Public Key Certificates

Simon Foley

November 18, 2013

Key Management Revisited

Long Term Keys. Used for authentication, including access control and integrity. Used for confidentiality: establishing session keys and protecting stored data.

Short Term Session Keys. Generated automatically and invisibly. Used for one message or session and then thrown away.

Key Management Problems. Key certification; Distributing Keys (obtaining somebody's public key or distributing your own); Establishing a shared key with another party (confidentiality: is it really known only to the other party? authentication: is it really shared with the intended party?); Key storage; Revocation (revoking published keys; determining whether a published key is still valid).

Some (poor) Key Distribution Schemes

Eve places K_E in a public place (on a server), with message specifying that it is *Alice's key*.

or, Alice sends her public key with the message to Bob:

$$Alice \rightarrow Bob : K_A, \{\{T_A, M, Bob\}_{sKA}\}_{K_B})$$

or, Alice gives her public key to Bob in person [out of band]:

$Alice \rightarrow Bob :$

```
 00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
e8:35:1c:9e:27:52:7e:41:8f    65537 (0x10001)
```

Public Key Certificates

How does one come to trust that a public key K_A is owned by principal Alice?

Have *Certification Authorities* issue *public key certifiates* that associate a principal's name with their public key.

- K_T: widely known public key owned by trusted third party Trent.
- Assume that we trust statements that K_T signs.
- Alice (securely) presents a public key K_A that she owns to Trent and asks for a signed certificate:

$$cert = \{Alice, K_A, validityPeriod\}_{sK_T}$$

- Alice presents certificate with her signed message

$$A \rightarrow B : cert, \{a\ message\ from\ Alice\}_{sK_A}$$

- If Bob knows/trusts K_T he can confirm that message is from Alice.

Using Certificates: Example

Trusted **Certification Authority (CA):** certifies the authenticity of users (that a public key is owned by a particular principal).

CA digi-sign generates key-pair K_B^{-1},K_B and K_B is widely known.

Systems Manager for `cs.ucc.ie` (Dave) generates key-pair K_D^{-1}, K_D; *securely* sends request *issue certificate for* $[Dave, \ldots, K_D]$ to digi-sign.

CA *digi-sign* generates a certificate ($cert_D^B$).

Name:	Dave . . .
Issuer:	digi-sign
. . .	. . .
RSA Public Key:	K_D
Signature:	$RSA(K_B^{-1}, h(\text{above fields}))$

Dave sends mail message M to Simon: $[M, RSA(K_D^{-1}, h(M)), cert_D^B]$
Simon, assuming he knows K_B can use $cert_D^B$ to verify K_D and then verify the signature on the email message.

Certification Authority

Certification Authority (CA) guarantees (to some degree) the relationship between a public key and an end entity. CA may perform out-of-band (telephone, face-to-face) verification of the person's identity.

Anything (owning a public key) can have a certificate: person; role (eg, system administrator); organization; pseudonym; piece of hardware or software; an account (bank or credit card).

CA may certify other attributes of the key: a bona fide business; can trust their web server; trust code they write; over 18; ...

Using Certificates: Example

Everyone could get certificates from CA in this way, but it's not feasible to expect a single (worldwide) CA to issue all certificates.

Set up a hierarchy of CAs, eg, Dave could act as a CA for UCC CS.

Dave issues certificate $cert_S^D$ for Simon's public key K_S.

Name:	Simon . . .
Issuer:	Dave
. . .	. . .
RSA Public Key:	K_S
Signature:	$RSA(K_D^{-1}, h(\text{above fields}))$

Simon sends mail message M to secretary as

$$[M, RSA(K_S^{-1}, h(M)), (cert_D^B, cert_S^D)]$$

Given well known K_B, secretary can use certificate chain $(cert_D^B, cert_S^D)$ to validate K_S and verify signature.

Certificate chains can be stored locally or sent with every message.

Some Public Key Protocols: A Simplified Sketch of SSL

Alice uses an SSL-style protocol to connect to the web-server Bob.

Msg1	$A \rightarrow B$	$I'm\ A,\ handshake\ info$
Msg2	$B \rightarrow A$	$I'm\ B,\ handshake\ info,\ certificate\ chain\ for\ K_B$
Msg3	$A \rightarrow B$	$\{A, N_A, K_{AB}, \ldots\}_{K_B}$
Msg4	$B \rightarrow A$	$\{B, N_A, N_B \ldots\}_{K_{AB}}$
Msg5	$A \rightarrow B$	$\{A, N_B + 1\}_{K_{AB}}$
...	$A \leftrightarrow B$	$\{data, etc\}_{K_{AB}}$

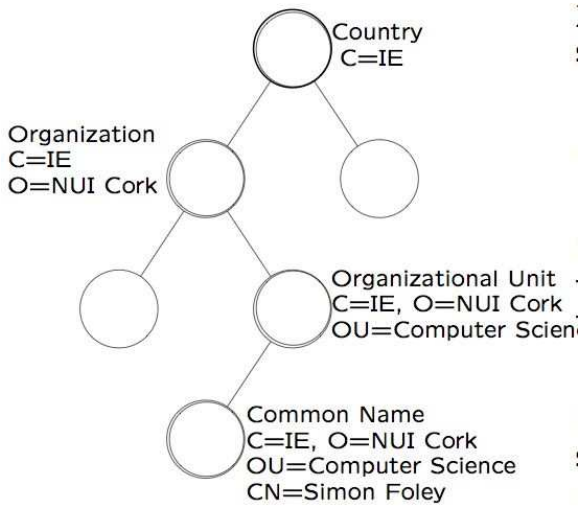
In this protocol, we assume that Alice knows and trusts the public key of the CA that signed B's certificate.

