Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 12, 2010

D. McGrew
M. Pritikin
Cisco Systems
May 11, 2010

The Compressed X.509 Certificate Format
draft-pritikin-comp-x509-00

Abstract

This note defines the Compressed X.509 Format (CXF), a compact format for X.509 certificates and CRLs, and a way to translate between the compact format and standard X.509 encoding. The translation consists of the lossless compression algorithm DEFLATE with a particular predefined dictionary. Protocol bindings for TLS and IKE are designated.

This format is useful in a constrained environment, where bandwidth and/or storage is low, and it preserves interoperability with standards-compliant X.509 certificates and the systems for certificate processing, issuance, and management.

Table of Contents

1. Introduction

X.509 certificates and Certificate Revocation Lists (CRLs) are commonly used in the industry in general and in IETF protocols in particular [RFC5280]. These certificates use the ASN.1 encoding, which provides great flexibility, but which is verbose. Certificates are often several kilobytes long, even though the cryptographic data that they carry is often no more than 512 bytes, and can be as low as 64 bytes.

There is considerable interest in the use of standard communication protocols on low-power, low-speed, and low-cost devices. For example, Smart Grid networks are anticipated to use 802.15.4, where the maximum packet size is 108 bytes [RFC4944] and data rates are as low as 20 kbit/sec. While it is highly desirable to use digital certificates on these devices, and to use common standards like X.509 and PKIX [RFC5280], it is also desirable to avoid verbose encodings that take up storage space and bandwidth. With CXF, both of these goals can be met.

In this note we define a compact format for X.509 certificates, CRLs, and structured data containing X.509 certificates, and we define methods to translate between the compact format and the standard X.509-based encodings. The DEFLATE algorithm [RFC1951] is used, with a predefined CXF-specific dictionary; translation to the compressed format uses the compression algorithm, and the translation back to standard X.509 uses the decompression algorithm. The predefined dictionary used in the compressor/decompressor has been chosen to contain substrings that are typical for X.509 and PKIX . The use of this dictionary makes DEFLATE efficient even when it is applied to a single X.509 certificate. The background to this approach is discussed in Section 6.

One important certificate profile, in which compactness is desirable, is defined by the IEEE 802.1AR Secure Device Identity standard [802.1AR]. This X.509 profile is used for manufacturer-installed certificates, and it aims to be compact by requiring only the certificate fields that are vital in that function.

1.1. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Compressed X.509 Format (CXF)

This section defines the Compressed X.509 Format (CXF). A CXF certificate corresponds to an X.509 certificate in Distinguished Encoding Rules (DER) form. A DER certificate is converted into a CXF certificate by applying the DEFLATE compression algorithm, using the CXF "preset dictionary" (in the terms of [RFC1951]) that is defined in the following subsection.

A CXF certificate is converted into a DER certificate by applying the DEFLATE decompression algorithm, using the CXF dictionary.

CXF can be applied to individual certificates, as described above, or to any sequence of X.509 certificates, such as a certificate chain, or any ASN.1 encoded data containing X.509 certificates. In this note we define the application of CXF to:

individual X.509 Signature Certificates,

a sequence of X.509 certificates that are concatenated together, with no delimiters or other encoding (as in the TLS certificate_list [RFC5246]),

PKCS #7 wrapped X.509 certificate(s) (as used in IKE [RFC4306])

a CertificateBundle (as defined in IKE [RFC4306])

In each of the above cases, translation to and from compressed format uses the DEFLATE algorithm with the CXF dictionary.

2.1. The CXF Dictionary

DEFLATE compression algorithm allows the use of a "preset dictionary" to make compression more efficacious for particular applications; it it especially benefits the compression of shorter inputs. When a preset dictionary is used in the compression algorithm, the dictionary is fed into the compressor, and no output is produced while the dictionary is processed, but the compressor state is updated and maintained; after that, the data input is compressed. Decompression with a preset dictionary works similarly; data that is compressed with a particular preset dictionary must be decompressed with the same preset dictionary, or the output of the decompressor will not match the input of the compressor.

CXF uses a particular dictionary that has been selected to provide good compression on X.509 formatted certificates and CRLs. The details of how the dictionary was constructed are presented in Appendix A.

2.2. Other Considerations

In many cases, X.509 certificates are much larger than the cryptographic data that they carry because they include an abundance of optional fields. A certificate issuer can significantly reduce the size of a certificate by omitting non-essential fields. This is a valuable strategy that SHOULD be used whenever CXF is used.

