Internet Draft
draft-ietf-pkix-okid-01.txt
February 24, 2002
Expires in six months

Paul Hoffman
VPN Consortium

Out-of-Band Certificate and Key Identifier Protocol (OCKID)

Status of this memo

Abstract

In general, certificates need not be communicated with communication or storage media that are integrity-secure or authentic. This is because certificates are digitally signed and users are expected to validate the signatures using configured trust anchors. However, distribution of trust anchor certificates, self-signed end-entity certificates, or bare (unsigned) public keys requires a mechanism for establishing the authenticity of the certificate or public key.

When a user receives a certificate or public key of this sort through a communication or storage medium that is not known to be integrity-secure and authentic, the user needs to verify the value of the certificate or public key over an integrity-secure, authentic, out-of-band channel. The Out-of-Band Certificate and Key Identifier Protocol (OCKID) is a user-friendly key identifier that can be used for the out-of-band verification of a certificate or bare public key.

1. Introduction

A typical scenario for using digital certificates is that an end user or system will acquire at one or more CA certificates for use as a trust anchors. Trust anchor certificates are usually self-signed (that is, the Issuer and Subject names are the same). It is common to acquire trust

anchor certificates through channels that are not integrity-secure or authenticated such as unprotected data streams or web sites. In such a scenario, an attacker could substitute a certificate of his own choosing. The same attack can be used in protocols that use bare public keys as trust anchors.

In order to detect and reject such attacks, a user receiving a certificate or bare public key must use a method to verify that the certificate or public key is the one that the sender intended. One means of achieving this is to communicate the hash of the certificate or public key over a communication path which the user believes to be integrity-secure and authenticated. (The certificate or public key itself is considered too large a value for users to conveniently and reliably communicate.)

The OCKID provides a standard representation for out-of-band human communication of the hash of a certificate or public key, in support of the security requirements described here. The OCKID is designed to be easy to transmit through out-of-band mechanisms. Specifically, it is easy to read on the telephone, assuming that both parties understand the names of the ASCII alphabetic characters. It is also easy to type on a keyboard with ASCII capabilities.

2. OCKID format

An OCKID string has the following format:

WW-XXXX-XXXX-XXXX-XXXX-XXXX

where "WW" is the type identifier and the Xs are characters from the hash string. The dash characters in the OCKID string MUST be used. All characters in the OCKID string MUST be uppercase. However, a system that is receiving an OCKID MAY accept the OCKID string in uppercase or lowercase and MAY accept the string without hyphens.

2.1 Type identifiers

This document defines four OCKID profiles. The identifiers for these profiles are:

EE - PKIX end entity certificates

CA - PKIX certification authority certificates

AA - PKIX attribute authorities

BK - Bare key

Other profiles may be defined later in other RFCs (see the IANA Considerations appendix). Each new profile will define its own two-letter type identifier.

2.2 Hash string

The hash string consists of 16 characters that represent an 80-bit hash of a key. The hash string is created using the following steps:

1. Let A equal the SHA-1 [SHA1] hash of the certificate or bare public key being identified. This will always be 160 bits.

2. Let B equal the left-most 80 bits of A (assuming big-endian representation of A).

3. Let C equal the Base32 transformation of B (see Table 1 below). This will always be 16 characters.

4. Let the output be the first four characters of C, followed by a hyphen, followed by the second four characters of C, followed by a hyphen, followed by the third four characters of C, followed by a hyphen, followed by the fourth four characters of C, followed by a hyphen, followed by the last four characters of C.

The Base32 algorithm is as follows:

1. If there are no more bits in the input, stop.

2. If there are fewer than five bits in the input, stop with a fatal error.

3. Remove the left-most five bits from the input and call this value X. Look up X in Table 1, and add the corresponding character for X to the string C.

4. Go to step 1.

Table 1: Base32 conversion

Bits	Character	Bits	Charac
00000	A	10000	S
00001	B	10001	T
00010	C	10010	U
00011	D	10011	V
00100	E	10100	W
00101	F	10101	X
00110	G	10110	Y
00111	H	10111	Z
01000	J	11000	2
01001	K	11001	3
01010	L	11010	4
01011	M	11011	5
01100	N	11100	6
01101	P	11101	7
01110	Q	11110	8
01111	R	11111	9

(Note that all the characters are uppercase and that the characters "I", "O", "0", and "1" are not used.)