X509 Public Key Infrastructure

X500 Names for X509 Certificates

We need to be able to uniquely name principals if we are to support universal certificates that bind principal (identities) to their public keys.

X500: A proposed hierarchical naming scheme. The *intention* was that X500 Distinguished Names can be used to give a guaranteed name for everything in the world!



C=country,
S=state,
L=locality,
O=organization,
OU=organization unit,
CN=common name,
...

X500 DNs are used to name principals in X509 certificates.

Some of the fields in an X509 Certificate

- Version
- Serial number (with issuer, uniquely identifies certificate).
- Signature Algorithm, for example sha1RSA.
- Issuer (name)
- Valid from (typically when issued)
- Valid to (expiry date)
- Subject (X500 distinguished name of subject)
- public key of subject
- ...

X509 Certificate Example


```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
       OU=Certification Services Division, CN= Thawte Consulting
       Validity
           Not Before: Jul  9 16:04:02 2007 GMT
           Not After : Jul  9 16:04:02 2009 GMT
       Subject: C=IE, ST=Munster, L=Cork, O=NUI Cork,
                OU=Computer Science, CN=Simon Foley
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
            ......
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
            ......
```

X509 Certificate Creation in Java

Java KeyStore stores and manage keys and certificates for a principal.
Includes principal's private keys and public keys (certificates) that it trusts.
keytool provides command-line utility to access and manage user keystore.

```
keytool -genkey -keystore mykeystore -alias SimonsDSA \
        -keyalg DSA -keysize 1024 -storepass spasswd \
        -sigalg DSA -validity 90 -keypass kpasswd \
        -dname "CN=Simon Foley, OU=Department of Computer Science,\
O= College Cork, L=Cork, C=IE" |
```

keytool -genkey generates a new key pair (given parameters)

The key pair is used to create a private key entry in the keystore.

The public key is placed inside a self-signed X509 certificate and is 'trusted'.

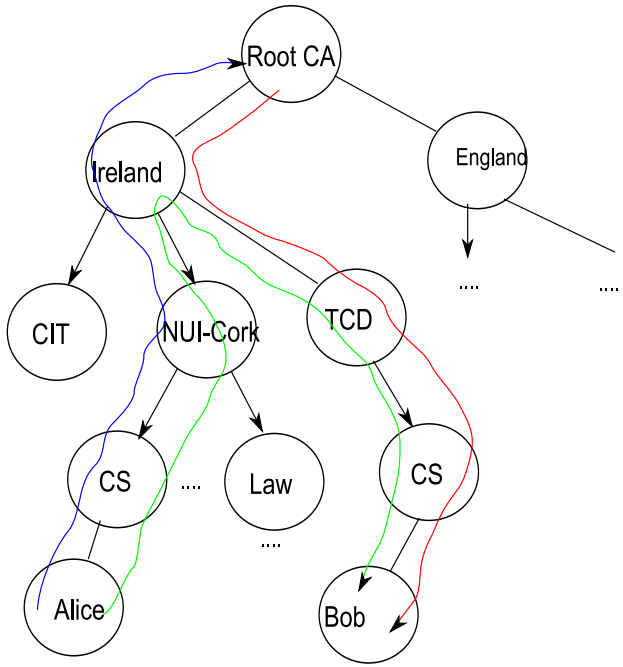
X509 Certificate Creation in Java


```
keytool -list -keystore mykeystore -alias simonsDSA -storepass spasswd -v
  Alias name: simonsdsa
  Creation date: Mon Jan 21 11:47:39 GMT+00:00 2002
  Entry type: keyEntry
   Certificate chain length: 1
   Certificate [1]:
  Owner: CN=Simon Foley, OU=Department of Computer Science,
              O=University College Cork, L=Cork, C=IE
   Issuer : CN=Simon Foley, OU=Department of Computer Science,
              O=University College Cork, L=Cork, C=IE
   Serial number: 3c4bffdb
   Valid from: Mon Jan 21 11:47:39 GMT+00:00 2002
            until : Sun Apr 21 11:47:39 GMT+00:00 2002
    Certificate fingerprints :
         MD5: 39:E3:3F:C1:9F:FF:4C:39:72:7B:E7:90:77:C1:7B:36
         SHA1: 78:C7:7C:AE:21:00:A3:9A:36:A2:82:71:5C:4C:EF:EB:A3:8A:E7:49
```

(Originally) Proposed X509 Public Key Infrastructure (PKI)

Certificate Authority issues certificates:

- Root CA,
- CA hierarchy, leading from root CA. CA can delegate its authority to other intermediate CAs and/or principals. All principals (leaves in hierarchy) trust all CAs to be honest and competent.



How does Alice learn Bob's public key?