3. Usage in TLS

It is possible to use CXF certificates in place of X.509 certificates whenever all communicating parties are aware of the format.

One important use case is the TLS protocol [RFC5246]. The TLS "cert_type" extension [RFC5081] allows the negotiation of a certificate format. Section 9 defines the cert_type value that is used to indicate the use of CXF in TLS, which is called a CXFCertificate.

Recall that a TLS Certificate is defined as a sequence (chain) of X.509v3 certificates:

```
opaque ASN.1Cert<1..2^24-1>;

struct {
    ASN.1Cert certificate_list<0..2^24-1>;
} Certificate;
```

A CXFCertificate is a Certificate, as defined above, which has been compressed as described in Section 2.

4. Usage in IKE

Another important usage is IKE [RFC4306]. The IKE Certificate Payload can carry certificate data in one of several different formats, and it indicates the format via its Certificate Encoding field. These formats include X.509 Signature Certificate and PKCS #7 wrapped X.509 certificate(s). Section 9 defines the Certificate Encoding values is used to indicate the use of CXF with these formats in IKE.

5. Other uses

One potentially interesting usage for CXF is the distribution of Certificate Revocation Lists (CRLs) and Delta CRLs as indicated by

the CRL Distribution Point (CDP) extension recommended in [RFC5280], Section 4.2.1.13. In this section, we consider some possible applications of CXF to CRLs, but the complete definition of such mechanisms is outside the scope of this note.

CXF could be applied to the following transport mechanisms:

HTTP - in this case, the CDP URI refers to a single DER encoded CRL as specified in [RFC2585]. The use of the CXF format could be indicated by a new media type, e.g. "application/pkix-xcrl".

FTP - a new file extension, e.g. ".xrl", could indicate that the CRL is in the CXF format.

LDAP - The use of the CXF format could be indicated by a new ASN.1 attribute, e.g. compressedcertificateRevocationList.

These encodings could apply equally well to delta-CRLs which are distinguished after decompression by checking the "delta CRL indicator" CRL extension [RFC5280] .

6. Background: Compression and Certificates

The data inside of a certificate can be roughly classified as follows:

Cryptographic data, which is encoded in ASN.1 BIT STRING fields. This data is incompressible.

Object Identifiers (OIDs), encoded in ASN.1 OBJECT fields. This data is compressible, especially with a well-defined dictionary.

Character strings, encoded in ASN.1 PRINTABLESTRING, IA5STRING, and OCTET STRING fields. This data is compressible, and can benefit from a well-defined dictionary. A dictionary that is tailored to a particular certificate profile can take advantage of substrings that are typical to the profile, such as "http://".

Time and numbers, encoded as UTCTIME and INTEGER fields. This data is compressible, and can benefit from a well-defined dictionary.

Encoding overhead. This data is compressible, especially with a well-defined dictionary.

For example, the data classification for a particular RSA-1024 certificate issued by a commercial certificate provider is

```
Cryptographic data:    270 bytes
Encoding overhead:     164 bytes
OIDs:                   75 bytes
Character strings:     297 bytes
Numbers and time:       28 bytes
total length:          831 bytes
```

The compression ratio of a CXF certificate is the length of that certificate divided by the length of the corresponding X.509 DER certificate. The compression ratio for the above certificate can be no lower than 0.32, considering the incompressible nature of the cryptographic data.

6.1. Efficiency

Self-signed certificates contain a lot of redundant information because several fields will appear twice. The compression ratio on self-signed certificates will be lower (that is, better) than for typical certificates.

A certificate chain also contains redundancy, similar to that of a self-signed certificate. CXF SHOULD applied to an entire chain of certificates, as a single atomic unit, as opposed to applying CXF individually to each of the certificates in a chain. Following the recommendation to apply CXF to the entire chain results in a more compact output, because the compression algorithm can take advantage of the redundancy. The protocol bindings defined in this note follow that recommendation.

In a software implementation, the CXF dictionary, decompressor, and compressor take up storage space. Any use of CXF to save memory (as opposed to bandwidth) will need to balance the additional memory required by these functions against the memory savings gained by using CXF certificates instead of uncompressed X.509 certificates.

