SPKI Examples
INTERNET-DRAFT
Expires: 15 September 1998

Bill Frantz
Electric Communities

Butler Lampson
Microsoft

Ron Rivest
MIT Laboratory for Computer Science

Brian M. Thomas
Southwestern Bell

Tatu Ylonen
SSH

10 March 1998

SPKI Examples
---- --------

<draft-ietf-spki-cert-examples-01.txt>

Status of This Document

This document supersedes the draft filed under the name draft-ietf-spki-cert-examples-00.txt. This draft contains examples of SPKI structures for various applications. The structure definition is to be found in draft-ietf-spki-cert-structure-*.txt and the theory behind SPKI certificates is to be found in draft-ietf-spki-cert-theory-*.txt.

Distribution of this document is unlimited. Comments should be sent to the SPKI (Simple Public Key Infrastructure) Working Group mailing list <spki@c2.net> or to the authors.

This document is an Internet-Draft. 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. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a

Ellison, et al.
To learn the current status of any Internet-Draft, please check the
id-abstracts.txt listing contained in the Internet-Drafts Shadow
Directories on ds.internic.net (East USA), ftp.isi.edu (West USA),
nic.nordu.net (North Europe), ftp.nis.garr.it (South Europe),
munnari.oz.au (Pacific Rim), or ftp.is.co.za (Africa).

Abstract

SPKI structures are defined for public keys, hash values, access
control list (ACL) entries and certificates. This document gives
examples of such structures for real world applications. The
examples here are not tied to any specific application and should be
taken as informative examples rather than standard formats. However,
one applications are fielded using such structures and we have
experience with them, we can consider publishing those formats as
proposed standards. That effort is considered out of scope for this
document.
Table of Contents

Status of This Document....................................1
Abstract...................................................2

Table of Contents........................................3

1. SPKI Basic Structures....................................4

2. Tag Examples............................................6
2.1 FTP....................................................6
2.2 HTTP...................................................6
2.3 TELNET................................................6
2.4 Public Key Protected File System tags.................6
2.5 Authority to spend money............................8
2.6 Purchase order signing permission.....................8

3. Keyholder examples.....................................9
3.1 Locator certificate.....................................9
3.2 Insurance certificate................................10
3.3 Auto-certificate......................................10

4. Object certificates....................................11
4.1 PICS-like ratings certificate........................11
4.2 Virus checking certificate.............................11

5. Full sequence, with auto-certificate....................12

Authors’ Addresses........................................14
Expiration and File Name..................................15
1. SPKI Basic Structures

An SPKI certificate has five fields of interest during trust computations:
1. Issuer -- the principal issuing this certificate and granting the permission or rights it communicates
2. Subject -- the principal or name acquiring the permission or rights
3. Delegation -- a flag noting whether the Subject is also acquiring the right to delegate all or part of the permission it acquires through this certificate
4. Authorization <tag> -- a field specifying the permission being communicated
5. Validity -- a specification of the dates or on-line conditions under which the certificate is assumed to be valid

An SPKI ACL entry generates the same five fields, for trust computation, but does not need to contain them all. The Issuer is always "self" and is omitted from the structure. If the ACL is held in editable memory, then the Validity fields can be omitted on the assumption that if the owner of the ACL wants to declare an entry invalid, he can edit it immediately. However, for ACLs built into executable software, running remotely, this assumption may not hold true and one might want validity fields.

The purpose of SPKI structures is to communicate permission or rights from one keyholder to another. If we consider this flow to go horizontally from left to right, at the left end of the flow one finds the injection of permission. This is always in the form of an access control list (ACL) entry. An ACL entry is like a certificate, except that it has no issuer and is not signed. It is protected in the application by other means.

To the right of the ACL entry, one finds certificates. At least for the sake of discussion, there are two forms of certificate: one that communicates permissions and one that defines names. It is possible to unify the two, but for our purposes, they will be considered separately.

A basic SPKI certificate communicates a permission from one key to another key (equivalently from one principal to another principal or from one keyholder to another keyholder). In the process, it can pass all permissions it has been handed, in which case it uses (tag (*)) as its tag field or it can pass only certain permissions. The former can be considered a group membership certificate. That is, the subject key is a full member of the group defined by the issuer key. Any permissions given the issuer are inherited by the subject. If the subject is a name rather than a key, then the permission is being granted to the named key. That is, it is assumed that the name will be reduced to a key before permissions propagate. This is
especially important when considering the propagation of the 
permission to delegate. A named key may be given a permission 
without the permission to delegate, but that name could define a 
group and could therefore represent implicit delegation.