Leaf principals know their nearest CA.

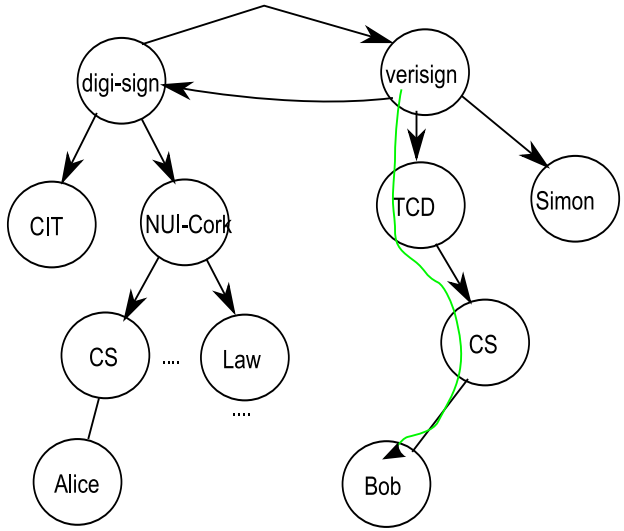
Each (intermediate) CA certifies its parent CA in reverse direction

Route from Alice to Bob provides certificate chain: *Alice trusts CA CS (in UCC) who trusts CA Ireland who trusts CA TCD who trusts CA CS(in TCD) who says Bob's key is* K_B.

X509 PKI in Practice

No clear plan on how to organize X500 directory in reality: hierarchical naming suits military and government but does not work for businesses or individuals.

In practice, we have just one hierarchy per organization and/or have many commercial CAs that sign certificates for each other and for other principals such as UCC and for individuals.



How does Alice learn Bob's public key?

Alice knows and trusts a number of root CAs: eg, verisign.

Route from CA Alice trusts to Bob provides certificate chain.

Some CAs only certify end-principals: note the danger of delegating to customer's CA (must be trusted to delegate).

t

X509 Certificate Extensions

While version, serial number, signature algorithm, issuer, validity, subject and public key are standard fields, an X509 certificate provides extension fields including,

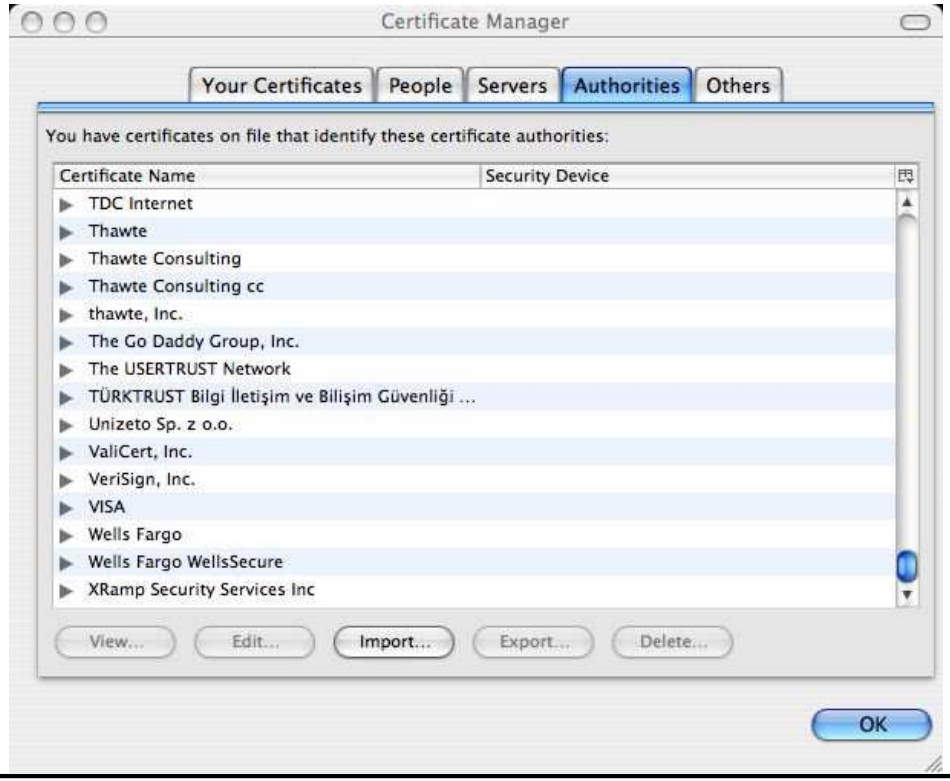
- Subject Alternative name (eg, email address, etc)
- Basic constraints: is the subject a CA, an end-entity, or max length of chain from CA to end-entity?
- Name constraints: names that CA has authority over.
- Key usage: how the key may be used (eg for client/server web authentication), ...

Use your browser to look at the certificate chain at `https://www.microsoft.com`.
CA root GTE CyberTrust Global Root (CN) issues a certificate stating that Microsoft Internet Authority (CN) has key ... and acts as an (intermediate) CA. This intermediate CA issues a certificate for (intermediate CA) Microsoft Secure Server Authority (CN), which in turn issues a certificate for www.microsoft.com.