7. Rationale

The general approach of defining a compact format for certificates by compressing X.509 is appealing in several ways. It reduces the data-encapsulation overhead of ASN.1 certificates without requiring changes to any ASN.1 encoder or decoder, thus addressing a major deficiency with ASN.1 while preserving the utility of the many applications that use it. Another appeal of this approach is the fact that compression is a relatively expensive operation, and decompression is relatively inexpensive. This disparity in cost is well matched for how certificates are used; compression can occur at the time that a certificate is created, and certificates can be

stored and carried in compressed form. For many uses of certificates, it is sufficient to use decompression but not compression.

The DEFLATE algorithm is well-suited to compressing ASN.1 certificates, because it internally uses three compression modes: no compression, compressed with LZ77 then fixed Huffman codes, and compressed with LZ77 then dynamic Huffman codes. The DEFLATE compressor separates its input into logically distinct blocks, each block of which uses one of the three compression modes. The incompressible cryptographic data can use the "no compression" mode; this avoids the expansion that would otherwise occur when data with no redundancy is run through a compressor, and it avoids the pollution of the LZ77 and dynamic Huffman code dictionary with irrelevant data. The other compression methods are suitable for compressing ASN.1, especially when used with a predefined dictionary. OIDs and their type and length codes can appear in the dictionary, as well as other common type and length codes and typical character strings. Repeated fields in certificates, such as those in self-signed certificates, are efficiently compressed via LZ77. A DEFLATE encoder can use a "lazy matching" strategy, in which several different partitions of the input data into distinct blocks are tried, and the partition that leads to the most compact output is used.

The DEFLATE algorithm is commonly used in the ZLIB Compressed Data Format [RFC1950]; this fact allows the implementers of CXF to take advantage of existing implementations of ZLIB. However, the ZLIB format is not used because the extensibility that it provides is not needed, and because of its data overhead.

The DEFLATE algorithm is used by the Efficient XML Interchange (EXI) Format [W3C-EXI], which defines a compact representation for the Extensible Markup Language (XML). Thus, CXF is well suited for use in constrained environments that implement EXI, in which the DEFLATE algorithm will already be present.

A compressed format for ASN.1 has been defined: the Packed Encoding Rules (PER) [X.691]. CXF does not use PER, because its use of DEFLATE with a predefined dictionary provides good compression that specifically targets X.509 certificates as they are typically used, and because ZLIB and DEFLATE are commonly available and are useful in applications other than CXF. PER is not widely used, and it requires that the decoder have the complete abstract syntax of the data structure to be decoded, which would be awkward for X.509.

7.1. Open Questions

The CXF dictionary could be further refined in future versions of this document. It is an open question exactly what OIDs, and what PKCS7 specific strings, should be included. The dictionary is almost completely in valid ASN.1 DER format; we conjecture that the dictionary can be improved by deviating more from the DER format. (For example the current dictionary includes Subject Key Identifier and Authority Key Identifier strings.) This would enable the dictionary to contain the important typical substrings, without all of the associated ASN.1 data, thus enabling the dictionary to be smaller while producing the same compression ratios.

8. Security Considerations

There are no security considerations inherent to the certificate format defined in this note. This is because there is a one-to-one mapping between certificates in X.509 DER format and certificates in compressed format.

When an entity accepts a certificate from an untrusted source, the validity of the certificate is checked before any data from the certificate is accepted. This validation uses the digital signature in the certificate, and not the key in the certificate. If the compressed format is in use, and there was an error in the decompression algorithm, a mangled X.509 certificate would result, and the validation of the signature will fail.

9. IANA Considerations

The TLS Certificate Type (defined by the cert_type extension) used to indicate the use of CXF in TLS is *TBD1*.

The IKE Certificate Encoding value used to indicate the presence of a CXF-formatted X.509 Signature Certificate in an IKE Certificate Payload is *TBD2*.

The IKE Certificate Encoding value used to indicate the presence of a CXF-formatted PKCS #7 wrapped X.509 certificate(s) in an IKE Certificate Payload is *TBD3*.

The IKE Certificate Encoding registration procedures require expert review; the IPsec Maintenance and Extension (IPsec ME) Working Group is a suitable place to seek such review. The TLS Certificate Type registration procedure is by IETF Consensus. The TLS Working Group is a suitable place for the IESG to seek input on this prospective

assignment. This paragraph will not be needed after publication as an RFC, and it should be deleted by the RFC editor.

10. References

10.1. Normative References

[RFC1950] Deutsch, L. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

10.2. Informative References

[802.1AR] IEEE 802.1AR, "Secure Device Identity", IEEE SA-Standards Board LAN/MAN Standards Committee, 2009.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5081] Mavrogiannopoulos, N., "Using OpenPGP Keys for Transport Layer Security (TLS) Authentication", RFC 5081, November 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[W3C-EXI] World Wide Web Consortium (W3C), "Efficient XML Interchange (EXI) Format", Candidate Recommendation version 1.0, 2009.