A name certificate maps a name (in the issuer field) to either a key 
or a name (in the subject field). It is always a group membership 
style of certificate -- that is with (tag (*)).

The bulk of what changes with each example in the next section is the 
<tag> field, so we assume the following certificate template:

(cert
  (issuer (hash sha1 |TLCgPLF1GTzgUbcaYlkW8kGTEnUk=|))
  (subject (hash sha1 |Ve1L/7MqiJcj+LSa/l10f13tuTQ=|))
  ...
  (not-before "1998-03-01_12:00:00")
  (not-after "2012-01-01_00:00:00")
)

or ACL template:

(acl
  (entry
    (subject (hash sha1 |Ve1L/7MqiJcj+LSa/l10f13tuTQ=|))
    ...
  )
)

with a tag and possibly delegation field where the "..." is in the 
middle of the structures above.
2. Tag Examples

The <tag> fields listed here are not meant to be an exhaustive list of all possible <tag>s. Such is not possible. The final arbiter of what needs to be an <tag> and what parameters a particular <tag> needs is the designer of the code that verifies a certificate, e.g., to grant access. Listed here are <tag> fields we suspect might be useful and we present these here as a guide to the developer's imagination.

2.1 FTP

(tag (ftp cybercash.com cme))

This <tag> indicates that the Subject has permission to do FTP into host cybercash.com as user cme.

2.2 HTTP

(tag (http http://acme.com/company-private/personnel ))

This <tag> gives the Subject permission to access the web page at the given URI. To give permission for an entire tree under a given URI, one might use:

(tag (http (* prefix http://acme.com/company-private/personnel/ )))

2.3 TELNET

(tag (telnet clark.net cme ))

This <tag> gives the Subject permission to telnet into host clark.net as user cme.

2.4 Public Key Protected File System tags

(tag (pkpfs <pathname> <access> ))

refers to a hypothetical distributed file system whose access is controlled by public key challenge/response. The <pathname> can be a single pathname, a set of files (specified by normal "*" convention) or a directory sub-structure (specified by (* prefix ...)). For
example,

(tag (pkpfs //ftp.clark.net/pub/cme/spki.txt (* set read write)))

would give read and write access to that one file on that one host
machine, while

(tag (pkpfs //ftp.clark.net/pub/cme/spki.* (* set read write)))

would give read and write access to all files starting with "spki.".

(tag (pkpfs //ftp.clark.net/pub/cme/ add))

would give permission to add new files to that directory and

(tag (pkpfs (* prefix //ftp.clark.net/pub/cme/) read ))

would give read permission to all files in that directory.

The full specification of possible <access> specifications is up to
the implementer of this file system.

One might also want to grant disk quotas for this file system, e.g.,
via:

(tag (pkpfs-quota (* range le "50000" )))

One could have used

(tag (pkpfs-quota "50000" ))

to express the quota, but the (* range ...) form permits the user to
delegate a sub-quota to some other user.

Accounting for such quotas could be interesting, but one would
probably want to charge each K of disk to all users in the
certificate delegation chain, since the disk accounting system is not
aware of acts of delegation and therefore can not reduce the quota of
the delegator. To maintain good accounting, this would require each
file to have a list of accounts (key hashes) against which its size
is charged -- so that when a file is deleted, the appropriate quotas
can be adjusted. Since there would probably be only a small number
of different such chains of delegation, one might keep such lists
separately from file nodes. However, these are details left to the
designer of the PKPFS.
2.5 Authority to spend money

(tag (spend <bank> <account> (* range le <amount> )))

indicates that the subject has authority to authorize spending up to
<amount> per electronic check from <account> at <bank>. For example,

(propagate)
(tag (spend BankBoston "011000390 436 20608" (* range le "500.00")))

permits someone to spend up to 500.00 per electronic check from the
indicated checking account at BankBoston. [The account number above
was chosen randomly and is hopefully not a valid BankBoston account.]

(propagate) implies that this account holder has the permission to
delegate check-writing ability to others (e.g., family members or
temporary public keys (for use on a laptop while traveling)).

2.6 Purchase order signing permission

(tag (purchase (* range le <amount>) (* set <<items>> )))

might indicate permission to issue a purchase order. The amount of
the purchase order is limited by the second element of the (purchase
) S-expression and, optionally, a list of purchasable items is given
as the third element. The company whose purchase orders are
permitted to be signed here will appear in the certificate permission
chain leading to the final purchase order. Specifically, that
company’s key will be the issuer at the head of the (purchase )
chain.
3. Keyholder examples

The set of examples in this section deals with (keyholder) subjects. These are for human consumption, since the subject is a human being (the keyholder of a particular key) rather than a key in cyberspace. Note that it is not meaningful for a keyholder certificate to have the (propagate) flag.