X509 Certificates in Practice on the Web

While not part of the original X509 standard, it is now common to include a Web URL in the Common Name attribute of a certificate.

Most current web-browsers can look for this information when authenticating an SSL connection to a web-site. Web browsers come with a number of trusted CAs pre-loaded.



Certificate Revocation

CA may revoke certificate, for example, if the subject's private key becomes compromised.

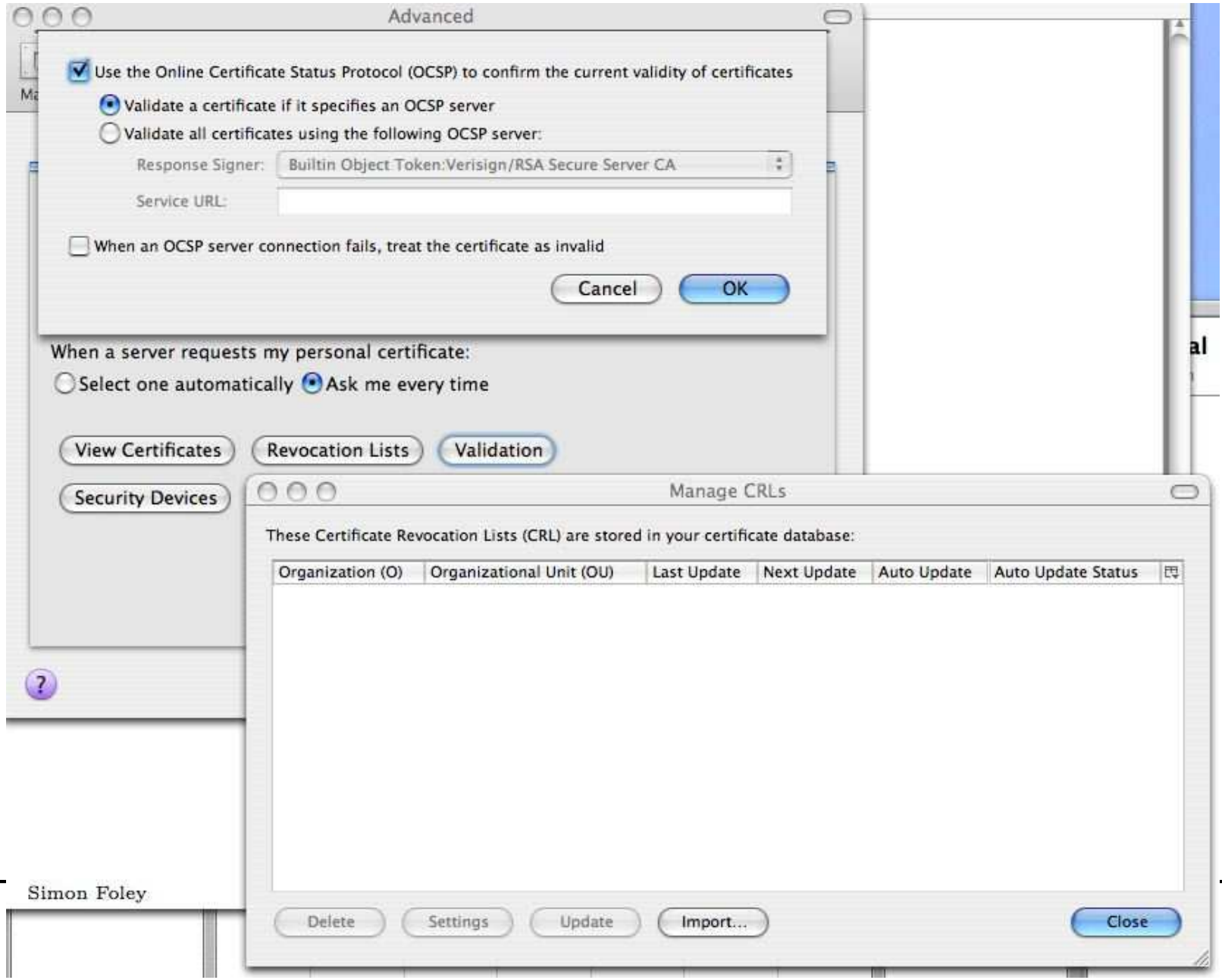
Certificate revocation list (CRL) gives a list of revoked certificate (serial numbers), signed and distributed by the CA. CRL also includes, information on when list was issued and when the next list will be issued.

CRL size increases over time until it reaches a stable size (do not include expired certificates in list). A Delta CRL just includes 'recent' revocations.

Alternatively, Online Certificate Status Protocol is used to confirm the validity of certificates. Certificate can include address of OCSP Server, or consult user-specified server.

More recent browsers support CRL/OCSP-based revocation.

Revocation in Firefox



About VeriSign
Company Information
Investor Relations
News
Press Release Archives
Awards
Logos and Copyright
Global Events
Careers at VeriSign
Customer Case Studies
Government Relations
VeriSign Labs
VeriSign Blogs
Worldwide Sites
Contact Us

News

VeriSign Exceeds 2 Billion Certificate Checks in a Single Day

In Less Than a Year, VeriSign Doubles Its Daily Tally of Online Certificate Status Protocol Checks

MOUNTAIN VIEW, CA – (Marketwire) -- April 15, 2010 – VeriSign, Inc. (NASDAQ: VRSN), the trusted provider of Internet infrastructure services for the networked world, today reported it surpassed 2 billion Online Certificate Status Protocol (OCSP) checks in a single day. The milestone, which was reached on Monday, April 12, represents an average of 23,000 online validations a second.