[X.691] ITU-T Recommendation X.691, "Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)", SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS OSI networking and system aspects - Abstract Syntax Notation One (ASN.1), 2002.

Appendix A. The construction of the CXF dictionary

The CXF dictionary was constructed to include data that are typical to X.509 and PKIX usage. In this section, we outline the data that were included in the dictionary, which can be categorized as encoding overhead, OIDs, character strings, and time.

The OIDs that are included are

X509v3 Authority Key Identifier

X509v3 Basic Constraints

X509v3 Subject Key Identifier

emailAddress

commonName

countryName

stateOrProvinceName

organizationName

organizationalUnitName

rsaEncryption

sha1WithRSAEncryption

The character string that is included is "http://www.com" The time that is included is 999912312359592.

The CXF dictionary has been produced using a rudimentary IEEE 802.1AR certificate that has been stripped of signatures and key data. The dictionary is in binary form and can still be said to be DER encoded, although it is no longer composed of valid [RFC5280] certificates.

The CXF dictionary is defined as the following sequence of hexadecimal bytes; the elements are to be read from left to right and

top to bottom:

```
0x30, 0x82, 0x01, 0x39, 0x30, 0x82, 0x01, 0x23,
0xa0, 0x03, 0x02, 0x01, 0x02, 0x02, 0x01, 0x01,
0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86, 0x48, 0x86,
0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00, 0x30,
0x19, 0x31, 0x17, 0x30, 0x15, 0x06, 0x03, 0x55,
0x04, 0x03, 0x13, 0x0e, 0x68, 0x74, 0x74, 0x70,
0x3a, 0x2f, 0x2f, 0x77, 0x77, 0x77, 0x2e, 0x63,
0x6f, 0x6d, 0x30, 0x1e, 0x17, 0x0d, 0x31, 0x30,
0x30, 0x35, 0x31, 0x31, 0x31, 0x39, 0x31, 0x33,
0x30, 0x33, 0x5a, 0x17, 0x0d, 0x31, 0x31, 0x30,
0x35, 0x31, 0x31, 0x31, 0x39, 0x31, 0x33, 0x30,
0x33, 0x5a, 0x30, 0x5f, 0x31, 0x10, 0x30, 0x0e,
0x06, 0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d,
0x01, 0x09, 0x01, 0x16, 0x01, 0x40, 0x31, 0x0a,
0x30, 0x08, 0x06, 0x03, 0x55, 0x04, 0x03, 0x13,
0x01, 0x20, 0x31, 0x0b, 0x30, 0x09, 0x06, 0x03,
0x55, 0x04, 0x06, 0x13, 0x02, 0x55, 0x53, 0x31,
0x0b, 0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x08,
0x13, 0x02, 0x57, 0x49, 0x31, 0x0b, 0x30, 0x09,
0x06, 0x03, 0x55, 0x04, 0x0a, 0x13, 0x02, 0x6f,
0x6e, 0x31, 0x0c, 0x30, 0x0a, 0x06, 0x03, 0x55,
0x04, 0x0b, 0x13, 0x03, 0x6f, 0x75, 0x6e, 0x31,
0x0a, 0x30, 0x08, 0x06, 0x03, 0x55, 0x04, 0x05,
0x13, 0x01, 0x20, 0x30, 0x1f, 0x30, 0x0d, 0x06,
0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01,
0x01, 0x01, 0x05, 0x00, 0x03, 0x0e, 0x00, 0x30,
0x0b, 0x02, 0x04, 0x6e, 0x86, 0xe5, 0x95, 0x02,
0x03, 0x01, 0x00, 0x01, 0xa3, 0x4d, 0x30, 0x4b,
0x30, 0x09, 0x06, 0x03, 0x55, 0x1d, 0x13, 0x04,
0x02, 0x30, 0x00, 0x30, 0x1d, 0x06, 0x03, 0x55,
0x1d, 0x0e, 0x04, 0x16, 0x04, 0x14, 0x1d, 0x29,
0x0a, 0xe9, 0xbb, 0xac, 0x0b, 0x1c, 0x4a, 0xe8,
0xf2, 0xa9, 0x06, 0x52, 0xfd, 0xab, 0xc2, 0xb5,
0x99, 0xc4, 0x30, 0x1f, 0x06, 0x03, 0x55, 0x1d,
0x23, 0x04, 0x18, 0x30, 0x16, 0x80, 0x14, 0x9f,
0xba, 0xff, 0x0d, 0x53, 0x2e, 0x12, 0x92, 0xbd,
0x47, 0x1a, 0xb7, 0x9f, 0x28, 0x8b, 0x9a, 0x5d,
0x74, 0xfa, 0x74, 0x30, 0x0d, 0x06, 0x09, 0x2a,
0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05,
0x05, 0x00, 0x03, 0x01, 0x00
```