3.1 Locator certificate

A locator certificate is one that is issued by a company that promises to keep track of the indicated keyholder until the not-after date, and promises to serve the keyholder with papers for the indicated fee in the indicated currency, up until the not-after date.

This kind of certificate might be used to back up a legal contract in which the keyholder might have to pay damages at some future date, in the event of non-performance, so that it becomes important to know how to sue that keyholder at that later date. An old fashioned "identity certificate" doesn't serve as well here because it lists some information about the keyholder that might be used to track that person down, but that information is old by the time the contract is signed and it is the ability to locate that keyholder at the end of the contract that is important. Maintaining that ability costs money and therefore the issuer of the locator certificate expects to be paid for its services. The issuer might also charge the keyholder at the time of certificate issuance, but that fee need not be indicated in the certificate.

For example:

```
(cert
  (issuer (hash md5 |u2k173Mlo1zkGmHdBf==|))
  (subject (keyholder (hash md5 |kuXyqx8jYwcZ/j7VfB+yg==|)))
  (tag (tracking-fee "150" USD))
  (not-after "2003-01-01_00:00:00")
)
```

notes in its tag field that the issuer will serve papers on the indicated keyholder for a tracking fee of $150 until the beginning of 2003.
3.2 Insurance certificate

Instead of tracking down a keyholder and serving papers on him or her, the person relying on a certificate might prefer that some insurance company pay the penalty amount in the event of non-performance, and then worry on its own about collecting that fee (plus damages, no doubt) from the keyholder. This kind of certificate, like an insurance policy, will cost the user of the certificate money at the time it is issued. It is therefore good for only one user.

For example:

(cert
  (issuer (hash md5 |u2k173Ml0bh5olzkGmHdbA==|))
  (subject (keyholder (hash md5 |kuXyqx8jYWdZ/j7Vffr+yg==|)))
  (tag (insured "50000" USD) (to (hash md5 |1r8ICryJw6v/B4MQdTU/Q==|))
       (for "Failure to perform under contract (on file): "
        (hash md5 |gPA50iM6yETsixLgo2kV1A==|))
  (not-after "2003-01-01_00:00:00")
)

represents a promise to pay $50000 to |1r8ICryJw6v/B4MQdTU/Q==| in the event that the keyholder of |kuXyqx8jYWdZ/j7Vffr+yg==| fails to fulfill the contract, |gPA50iM6yETsixLgo2kV1A==|.

3.3 Auto-certificate

There are some pieces of information about which the proper issuer is the subject. The name a keyholder prefers to be called, the phone number or e-mail address at which the keyholder can be reached, etc., are all items of information about which the keyholder is the best authority.

(cert
  (issuer (hash sha1 |1QvsTPF0/vqHPGODX/yEN8ro+sc=|))
  (subject (keyholder (hash sha1 |1QvsTPF0/vqHPGODX/yEN8ro+sc=|)))
  (tag
    (* set
     (name "Carl")
     (e-mail "cme@acm.org")
   )
  )
)

(KDQ6Y2VydCg2Om1zc3V1cig0Omhhc2g00nNoYTEyMDrVC+xM8XT++oc8Y4NFs/IQ3yuj6xykpKDo6c3ViAmVjdCg5OmmlteWhvbGR1cig0Omhhc2g00nNoYTEyMDrVC+xM8XT++oc8Y4NFs/IQ3yuj6xykpKSkp)

Ellison, et al.                                                [Page 10]
4. Object certificates

The certificates in this section have subjects that are objects rather than keys or keyholders. As with keyholder certificates, it is not meaningful for object certificates to have the (propagate) flag.

4.1 PICS-like ratings certificate

(cert
  (issuer (hash md5 |Ut9m14byPzdbCNZwdDjNQg==|))
  (subject
    (object-hash
     (hash md5 |vN6ySKWE9K6t6cP9U5wntA==|
      http://www.clark.net/pub/cme/home.html)))
  (tag (ratings (sex "1") (violence "0") (crypto "9"))))
)

This certificate should be self-explanatory. This assigns attributes to the indicated object (a web page with the indicated hash).