Passing the 2 billion OCSP check benchmark also highlights VeriSign's ability to keep pace with the demands of the Internet. In July of 2009, VeriSign announced it had surpassed 1 billion OCSP checks in a day. In less than a year, the volume of certificate checks handled by VeriSign has doubled.

A key link in the online security chain, OCSP offers the most timely and efficient way for Web browsers to determine whether a Secure Sockets Layer (SSL) or user certificate is still valid or has been revoked. Generally, when a browser initiates an SSL session, OCSP servers receive a query to check to see if the certificate in use is valid. Likewise, when a user initiates actions such as smartcard logon, VPN access or Web authentication, OCSP servers check the validity of the user certificate that is presented. OCSP servers are operated by Certificate Authorities, and VeriSign is the world's leading Certificate Authority.

"The ability to support this level of volume requires constant investment in a world-class infrastructure and a strong commitment to best practices," said Tim Callan, vice president of product marketing at VeriSign. "As such, VeriSign continues to be uniquely positioned to enable secure transactions well beyond the 2 billion milestone."

As the most respected and trusted SSL authority on the Web, VeriSign is the EV SSL Certificate provider of choice of 17,000 online businesses. In fact, 93 percent of the Fortune 500 and 97 of the world's 100 largest SSL-using banks secure their sites with SSL Certificates sold by VeriSign and its subsidiaries. To learn more

Contact Us

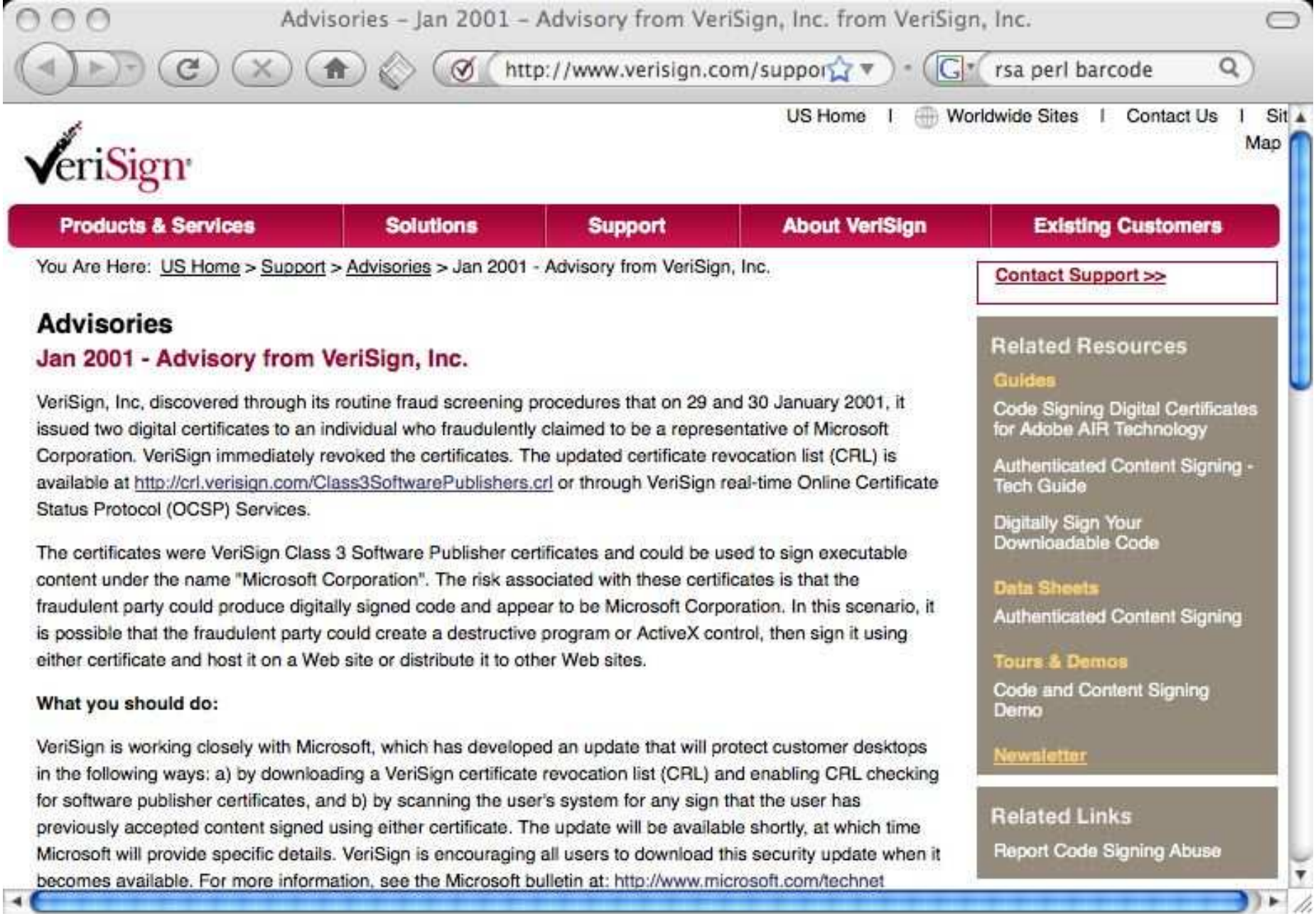
For media inquiries, please contact us at 650-426-5558 or at pr@verisign.com