For easy reference the following is the dictionary in textual and PEM format:

```
   Certificate:
       Data:
           Version: 3 (0x2)
           Serial Number: 1 (0x1)
           Signature Algorithm: sha1WithRSAEncryption
           Issuer: CN=http://www.com
           Validity
               Not Before: May 11 19:13:03 2010 GMT
               Not After : May 11 19:13:03 2011 GMT
           Subject: emailAddress=@, CN= , C=US, ST=WI, O=on,
   OU=oun/serialNumber=
           Subject Public Key Info:
               Public Key Algorithm: rsaEncryption
               RSA Public Key: (31 bit)
                   Modulus (31 bit): 1854334357 (0x6e86e595)
                   Exponent: 65537 (0x10001)
           X509v3 extensions:
               X509v3 Basic Constraints:
                   CA:FALSE
               X509v3 Subject Key Identifier:
                   1D:29:0A:E9:BB:AC:0B:1C:4A:E8:F2:A9:
   06:52:FD:AB:C2:B5:99:C4
               X509v3 Authority Key Identifier:
                   keyid:9F:BA:FF:0D:53:2E:12:92:BD:47:
   1A:B7:9F:28:8B:9A:5D:74:FA:74

       Signature Algorithm: sha1WithRSAEncryption
   -----BEGIN CERTIFICATE-----
   MIIBOTCCASOgAwIBAgIBATANBgkqhkiG9w0BAQUFADAZMRcwFQYDVQQDEw5odHRw
   Oi8vd3d3LmNvbTAeFw0xMDA1MTExOTEzMDNaFw0xMTA1MTExOTEzMDNaMF8xEDAO
   BgkqhkiG9w0BCQEWAUAxCjAIBgNVBAMTASAxCzAJBgNVBAYTAlVTMQswCQYDVQQI
   EwJXSTELMAkGA1UEChMCb24xDDAKBgNVBAsTA291bjEKMAgGA1UEBRMBIDAfMA0G
   CSqGSIb3DQEBAQUAAw4AMAsCBG6G5ZUCAwEAAaNNMEswCQYDVR0TBAIwADAdBgNV
   HQ4EFgQUHSkK6busCxxK6PKpBlL9q8K1mcQwHwYDVR0jBBgwFoAUn7r/DVMuEpK9
   Rxq3nyiLml10+nQwDQYJKoZIhvcNAQEFBQADAQA=
   -----END CERTIFICATE-----
```

Appendix B. Experimental results

To validate the approach the following tests were performed. 802.1AR certificates were compressed using CXF, and also compressed using DEFLATE without a preset dictionary in order to provide a comparison. CXF was implemented using a modified version of zlib-1.2.4's gzip utility, using the deflateSetDictionary function to make use of the CXF dictionary. In both cases, the compressor was set for the best possible compression (equivalent to the "gzip -9"). The table shows the compression ratio; that is, the output file size divided by the

original file size.

Input Data	DEFLATE w/o dictionary	CXF
RSA cert	780/753 = 1.04	647/753 = 0.86
ECC cert	369/355 = 1.04	258/355 = 0.73
RSA cert chain	1395/1572 = 0.89	1277/1572 = 0.81

In both the RSA and ECC case the certificate size is reduced by about 100 bytes. The differences in the computed compression ratio reflect the differences in size of the original certificate. CXF is able to compress even 802.1AR certificates, which minimize the number of ASN.1 fields. The compression ratio is better for the certificate chain because of the duplicate fields in each certificate; it would also be better for self-signed certificates. In contrast, DEFLATE without a preset dictionary is unable to compress the certificates, and it under-performs CXF on certificate chains.