4.2 Virus checking certificate

(cert
  (issuer (hash md5 |Ut9m14byPzdbCNZwdDjNQg==|))
  (subject
    (object-hash
     (hash md5 |szKslSK+SNzIsHH3wJAsTQ==| runemacs.exe)))
  (tag virus-free))
)

This certificate is even simpler. The issuer is declaring that the indicated object has been checked and is virus free, in the issuer’s opinion.
5. Full sequence, with auto-certificate

For one full example of a real certificate, the following sequence presents the public key used, calls for the verifier to hash it (and store it away, to be referred to later by its hash), gives a certificate body and then a signature (which by side-effect calls for the previous object to be stored and hashed by the signature algorithm’s hash function). The example used is a temporary auto-certificate.

(sequence
  (public-key
    (rsa-pkcs1-md5
      (e #11#)
      (n 5dC0qBNMzjRjF+z8Tr1CEn9May0vXB00WYrtw/7aH2WAz+x8er0WR+yn1CTR1S/68IW86Gclx8hiPycMeb1ICAbSYjHC/ghq2mwCZO7VQXJENzYr45))
  )
  (do hash md5)
  (cert
    (issuer (hash md5 |+gbUgUltGysNgewRwu/3hQ==|))
    (subject
      (keyholder (hash md5 |+gbUgUltGysNgewRwu/3hQ==|)))
    (tag
      (* set
        (name "Carl M. Ellison")
        (street "207 Grindall St.")
        (city "Baltimore MD")
        (zip "21230-4103"))
      (not-after "1998-04-15_00:00:00")
    )
    (signature
      (hash md5 |54LeOBILOUpskE5xRTSmmA==|
        hash md5 |+gbUgUltGysNgewRwu/3hQ==|
        |HU6ptoaEd7v4rTKBIrnpjBqDiWX9fBFbLy/MeHyJRry8iA3+f9Yh/bbuIn9xgcu11Z3gu9ULenu5bSbiJGDwxKlOuhTRG+lo1ZWhAaD5YnqmV9hKhws7U4KoenAhfouRshc8Wgb3RmMepi6t80Arcc6viuAF4CP+Zxc=|)
    )
  )
)

or, in base64:

{KDg6c2VxdWVuY2UoMTA6cHVibGljLWtleSgxMzpyc2EtcGtjczEtbbW1KDE6ZTE6ESkoMTpuMTI5QgXxfJW01b0qXw18Ej7ZcbZWKn1ItpE3BE SIDC0oypbR5xovbOXQ3EG1TT40Yxs/E6yAhJ/TLMtL1wdNfMebpC+2h9lgGcfsfHgzlKfSp9Qk0SOvr+vCFgelnNcfYj8nD4Nag0miXwvr41atpAmTulUFyРЕdC2K+O8kSgyOmRvNdpoYXNoMzptZDUpKDQ6Y2VydCg2Om1Zc3Vlcig0omhch2g0m1KTE2ovG1FJbRsrDYHsEclv94UpKSg30nN1YmplY3Q0ToPrZX1ob2xkZXioNDo1YXNoMzptZDUxNjy6Bt0SBW0bKw2B7BHC7/eFKskpKDM6dGrhKDE6KjM6c2V0RDQ6bmFtZTE1OkNhcmwgtS4gRwxasNvbk0NjpxdHJ
Authors’ Addresses

Carl M. Ellison
CyberCash, Inc.
207 Grindall Street
Baltimore MD 21230-4103 USA

Telephone: +1 410-727-4288
+1 410-727-4293 (FAX)
+1 703-620-4200 (main office, Reston, Virginia, USA)
EMail: cme@cybercash.com
cme@acm.org
Web: http://www.clark.net/pub/cme

Bill Frantz
Electric Communities
10101 De Anza Blvd.
Cupertino CA 95014

Telephone: +1 408-342-9576
Email: frantz@netcom.com

Butler Lampson
Microsoft
180 Lake View Ave
Cambridge MA 02138

Telephone: +1 617-547-9580 (voice + FAX)
EMail: blampson@microsoft.com

Ron Rivest
Room 324, MIT Laboratory for Computer Science
545 Technology Square
Cambridge MA 02139

Telephone: +1-617-253-5880
+1-617-258-9738 (FAX)
Email: rivest@theory.lcs.mit.edu
Web: http://theory.lcs.mit.edu/~rivest

Brian Thomas
Southwestern Bell
One Bell Center, Room 23Q1
St. Louis MO 63101 USA

Telephone: +1 314-235-3141
Expiration and File Name

This draft expires 15 September 1998.

Its file name is draft-ietf-spki-cert-examples-01.txt