNumber      PrintableString (SIZE (1..32)),
domainComponent   IA5String (SIZE (1..32)),
registeredId       OBJECT IDENTIFIER,
octetsName        OCTET STRING (SIZE (1..8))
}

AuthKeyId ::= SEQUENCE {
   keyIdentifier OCTET STRING OPTIONAL,
   authCertIssuer GeneralName OPTIONAL,
   authCertSerialNum OCTET STRING (SIZE(1..20)) OPTIONAL
}

X509Extensions ::= SEQUENCE OF Extension

Extension ::= SEQUENCE {

extnID OBJECT IDENTIFIER,
criticality BOOLEAN DEFAULT FALSE,
extnValue OCTET STRING
}

GeneralName ::= CHOICE {
  rfc822Name IA5String (SIZE (1..128)),
  dNSName IA5String (SIZE (1..128)),
  directoryName Name,
  uniformResourceIdentifier IA5String (SIZE (1..128)),
  iPAddress OCTET STRING (SIZE (1..16)),
  --4 octets for IPV4, 16 octets for IPV6
  registeredID OBJECT IDENTIFIER
}

-- Notes:
-- * The times are represented using Unix time, i.e. # of seconds
-- since the Unix epoch: http://en.wikipedia.org/wiki/Unix_time
-- The validFrom field permits 40-bit values to avoid problems in
-- 2038 (when 32-bit values won’t be enough).
--
-- The keyUsage field conveys a single octet equal to the
-- second octet of the DER encoding of the following BIT STRING

  KeyUsage ::= BIT STRING {
    digitalSignature (0),
    nonRepudiation (1),
    keyEncipherment (2),
    dataEncipherment (3),
    keyAgreement (4),
    keyCertSign (5),
    -- Use keyCertSign also for an ECQV certificate issuer
    cRLSign (6)
  -- the last bit in the byte is always zero (7)
END

6. Rules for Omitting Algorithm Fields

Following are the rules defining when and how omitting algorithm
fields is allowed.
6.1. Algorithm Fields in CA Certificates

cAAlgorithm: Omitting is only allowed when pKAlgorithm (or cAAlgorithm if pKAlgorithm is omitted in a superior certificate) of a superior certificate fully specifies the signature algorithm and its parameters (i.e. signature and hash algorithms plus any required parameters, e.g. in the case of ECC, the curve).

pKAlgorithm: If omitted in a CA certificate, cAAlgorithm specifies the signature and hash algorithms and any required parameters (e.g. curve) for pubKey Algorithm fields in End Entity Certificates.

6.2. Algorithm Fields in End Entity Certificates

cAAlgorithm: Omitting is only allowed when pKAlgorithm (or cAAlgorithm if pKAlgorithm is omitted in a superior certificate) of a superior certificate fully specifies the signature algorithm and its parameters (i.e. signature and hash algorithms plus any required parameters, e.g. in the case of ECC, the curve).

pKAlgorithm: If omitted in an end entity certificate, cAAlgorithm specifies the required parameters (e.g. curve and optionally signature & hash algorithms) for pubKey.

7. Security Considerations

The M2M Certificate Format is believed by the authors to have the same security characteristics as the X.509 certificate format.

8. IANA Considerations

None known.

9. Conclusions

The IETF and applicable Working Groups are encouraged to adopt the M2M certificate format as an optional alternative to the X.509 format in all applications in the Internet-of-Things space. There are significant size and bandwidth savings and no significant loss of features of practical importance.
10. References

10.1. Normative References


10.2. Informative References


11. Acknowledgments

Recognition is due to Rob Lambert and Jeremy Rowley for their critical reviews of the specification, and to the development team of TrustPoint Innovation Technologies for proving the specification works in practical implementations.

This document was prepared using 2-Word-v2.0.template.dot.
# Appendix A.

**Certificate Field Size Breakdown Tables**

This Appendix lists the certificate field sizes used in arriving at the certificate sizes in Figure 1. Use this to check our numbers and also to see where M2M saves the bytes.

## A.1. EE Small Case

<table>
<thead>
<tr>
<th>Field</th>
<th>M2M with parameter inheritance</th>
<th>M2M</th>
<th>X.509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic envelope</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Outside algorithm id</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Outside algorithm params</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Signature/CA calc value</td>
<td>65</td>
<td>65</td>
<td>66</td>
</tr>
<tr>
<td>Version</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Serial number</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Inside algorithm id</td>
<td>0</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Algorithm parameters</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Issuer</td>
<td>0</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>Validity</td>
<td>11</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>Subject</td>
<td>12</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>Subject algorithm</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Subject parameters</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Subject public key</td>
<td>31</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>Extensions envelope</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Key usage 7-bit extn</td>
<td>3</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>Basic constraints extn</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cert policies extn</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OCSP URL extn 20-char</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Subject alt name 10-char</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>136</td>
<td>155</td>
<td>241</td>
</tr>
</tbody>
</table>

Figure 2 EE Small Case Field Sizes in Bytes
### A.2. EE Medium Case

<table>
<thead>
<tr>
<th>Field</th>
<th>M2M with parameter inheritance</th>
<th>M2M</th>
<th>X.509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic envelope</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Outside algorithm id</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Outside algorithm params</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Signature/CA calc value</td>
<td>65</td>
<td>65</td>
<td>66</td>
</tr>
<tr>
<td>Version</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Serial number</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Inside algorithm id</td>
<td>0</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Algorithm parameters</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Issuer</td>
<td>0</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>Validity</td>
<td>11</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>Subject</td>
<td>22</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>Subject algorithm</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Subject parameters</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Subject public key</td>
<td>31</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>Extensions envelope</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Key usage 7-bit extn</td>
<td>3</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>Basic constraints extn</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cert policies extn</td>
<td>7</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>OCSP URL extn 20-char</td>
<td>22</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>Subject alt name 10-char</td>
<td>14</td>
<td>14</td>
<td>23</td>
</tr>
<tr>
<td>TOTAL</td>
<td>189</td>
<td>218</td>
<td>364</td>
</tr>
</tbody>
</table>

Figure 3 EE Medium Case Field Sizes in Bytes
A.3. CA Certificate Case

<table>
<thead>
<tr>
<th></th>
<th>M2M with parameter inheritance</th>
<th>M2M</th>
<th>X.509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic envelope</td>
<td>N/A</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Outside algorithm id</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Outside algorithm params</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Signature/CA calc value</td>
<td>65</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Serial number</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Inside algorithm id</td>
<td>7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Algorithm parameters</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Issuer</td>
<td>22</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Validity</td>
<td>11</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>22</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Subject algorithm</td>
<td>7</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Subject parameters</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Subject public key</td>
<td>31</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Extensions envelope</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Key usage 7-bit extn</td>
<td>3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Basic constraints extn</td>
<td>3</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Cert policies extn</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OCSP URL extn 20-char</td>
<td>22</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Subject alt name 10-char</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>207</td>
<td>338</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4 CA Certificate Case Field Sizes in Bytes

Authors’ Addresses

Warwick Ford
TrustPoint Innovation Technologies, Ltd.
700 S Monarch St Unit 203,
Aspen, CO 81611

Email: wford@wyltan.com