The RSA certificate chain used in the experiment is:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            94:8c:34:15:cd:80:ee:f9
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=http://www.com
        Validity
            Not Before: May 11 19:13:12 2010 GMT
            Not After : May 10 19:13:12 2020 GMT
        Subject: CN=http://www.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:aa:2b:8f:2a:6d:6e:fa:b1:de:ec:43:13:a5:03:
                    1d:67:00:2d:87:d6:07:4b:d8:bd:54:f6:72:cf:62:
                    be:a7:73:b4:5b:81:a8:03:a1:7c:7d:ef:88:a2:2c:
                    61:26:be:f3:5c:ce:a9:1c:b0:67:3e:d0:7c:67:1d:
                    78:35:e6:1c:c6:ad:3b:8d:41:05:2c:04:34:39:ce:
                    b1:5e:01:ce:c2:da:d7:cc:77:04:7c:02:7d:71:7b:
                    0c:3f:95:8d:35:8c:47:b2:0e:24:cd:4f:3b:ae:c3:
                    13:cf:a9:c3:55:24:35:7d:dc:f0:a8:c2:a0:52:83:
                    ed:84:10:3c:95:c8:7b:c4:ed:df:b8:83:d5:3f:63:
                    c5:39:33:e5:1f:58:3f:2b:f2:e6:b6:8e:87:65:51:
                    9e:46:c7:ba:f0:08:cf:85:e9:93:11:4f:a2:ec:da:
```

```
                    61:0e:1c:6b:3f:34:f8:40:10:8e:4c:8d:1f:5d:00:
                    10:c4:12:b2:fa:fa:3b:60:e9:d3:8c:19:3f:be:74:
                    6b:d0:cb:79:3e:12:18:b3:86:f5:ba:0b:f0:8a:e7:
                    56:be:d2:e5:49:53:26:bc:9f:6a:0c:f8:27:69:a5:
                    ac:25:6c:da:af:55:d2:f4:02:b5:e5:ef:64:b8:85:
                    24:de:9c:95:05:80:96:96:23:94:0e:7b:b4:d3:b7:
                    66:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                1B:7E:0F:F8:41:75:16:D0:01:5C:53:11:FA:
  05:BA:D8:B0:B1:94:63
            X509v3 Authority Key Identifier:
                keyid:1B:7E:0F:F8:41:75:16:D0:01:5C:53:
  11:FA:05:BA:D8:B0:B1:94:63
                DirName:/CN=http://www.com
                serial:94:8C:34:15:CD:80:EE:F9

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
        4b:94:46:36:a3:43:16:cc:2f:3d:7b:d2:8f:96:51:b1:93:43:
        08:38:d7:3a:61:f6:09:d5:f9:aa:08:3c:7d:b2:4e:dc:24:e3:
        2c:76:ca:16:d2:b0:ad:4e:1a:6e:92:77:e1:47:67:1b:c8:30:
        1a:52:17:4e:91:61:09:99:5d:b0:de:ab:04:f6:b8:4e:49:47:
        0e:df:af:f9:e1:be:a2:f0:8b:5b:38:33:d6:b5:b9:6c:be:15:
        eb:ec:58:7d:6b:88:ab:a0:c7:7c:a5:12:43:a0:8d:82:1e:6b:
        d2:a0:7a:e6:03:04:ad:2d:9f:73:be:5a:b3:9e:77:a9:a2:8c:
        7a:a0:6e:1b:cb:f4:92:5a:32:b6:03:f8:6e:d0:e0:e0:6a:f1:
        8a:52:3c:56:da:e1:40:cb:ea:2e:49:6f:d0:8c:d5:3f:6f:3f:
        67:9f:a4:38:96:3e:d3:17:d4:09:55:6c:5c:3f:78:a3:06:73:
        dd:81:6b:54:89:1c:55:96:16:c9:5c:c3:b1:a4:57:8b:5c:24:
        65:d1:8a:78:ec:1a:53:17:1a:7a:df:86:5e:af:0f:26:c5:7c:
        51:da:bb:51:af:0d:1a:55:e2:23:3d:54:f2:1c:42:4a:1f:60:
        89:13:4f:e1:d5:f9:e1:c0:0a:a3:17:b1:b9:b1:50:6b:b5:5e:
        8c:1c:f0:f7
-----BEGIN CERTIFICATE-----
MIIDLzCCAhegAwIBAgIJAJSMNBXNgO75MA0GCSqGSIb3DQEBCwUAMBkxFzAVBgNV
BAMTDmh0dHA6Ly93d3cuY29tMB4XDTEwMDUxMTE5MTMxMloXDTIwMDUxMDE5MTMx
MlowGTEXMBUGA1UEAxMOaHR0cDovL3d3dy5jb20wggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCqK48qbW76sd7sQxOlAx1nAC2H1gdL2L1U9nLPYr6nc7Rb
gagDoXx974iiLGEmvvNczqkcsGc+0HxnHXg15hzGrTuNQQUsBDQ5zrFeAc7C2tfM
dwR8An1xeww/lY01jEeyDiTNTzuuwxPPqcNVJDV93PCowqBSg+2EEDyVyHvE7d+4
g9U/Y8U5M+UfWD8r8ua2jodlUZ5Gx7rwCM+F6ZMRT6Ls2mEOHGs/NPhAEI5MjR9d
ABDEErL6+jtg6dOMGT++dGvQy3k+EhizhvW6C/CK51a+0uVJUya8n2oM+Cdppawl
bNqvVdL0ArXl72S4hSTenJUFgJaWI5QOe7TTt2aLAgMBAAGjejB4MB0GA1UdDgQW
BBQbfg/4QXUW0AFcUxH6BbrYsLGUYzBJBgNVHSMEQjBAgBQbfg/4QXUW0AFcUxH6
BbrYsLGUY6EdpBswGTEXMBUGA1UEAxMOaHR0cDovL3d3dy5jb22CCQCUjDQVzYDu
```