3. OCKID profile for PKIX end entity certificates

An end entity certificate is a PKIX [PKIX] certificate that does not have the CA bit set in the certificate. The CA bit MUST NOT be set in any certificate used with this profile.

The type identifier in the OCKID for a PKIX end entity certificate is "EE".

Systems that check an end entity OCKID MUST verify that the CA bit is not set in the certificate for which the OCKID is being matched. Such systems MUST verify that the type identifier in the OCKID is "EE".

Because the result of matching the OCKID to the end entity certificate is that the certificate will now become inherently trusted, the system MUST inform the user that the end entity certificate has become inherently trusted. The system SHOULD give the user a method for later removing the trust in the end entity certificate. The system MUST also check whether the certificate is properly signed, that is, that the public key in the certificate is in fact correctly verifies the contents of the certificate.

4. OCKID profile for PKIX CA certificates

A certification authority certificate is a PKIX certificate that has the CA bit set in the certificate. The CA bit MUST be set in any certificate used with this profile. Note that this certificate may or may not be a "root" certificate (that is, the issuer name may or may not be the same as the subject name).

The type identifier in the OCKID for a PKIX CA certificate is "CA".

Systems that check a CA certificate OCKID MUST verify that the CA bit is set in the certificate for which the OCKID is being matched. Such systems MUST verify that the type identifier in the OCKID is "CA".

Because the result of matching the OCKID to the CA certificate is that the certificate will now become a trust anchor, the system MUST inform the user of each of the following:

- That the certificate has become a trust anchor

- The policies used by the issuer of this certificate to issue subordinate certificates ([PKIX] section 4.2.1.5)

- The basic constraints placed on the issuer of this certificate, such as the depth of subordinate chain that can be issued under this certificate ([PKIX] section 4.2.1.10)

- The types of names for which the issuer of this certificate can create certificates ([PKIX] section 4.2.1.11)

- The policy constraints placed on the issuer of this certificate ([PKIX] section 4.2.1.12)

The system SHOULD give the user a method for later removing the trust in the CA certificate. The system MUST also check whether the certificate is properly signed, that is, that the public key in the certificate is in fact correctly verifies the contents of the certificate.

5. OCKID profile for PKIX attribute authority certificates

Attribute certificates are described in [ATCERT]. The entities that can issue attribute certificates are called "attribute certificate issuers" and "attribute authorities". The rules governing the certificates that can be used to issue attribute certificates are given in section 4.5 of [ATCERT]. Basically, any end entity (but not a CA) can be an attribute authority.

The type identifier in the OKID for a PKIX attribute authority certificate is "AA".

Systems that check an attribute authority OKID MUST verify that the CA bit is not set in the certificate for which the OKID is being matched. Such systems MUST verify that the type identifier in the OKID is "AA".

If the result of matching the OKID to the attribute authority certificate is that the certificate will now become inherently trusted for attribute certificates, the system MUST inform the user that the attribute authority certificate has become inherently trusted. The system SHOULD give the user a method for later removing the trust in the attribute authority certificate.

The system SHOULD give the user a method for later removing the trust in the AA certificate. The system MUST also check whether the certificate is properly signed, that is, that the public key in the certificate is in fact correctly verifies the contents of the certificate.

6. OCKID profile for bare public keys

A bare public key is simply a bit string.

The type identifier in the OCKID for bare public keys is "BK".

In systems where the result of matching the OCKID to the bare public key is that the public key will now become inherently trusted, the system MUST inform the user that the bare public key has become inherently trusted. The system SHOULD give the user a method for later removing the trust in the bare public key.

7. Example

Assume that the SHA-1 hash of the public key in an end entity certificate is cc487d7aa6228613e997d760a10a9e920fb06f49. The steps for creating the OKID string are:

```
A = (in hex) cc487d7aa6228613e997d760a10a9e920fb06f49

B = (in hex) cc487d7aa6228613e997

                c    c    4    8    7    d    7    a    a    6
B = (in binary) 1100 1100 0100 1000 0111 1101 0111 1010 1010 0110
                2    2    8    6    1    3    e    9    9    7
                0010 0010 1000 0110 0001 0011 1110 1001 1001 0111

                  3     T     E     H     4     8     X     G
B = (as quintets) 11001 10001 00100 00111 11010 11110 10101 00110
                  E     L     D     B     H     4     N     Z
                  00100 01010 00011 00001 00111 11010 01100 10111

C = 3TEH48XGELDBH4NZ

Output = EE-3TEH-48XG-ELDB-H4NZ
```

8. Security Considerations

8.1 Strength of the 80-bit hash

An 80-bit hash such as the one used in this protocol is currently adequate for preventing substitution attacks against an unprotected certificate. If Mallory knows Alice's certificate and wants to create a different certificate that he can substitute for Alice's during an unprotected certificate exchange, he would have to generate approximately 2^79 certificates in order to find one that would impersonate Alice.