Stay Informed RSS

Read the VeriSign Blogs

VeriSign Collaborates With Industry Leaders to Bring Trust to the Cloud

VeriSign Exceeds 2 Billion Certificate Checks in a Single Day


Advisories – Jan 2001 – Advisory from VeriSign, Inc. from VeriSign, Inc.
http://www.verisign.com/suppor
rsa perl barcode
US Home | Worldwide Sites | Contact Us | Sit Map
VeriSign
Products & Services
Solutions
Support
About VeriSign
Existing Customers
You Are Here: US Home > Support > Advisories > Jan 2001 - Advisory from VeriSign, Inc.
Contact Support >>
Advisories
Jan 2001 - Advisory from VeriSign, Inc.
VeriSign, Inc, discovered through its routine fraud screening procedures that on 29 and 30 January 2001, it issued two digital certificates to an individual who fraudulently claimed to be a representative of Microsoft Corporation. VeriSign immediately revoked the certificates. The updated certificate revocation list (CRL) is available at http://crl.verisign.com/Class3SoftwarePublishers.crl or through VeriSign real-time Online Certificate Status Protocol (OCSP) Services.
The certificates were VeriSign Class 3 Software Publisher certificates and could be used to sign executable content under the name "Microsoft Corporation". The risk associated with these certificates is that the fraudulent party could produce digitally signed code and appear to be Microsoft Corporation. In this scenario, it is possible that the fraudulent party could create a destructive program or ActiveX control, then sign it using either certificate and host it on a Web site or distribute it to other Web sites.
What you should do:
VeriSign is working closely with Microsoft, which has developed an update that will protect customer desktops in the following ways: a) by downloading a VeriSign certificate revocation list (CRL) and enabling CRL checking for software publisher certificates, and b) by scanning the user's system for any sign that the user has previously accepted content signed using either certificate. The update will be available shortly, at which time Microsoft will provide specific details. VeriSign is encouraging all users to download this security update when it becomes available. For more information, see the Microsoft bulletin at: http://www.microsoft.com/technet
Related Resources
Guides
Code Signing Digital Certificates for Adobe AIR Technology
Authenticated Content Signing - Tech Guide
Digitally Sign Your Downloadable Code
Data Sheets
Authenticated Content Signing
Tours & Demos
Code and Content Signing Demo
Newsletter
Related Links
Report Code Signing Abuse

Null prefix certificates and X509 Implementation (pre July 2009)

In C, a string is a sequence of characters terminated by a null value.

0x44 ('D')	0x41 ('A')	0x54 ('T')	0x41 ('A')	0x00 ($\backslash 0$ NULL)

In Pascal, a string implementation uses the length of the char sequence.