```
+TAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQBLlEY2o0MWzC89e9KP
llGxk0MIONc6YfYJ1fmqCDx9sk7cJOMsdsoW0rCtThpuknfhR2cbyDAaUhdOkWEJ
mV2w3qsE9rhOSUcO36/54b6i8ItbODPWtblsvhXr7Fh9a4iroMd8pRJDoI2CHmvS
oHrmAwStLZ9zvlqznnepoox6oG4by/SSWjK2A/hu0ODgavGKUjxW2uFAy+ouSW/Q
jNU/bz9nn6Q4lj7TF9QJVWxcP3ijBnPdgWtUiRxVlhbJXMOxpFeLXCRl0Yp47BpT
Fxp634Zerw8mxXxR2rtRrw0aVeIjPVTyHEJKH2CJE0/h1fnhwAqjF7G5sVBrtV6M
HPD3
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=http://www.com
        Validity
            Not Before: May 11 19:13:17 2010 GMT
            Not After : May 11 19:13:17 2011 GMT
        Subject: CN=3
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:da:cf:0c:80:4b:7b:5d:c6:05:83:21:6d:6f:34:
                    c8:19:95:6a:74:0a:1b:26:fb:d4:35:7f:02:3d:47:
                    c0:a3:13:2b:e1:b6:0b:d1:91:0e:79:cf:d1:93:a4:
                    07:d9:ce:de:e9:3d:be:a9:19:c6:52:03:96:57:c1:
                    93:e8:18:b9:06:e1:a8:14:8a:22:70:72:85:52:56:
                    cd:d5:e3:e7:8c:22:b5:f3:ce:df:a2:67:59:7b:39:
                    a8:f0:69:1f:3d:73:4b:ce:f0:00:27:a5:81:5d:e2:
                    ba:26:3b:21:8a:74:11:d3:a3:23:fb:40:27:b2:0f:
                    ac:61:63:31:13:cf:d0:27:47:4b:bf:d9:67:f5:29:
                    4b:a6:76:0a:28:dc:55:68:57:a2:de:be:6f:da:92:
                    24:1e:99:80:81:00:0d:47:c7:b7:98:c5:97:9f:ac:
                    7a:c4:bc:16:93:b8:9d:27:f9:5a:4f:63:b5:45:6a:
                    b2:cb:26:71:b6:94:6b:65:ff:33:94:0f:da:7e:40:
                    74:19:ca:34:3e:3d:e0:b5:cc:d9:b9:e8:bc:0b:95:
                    d4:9c:23:89:a0:6b:b3:7b:ef:8e:a5:79:6e:e6:9f:
                    6c:5d:1a:2f:39:b9:78:1a:bd:81:37:87:45:60:fb:
                    d6:cb:94:33:c7:e3:06:60:9a:7a:f0:f2:b0:a7:cb:
                    6d:99
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                BB:A7:D8:EE:6D:EE:79:60:0D:AB:38:35:5C:0C:
7E:70:EA:C1:FF:6E
            X509v3 Authority Key Identifier:
```

```
                keyid:1B:7E:0F:F8:41:75:16:D0:01:5C:53:11:
   FA:05:BA:D8:B0:B1:94:63