Note that Mallory would not have to compute 2^79 signatures; he could alter the contents of the unsigned parts of the certificate and therefore only need to calculate 2^79 hashes. Creating 2^79 hashes is currently infeasible.

Chips that do DES encryption at 100 megabits (1e8) per second cost about $100 dollars in quantity today. Assuming that SHA-1 runs about as fast as DES, and that a typical certificate is 5000 bits long, a $100 chip could do about 2e4 hashes per second on these 5e3-bit certificates. It takes each $100 chip about trillion (1e12) years to do 2^79 hashes:

```
   x = 2^79 hashes / ( (2e4 hash/sec) * (3.16e7 sec/year) )
```

If Mallory wanted to duplicate Alice's OCKID within 10 years, he would have to spend about ten trillion dollars.

OCKIDs for bare public keys are even stronger, because Mallory would have to produce 2^79 valid key pairs instead of 2^79 hashes in order for the result to be useful. Creating key pairs is orders of magnitude slower than calculating hashes.

Moore's laws probably apply to hash accelerator chips, but even with the speed of the chips doubling every 1.5 years, the attack for Mallory is infeasible for certificates and keys that have a lifetime of less than **20 years. In 20 years, today's chips will run about 2^12 (4096) times as** fast, meaning that Mallory would need to spend tens of billions of dollars to duplicate a certificate within a year.

8.2 Birthday attack

If Mallory is only attacking Alice's OCKID, he has to do 2^79 operations. If he can attack a very large number of certificates or keys, the birthday attack makes it easier for him to find a certificate or key to attack, but doesn't make it easier for him to specifically attack Alice. Mallory needs to do a smaller number of hash operations in order to find one collision he can use.

This effect is mitigated by the fact that Mallory not only needs to find a matching OCKID, he also must be able to inject the bogus certificate or key into the stream just as Bob is retrieving it. Because the main use of OCKIDs is for trusted certificates and keys, people will retrieve them much less often than certs in a hierarchical trust chain. Injecting bogus certificates or keys is easy for Mallory if he controls a common certificate or key repository.

8.3 Security of the out-of-band transfer

The out-of-band transfer must be secure from substitutions. That is, an attacker must not be able to act as a man-in-the-middle for the transfer of the OCKID.

9. Acknowledgements

Stephen Kent and Russ Housley contributed a great deal to the initial draft of this document. Peter Gutmann helped refine the Base32. Many people pointed out problems with the initial idea of hashing just the public key in a PKIX certificate.

10. References

[ATCERT] Internet Attribute Certificate Profile for Authorization, draft-ietf-pkix-ac509prof

[PKIX] Internet X.509 Public Key Infrastructure Certificate and CRL Profile, draft-ietf-pkix-new-part1

[SHA1] US Secure Hash Algorithm 1 (SHA1), RFC 3174

A. IANA Considerations

New OCKID profiles can be defined only in standards-track RFCs. The RFC defining the OCKID profile must fully define the environment for the key

usage, and must specify a type identifier that does not conflict with any previous OCKID type identifier.

IANA will set up a registry of OCKID profiles. Each entry in the registry will have three fields:

- Profile name: a string describing the profile

- Defining RFC: the standards-track RFC for the profile

- Type identifier: the two-letter identifier. The letters must be upper-case ASCII. The type identifiers must be unique across all entries.

The first four entries in the registry are:

Profile name: PKIX end entity certificates
Defining RFC: [this RFC]
Type identifier: EE

Profile name: PKIX CA certificates
Defining RFC: [this RFC]
Type identifier: CA

Profile name: PKIX attribute authority certificates
Defining RFC: [this RFC]
Type identifier: AA

Profile name: Bare public keys
Defining RFC: [this RFC]
Type identifier: BK

B. Author's Address

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
paul.hoffman@vpnc.org

C. Changes between -00 and -01

- Changed the title.

- Changed the hash from being over the key to being over the whole cert.

- Changed the Base32 characters.

- Added profiles for attribute authorities and bare keys

- Changed the Security Considerations section considerably.