A malicious user, requests a certificate from CA for domain

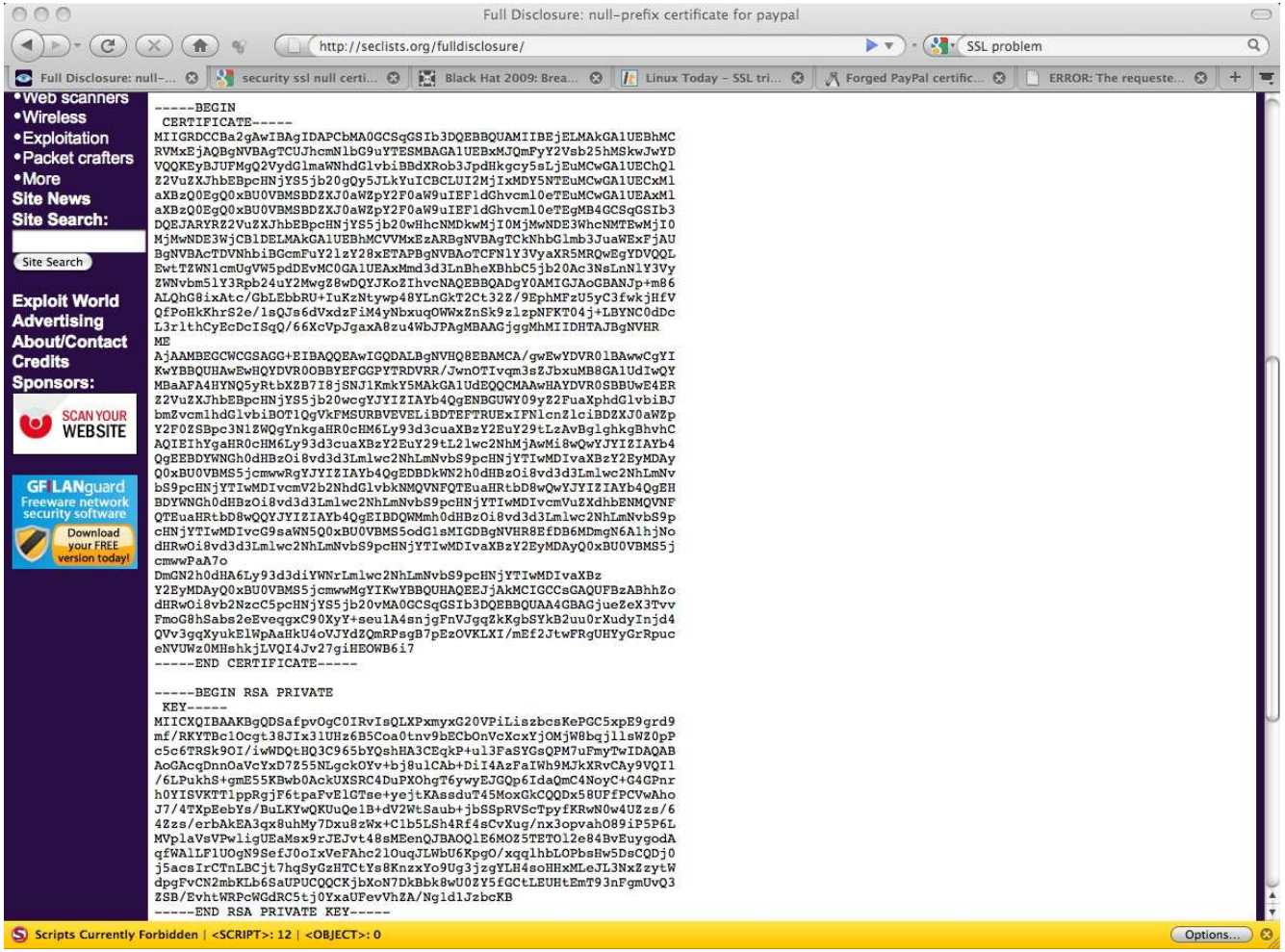
$$\texttt{paypal.com}\backslash 0\texttt{.attacker.com}.$$

CA, which processes all data in the sequence (like Pascal), confirms that user owns the domain `attacker.com` and issues/signs a certificate.

Attacker creates a website that looks like paypal, and uses this certificate to prove its identity. Alternatively, attacker carries out a man in the middle attack between a victim and the paypal site.

When checking the domain, Mozilla-based implementations of SSL (prior to July 2009), treated the URL (in the common name of the certificate) as a C string and believes that the owner of the public key is `paypal.com`!

Null Prefix certificate for paypal



Pretty Good Privacy

Insecure Email using RFC 2554 SMTP

Conventional email does not provide authentication of sender: anyone can interact with an SMTP server and forge an email address.

```
> telnet smtp.ucc.ie 25
helo cosmos.ucc.ie
mail from: <biffo@gov.ie>
rcpt to: <s.foley@cs.ucc.ie>
data
.......
```

In practice, this is not necessarily foolproof since it may be possible for the administrator of the SMTP server to use the log of external connections to deduce the IP address of the system that made the request.

And it may be illegal.

Secure Email using Pretty Good Privacy

PGP provides support for secure and authentic exchange of email.

A sender A signs an email with their public key (K_a), while the recipient B can use the sender's key to confirm the origin/signature of an email message. The public keys can also be used to perform exchange of a symmetric session key that can be used to encrypt the email message for secrecy.

Can be used in a number of configurations, for example,

$$A \rightarrow B : \{email\}_K, \{\{K, h(email), \ldots\}_{sKa}\}_{K_b}$$

where K is a symmetric session key and K_b is the recipient's public key.

We assume that B knows that the public key K_a is owned by the holder of the email address `alice@cs.ucc.ie` (and similarly for K_b).

This provides email authentication, integrity and secrecy, and digital signature.

Pretty Good Privacy: Public Key Certificates

If the recipient of an email message does not know the sender's public key, then in order to authenticate the sender they will need a public key certificate (for the sender).

PGP Certificate: denote by $CERT_{K_A}^{K_B}$ which provides the email address for the owner of public key K_A as asserted by K_B.

Field	Description
A	subject email address
K_A	subject public key
K_B	signer's public key
signature=$RSA(K_B^{-1}, h(\langle A, K_A, T\rangle))$	

A PGP certificate is a binding between an email address an a public key. Public keys are the unique identifiers (not the names).

Web of Trust: (unlike X509), no formal certification paths, no central authority, individuals sign each others' keys.

Web of Trust Example

Alice has a public key K_A and self-signs it to create

$$CERT_{K_A}^{K_A} = (\texttt{alice@cs.ucc.ie}, K_A, K_A, \ldots)$$

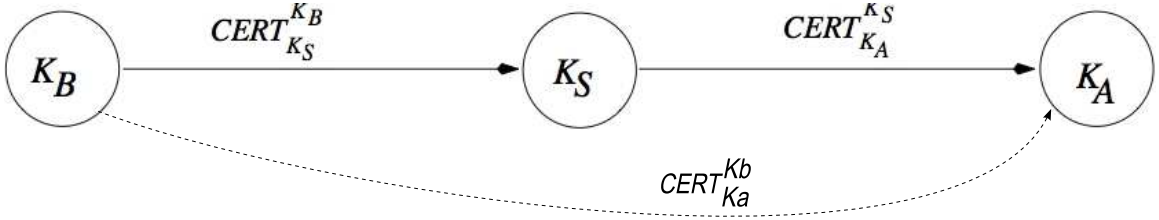
Simon (public key K_S) is happy with the validity of this public-key (eg got in person) so he signs a copy, giving $CERT_{K_A}^{K_S}$ ("adds to his key-ring").