       Signature Algorithm: sha1WithRSAEncryption
           46:9b:5e:d5:ab:e7:66:4d:2c:d8:fd:44:67:08:87:5e:57:a7:
           69:f4:f4:1b:46:39:61:b3:d5:f4:9c:11:b6:17:8c:a8:bc:10:
           0b:43:13:b3:ab:a0:99:a8:67:5c:b1:82:df:54:44:46:89:70:
           d4:5d:0d:7c:16:8e:29:fa:c5:ae:2d:08:c4:65:b4:8e:ce:58:
           04:f7:50:99:8f:c7:a3:bb:4d:da:64:72:99:a0:dd:c5:f5:fd:
           c8:a6:be:78:ce:96:19:a9:89:f0:01:1d:88:e0:56:6e:0e:16:
           d9:3a:2e:63:d4:07:54:9a:6d:b9:b5:e4:52:9c:ed:4d:6f:26:
           cd:8b:de:1e:b7:cb:cd:1c:76:e2:87:4e:9b:7b:3b:9d:5f:63:
           3f:7f:2e:e7:ac:84:e9:3d:68:38:60:ae:85:b4:92:99:62:3c:
           5d:50:b4:94:65:89:3b:42:30:16:68:0a:18:21:23:de:7b:83:
           2b:df:1a:2c:5c:4d:19:94:3c:ba:be:76:eb:50:4c:b2:8f:bc:
           3b:d9:6b:50:30:06:c7:c4:17:a0:1a:16:75:83:1e:f0:de:c2:
           4f:da:2f:3a:a0:c6:d3:82:6a:10:7e:b7:ed:72:cd:ea:d2:de:
           60:eb:57:6e:9b:ab:02:c1:09:04:c5:f8:a2:dd:59:70:69:d3:
           b7:b6:ae:91
   -----BEGIN CERTIFICATE-----
   MIIC7TCCAdWgAwIBAgIBAzANBgkqhkiG9w0BAQUFADAZMRcwFQYDVQQDEw5odHRw
   Oi8vd3d3LmNvbTAeFw0xMDA1MTExOTEzMTdaFw0xMTA1MTExOTEzMTdaMAwxCjAI
   BgNVBAMTATMwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDazwyAS3td
   xgWDIW1vNMgZlWp0Chsm+9Q1fwI9R8CjEyvhtgvRkQ55z9GTpAfZzt7pPb6pGcZS
   A5ZXwZPoGLkG4agUiiJwcoVSVs3V4+eMIrXzzt+iZ1l7OajwaR89c0vO8AAnpYFd
   4romOyGKdBHToyP7QCeyD6xhYzETz9AnR0u/2Wf1KUumdgoo3FVoV6Levm/akiQe
   mYCBAA1Hx7eYxZefrHrEvBaTuJ0n+VpPY7VFarLLJnG2lGtl/zOUD9p+QHQZyjQ+
   PeC1zNm56LwLldScI4mga7N7746leW7mn2xdGi85uXgavYE3h0Vg+9bLlDPH4wZg
   mnrw8rCny22ZAgMBAAGjTTBLMAkGA1UdEwQCMAAwHQYDVR0OBBYEFLun2O5t7nlg
   Das4NVwMfnDqwf9uMB8GA1UdIwQYMBaAFBt+D/hBdRbQAVxTEfoFutiwsZRjMA0G
   CSqGSIb3DQEBBQUAA4IBAQBGm17Vq+dmTSzY/URnCIdeV6dp9PQbRjlhs9X0nBG2
   F4yovBALQxOzq6CZqGdcsYLfVERGiXDUXQ18Fo4p+sWuLQjEZbSOzlgE91CZj8ej
   u03aZHKZoN3F9f3Ipr54zpYZqYnwAR2I4FZuDhbZOi5j1AdUmm25teRSnO1NbybN
   i94et8vNHHbih06bezudX2M/fy7nrITpPWg4YK6FtJKZYjxdULSUZYk7QjAWaAoY
   ISPee4Mr3xosXE0ZlDy6vnbrUEyyj7w72WtQMAbHxBegGhZ1gx7w3sJP2i86oMbT
   gmoQfrftcs3q0t5g61dum6sCwQkExfii3VlwadO3tq6R
   -----END CERTIFICATE-----
```

Authors' Addresses

David A. McGrew
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
USA

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm

Max Pritikin
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
USA

Phone: (408) 424 5141
Email: pritikin@cisco.com