In signing the certificate, the owner of K_S is asserting that he trusts anything that is signed by the key K_A.

If Bob (K_B) has $CERT_{K_S}^{K_B}$ and is presented with $CERT_{K_A}^{K_S}$ he validates signature on $CERT_{K_A}^{K_S}$. If satisfied, he may add K_A to his key-ring, signing a new certificate $CERT_{K_A}^{K_B}$

Bob can make this new certificate $CERT_{K_A}^{K_B}$ publically available, etc.

Chain of Trust:

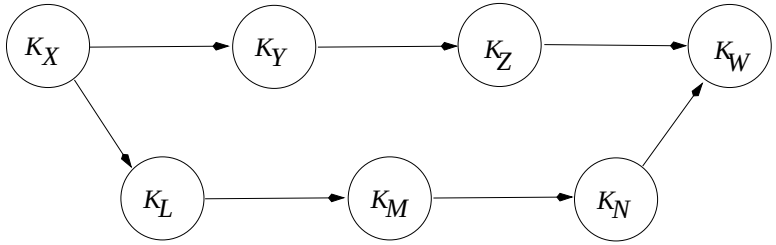


Transitive Chains of Trust

K_X completely trusts K_Y $(CERT_{K_Y}^{K_X})$
K_Y completely trusts K_Z $(CERT_{K_Z}^{K_Y})$
K_Z completely trusts K_W $(CERT_{K_W}^{K_Z})$

X trusts what Y signs, and Y trusts what Z signs, therefore should X really trust what Z signs (and add K_W to key-ring)?

Trust is not necessarily transitive. For example, X may know something about K_W which makes it untrustworthy.



Strategy: if trust is uncertain, look for additional trust chains

Trust management in PGP

With each key, the principal keeps a record of the degree to which it trusts the key with regard to the certificates that the key might sign

Completely Trusted. If any other key is signed by this key then add the new key to key-ring. Alice is saying that she trusts Bob to vouch for the validity of any key.

Marginally Trusted. Certificate (key) must be signed for by two or more other keys before added to key-ring. Alice does not trust Bob very much and needs to have claims about keys corroborated by others.

Untrusted. Do not use this key in determining whether other keys can be added to key-ring. Alice does not trust Bob to vouch for any key at all!

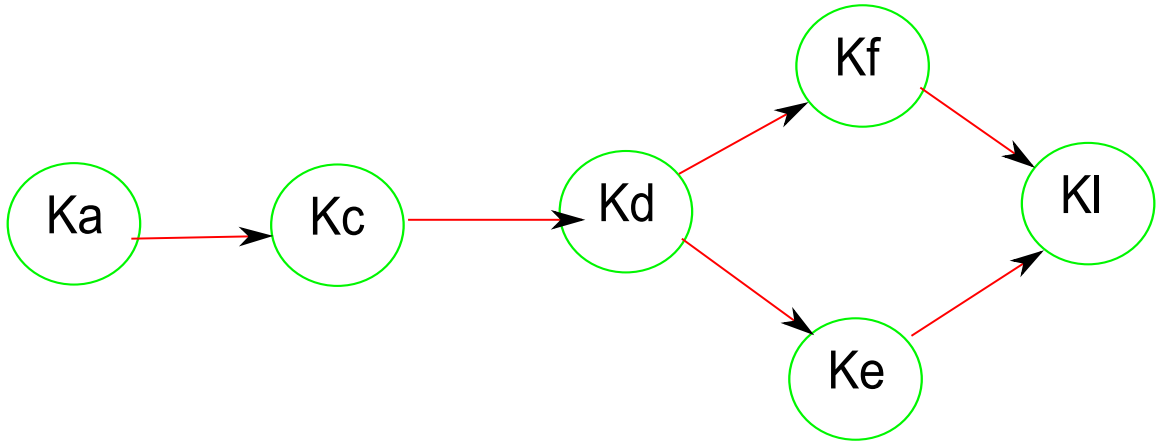
Unknown. A level of trust can not be determined for this key.

Finding Chains of Trust in PGP

Principal Ka has the following keys (and degree to which the principal trusts anything signed by the key) on his key-ring.

Kc	Kd	Ke	Kf
trusted	trusted	marginal	marginal

The following Graph of certificates is found that connects Ka to Kl;



Ka trusts what each key signs and in the case of marginally trusted keys (anything signed by Kf and Ke) it can find corroborating certificates. Thus, Ka can believe that Kl is, for example, email address is `liam@cs.ucc.ie`.

Finding Chains of Trust

There are a number of public databases available on the Web which will find chains of trust from a source key to destination key.

Since chains of *certificates* are returned then its not necessary to trust the server, since the validity of the chain can be confirmed by the requester.